اونيۏرسيتي مليسيا ڤهڠ السلطان عبد الله
UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

BCS3153

SOFTWARE EVOLUTION & MAINTENANCE

SEMESTER II 2024/2025

Project : KAFA Management System

Group : 6

Lab Section : 01B

Lecture : Pn. Ku Saimah Binti Ku Ibrahim

Student Detail:

Name	Student ID
Umairah Shuhada Binti Ahmad	CB22090
Gan Jun Wei	CB22106
Bagaber AbdulRaheem Mohammed Sheikh	CB22016
Nur Alya Syakirah Binti Nasarudin	CB22141

TABLE OF CONTENT

1. Introduction

1.1. Project Overview

The KAFA Management System (KMS) is a web-based application developed to streamline and modernize the management of KAFA schools. It is designed to serve multiple user roles including KAFA Admins, Teachers, Parents and MUIP Admins, providing a centralized platform to handle both administrative and academic operations efficiently. The system aims to enhance coordination, data accessibility and communication among all stakeholders, ultimately improving the overall management and educational experience within KAFA institutions.

KMS is structured into five main modules: Manage Account, Manage Schedule, Manage Result, Manage Report and Manage Bulletin. Each module plays a vital role in ensuring that various school processes are organized and executed effectively. Developed using the Laravel framework and based on the Model-View-Controller (MVC) architecture, the system is designed for scalability, maintainability and security. With a focus on user-friendliness and structured access control, KMS offers a reliable digital solution for performance tracking and school-community engagement.

1.1.1 Manage Profile

This module involves four main user roles: KAFA Admin, Teachers, Parents, and MUIP Admin. The Manage Profile module allows all users to view and update their personal profile information after registering and logging into the system. Users can modify fields such as full name, contact number, email address, and gender, as well as change their password. The module also supports identity card verification and now includes the ability to upload a profile picture. These features ensure that users maintain accurate and personalized records within the KAFA Management System, while enhancing identity verification and overall system usability.

1.1.2 Manage Schedule

This module involves three main roles: KAFA Admin, Teachers, and Parents. The KAFA Admin is responsible for creating classes for students and assigning teachers to those classes. After a class is set up, the assigned teacher can create activities and add them to the class schedule, which will be used throughout the academic year. Teachers can also manage the class schedule by updating existing activities, adding new ones, or removing them as needed. Parents will have access to view their child's class schedule to stay informed about their child’s KAFA activities.

1.1.3 Manage Result

The Manage Result involves Teachers and Parents as their primary actors, this module allows for easy communication regarding student performance. Teachers can update and establish grades, while providing helpful feedback on the students' progress. Parents are provided access to view their child's grades, alongside the related feedback. This feature creates openness and allows Parents to stay actively involved in their child's learning and education process. Whenever the Teachers update or include grading, KAFA admin will have sanctioned it before finalizing. This is for the sake of accuracy and consistency of students' assessment before finalizing to confirm the integrity of the data on students' assessments.

1.1.4 Manage Bulletin

The Manage Bulletin module serves as the central communication hub within the KAFA Management System. It allows authorized users including KAFA Admins, Teachers and MUIP Admins to create, manage and publish announcements or notices that are visible to the entire school community including parents. To maintain appropriate content and messaging, teacher-submitted bulletins require approval from the KAFA Admin before being published, while MUIP Admins can post directly without approval. This module ensures that important updates, events and information are effectively communicated and easily accessible to all users through a centralized digital bulletin board.

1.2. Module Delegation

The manage profile module allows KAFA Admins, Teachers, Parents, and MUIP Admins to view and update their personal information after login. Users can edit details such as name, contact number, email, gender, and password. The module also supports identity card verification and uploading a profile picture, ensuring accurate user records and enhancing system usability.

The manage schedule module coordinates KAFA Admins, Teachers, and Parents. KAFA Admins create classes and assign teachers. Teachers manage class activities by creating, updating, or removing them in the schedule used throughout the school year. Parents can view their child's schedule to stay updated on KAFA activities.

The manage result module enables KAFA Admins, Teachers and Parents a common platform to work together in the management of students' academic performance. Each of the three roles has specific roles to enable appropriate result processing, validation, as well as open visibility of student progress.

The manage bulletin module enables KAFA Admins, Teachers and MUIP Admins to post announcements visible to all users including parents. Teacher posts require KAFA Admin approval while MUIP Admins can publish directly. It ensures effective and centralized communication of important school updates.

MODULE	EXPLANATION	PERSON IN CHARGE
Manage Profile	1. <u>KAFA Admin</u> KAFA Admin can view and update their own profile details, including name, contact, email, and password. They are also responsible for registering teachers and managing their account information, including uploading identity verification documents and assigning default credentials. 2. <u>Teacher</u> Teacher users can manage their profile by editing personal information such as contact number, email, and password. They may also upload identity verification documents to ensure their records are accurate and comply with the system's requirements. 3. <u>Parent</u> Parent users are able to update their personal profile and contact information after registration. They can change their password and upload identification documents when necessary. These features help maintain accurate records and enhance communication between the school and the parents. 4. <u>MUIP Admin</u> MUIP Admin has similar access to manage and update their profile information, ensuring that their account remains accurate and up-to-date. This supports proper	BAGABER ABDULRAHEEM (CB22016)

	oversight and identification within the system.	
Manage Schedule	1. <u>KAFA Admin</u> KAFA Admin is responsible for creating classes, assigning students and teachers to each class. They are able to edit and delete the classroom if it is no longer available. KAFA Admins have to ensure each class is properly set up, enabling the smooth operation of KAFA activities throughout the academic year. 2. <u>Teacher</u> Teachers are responsible for managing the activities within their assigned classes. Once assigned by the KAFA Admin, teachers can create and schedule activities for the class, forming a complete schedule to be used for the school year. They are also able to update the class schedule as needed, which includes modifying existing activities, adding new ones, or deleting outdated ones to keep the schedule relevant and up to date. 3. <u>Parent</u> Parents have view-only access to their child's class schedule. This allows them to stay informed about the planned KAFA activities and any updates made by the teacher. By regularly checking the schedule, parents can support their child's participation and engagement in KAFA-related learning and	Gan Jun Wei (CB22106)

	events.	
Manage Result	1. <u>KAFA Admin</u> KAFA Admin can establish and conduct academic sessions, view the results submitted by the students, as well as approve or reject them to maintain the assessment records free from errors and inaccuracies. 2. <u>Teacher</u> Teacher are able to input and edit students' marks, select subjects for test, and provide feedback through the result management and assessment modules, which result in accurate performance tracking. 3. <u>Parent</u> Parent can view and access their child's approved results, including session details, subject-wise marks, grades, and teacher comments, through which they stay updated and engaged with the education of their child.	Umairah Shuhada Binti Ahmad (CB22090)
Manage Bulletin	1. <u>Teacher</u> Teachers can create bulletin announcements by filling in the notice form. However, their announcements require approval from the KAFA Admin before being published on the bulletin board. 2. <u>MUIP Admin</u> MUIP Admin can create and publish bulletin announcements directly without any approval process. Their announcements are immediately visible to all users in	Nur Alya Syakirah Binti Nasarudin (CB22141)

	the system including parents. 3. <u>KAFA Admin</u> KAFA Admin reviews and approves or rejects bulletin announcements submitted by teachers. Once approved, the announcement is published and becomes visible to all users. KAFA Admin can also directly create and publish bulletins without needing approval. 4. <u>Parent</u> Parents have access to view approved bulletin announcements published by Teachers, KAFA Admins and MUIP Admins. This feature helps ensure that parents stay informed about important school activities and official communications	

1.3. Report Organization

The report comprises three different main section: Section 1- Introduction, Section 2- Implementation of Change, and Section 3- Implementation of Project. The introduction is the first section which comprises project overview, module delegation and project organization. Here the project description and details are discussed to provide the context or the idea of the project. Next is Section 2, which is essentially seeking changes proposed for the project. Here, it will define type of changes being asked to the system by the Change Request form. This is required to determine the maintenance activity. Lastly is the implementation of the project in the final section. Here, it discusses all the project and tools used for this project.

2. Change Implementation [Individual]

2.1. Types of maintenance chosen

Module	Type of Maintenance Category	Explanation (Maintenance Activity)	Person In Charge
Manage Profile	Corrective Maintenance (IC Number Validation Fix)	Resolved the issue where the IC Number field accepted letters or symbols by enforcing numeric-only input, ensuring accurate and valid identity records during profile update or registration.	Bagaber Abdulraheem (CB22016)
	Perfective Maintenance (Profile Picture Upload)	Added a new feature allowing users to upload a profile picture during registration to enhance personalization and improve user identity management.	
Manage Schedule	Preventive Maintenance (Email verification upon registration)	Implemented email verification for new user registration to ensure the email is valid and belongs to the user, while enhancing the security of the	Gan Jun Wei (CB22106)

		system by refraining unverified users from entering the system.	
	Perfective Maintenance (Analytical dashboard for teachers and students in the system.)	Implemented an analytical dashboard showing the current state of teachers and students, and acts as a reminder for KAFA admin to assign teachers and students into the classroom.	
	Perfective Maintenance (Dynamic week-view schedule)	Implemented dynamic buttons enabling parents and teachers to navigate the schedule week by week, viewing past or future weeks with a single click.	
	Corrective Maintenance (The list of session year not link to Teacher when the KAFA Admin added)	The system was corrected to make the recently added session year by the KAFA Admin properly associated with the respective teachers, thus facilitating correct assignment and recovery based on the academic session.	Umairah Shuhada Binti Ahmad (CB22090)

Manage Result	Enhancement Maintenance (Added filter button session year at assessment list page)	The system was enhanced by showing the list of assessments by session year, making it easier for teachers to select assessments to take further action.	
	Enhancement Maintenance (Added message error)	The system was enhanced by showing a 'Not Available' message, reducing user confusion when there are no results and improving user friendliness.	
Manage Bulletin	Enhancement Maintenance (Added filter button and notice info on bulletins)	The system was enhanced by showing the notice's name and date on each bulletin, along with a filter feature to view bulletins by date ranges like Today and Yesterday improving user experience and navigation.	Nur Alya Syakirah Binti Nasarudin (CB22141)
	Enhancement Maintenance (Notice delete button added)	The system was enhanced by enabling Teachers and MUIP Admins to delete their own notices, allowing Teachers to remove both	

		approved and pending posts which improves control over posted content.	
	Enhancement Maintenance (Bulletin board notice download button)	The system was enhanced by adding a download button on the KAFA Admin dashboard, allowing bulletin notices to be downloaded as PDF for easier reference and record-keeping.	

2.2. Manage Profile (Abdulraheem Bagaber CB22016)

Type of Maintenance Chosen 1

2.2.1 Corrective Maintenance

Within the Manage Profile module, corrective maintenance was applied to address a crucial validation flaw involving the **Identity Card Number (IC Number)** input field. Previously, the system allowed users to enter alphabetic characters and special symbols, which violated the system's data format constraint that required a strictly numeric 12-digit IC number, as defined in RQ142-KMS-001.

This flaw posed a significant risk to **data integrity**, especially since the IC number serves as a unique identifier across multiple modules such as login, registration, and profile management. Users could enter values such as 1234abcd5678, which would be accepted and stored in the database, potentially causing issues in user authentication and record retrieval.

The corrected implementation includes both front-end and back-end validation to ensure the field accepts only numerical values (0–9). A real-time input check was added, triggering an error message — *"Identity Card Number must contain only numbers"* — whenever invalid characters are entered. This change ensures users are immediately notified of input errors, improving data quality and user experience.

Additionally, the error message is visually highlighted using a red input border and descriptive text, complying with usability guidelines and improving accessibility. The implementation supports consistent enforcement of the 12-digit numeric format across all user roles (KAFA Admin, Teachers, Parents, MUIP Admin), thereby ensuring **uniformity** and **system-wide reliability**.

By enforcing strict IC Number input validation, this corrective maintenance:

- Prevents the entry of corrupted or malformed IC data
- Ensures compliance with national identification standards
- Reduces the risk of mismatches or failures during login and user lookup operations

2.2.2 Change Request Form

CHANGE REQUEST FORM

Project Description	
Project Name	KAFA Management System
Project ID Number	
Change Description	
Change Request Submitter Name	BAGABER ABDULRAHEEM MOAHMMED
Change Request Name	IC Number Field Validation Correction
Type of CR	☐ Enhancement ☑Corrective
Description of Change	Enforced numeric-only input validation for the Identity Card Number field in the Manage Profile form to prevent users from entering letters or symbols.
Reason of Change	To ensure accurate and valid IC number records, prevent data entry errors, and comply with the required 12-digit numeric format.
Date of Request	2025-06-10

Priority of Change☑	☐ Low	☐ Medium	☑ High	☐ Critical
Approval Description				
Decision ☑	☑ Approved		☐ Approved with Conditions	☐ Rejected
Decision Date	2025-06-11			
Approval Signature			Approval Date : 2025-06-01	

2.2.3 Proposed Change

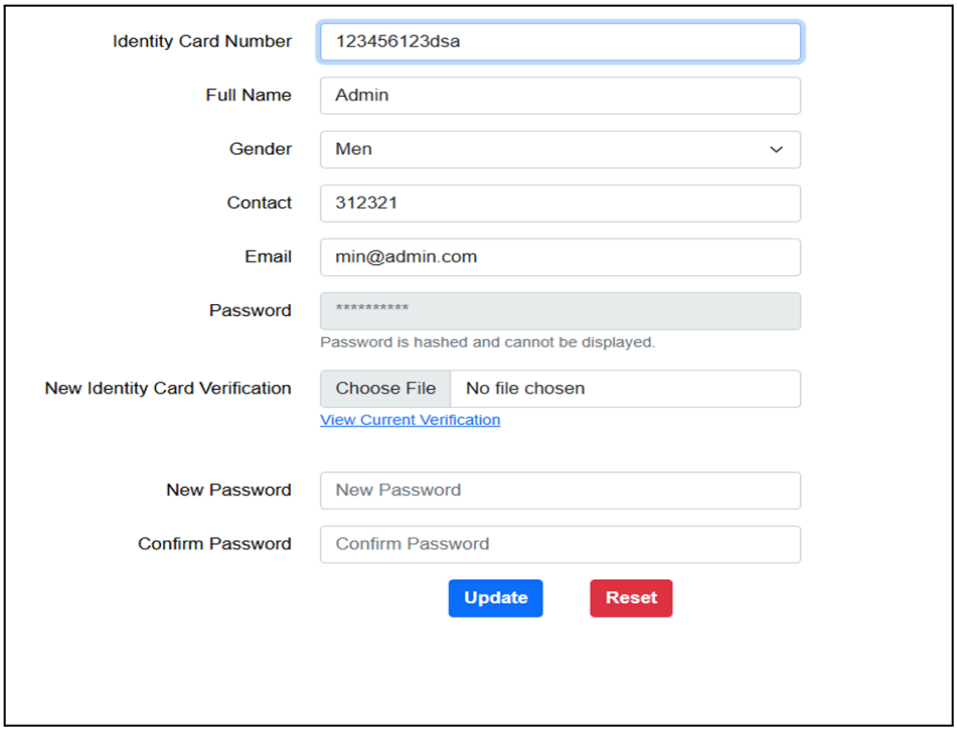
The primary goal of the proposed corrective change is to ensure that the **Identity Card Number (IC Number)** field in the Manage Profile module adheres to a strict numeric-only format. In the previous version of the system, users were allowed to input alphabetic characters and symbols in the IC Number field during profile creation or updates. This behavior contradicted the system's defined requirement, which clearly specifies that the IC Number must contain exactly 12 numeric characters.

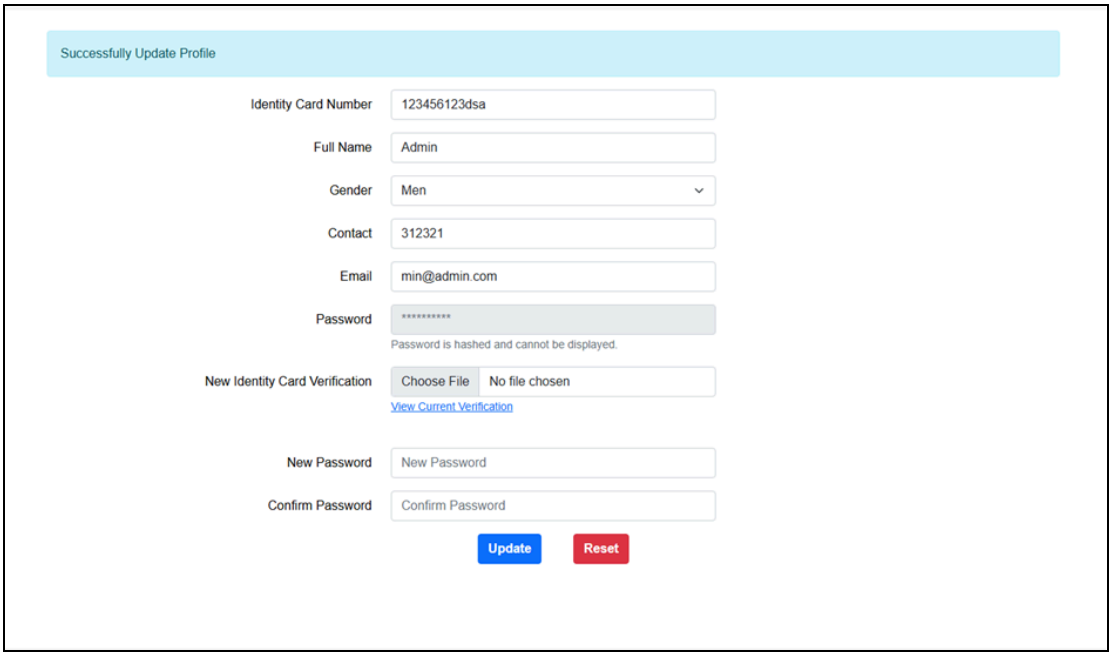
Without proper validation, inaccurate or malformed IC Numbers risked being stored in the database, potentially leading to issues such as login failures, mismatched records, and inefficiencies in user management across the KAFA Management System. To address this, the input logic for the IC Number field was corrected to accept digits only, with all non-numeric characters immediately rejected at the user input level.

The enhancement involved adding input validation on both the frontend and backend. On the frontend, JavaScript was used to prevent non-numeric characters from being typed or pasted into the input field.

2.2.4 Implemented Change

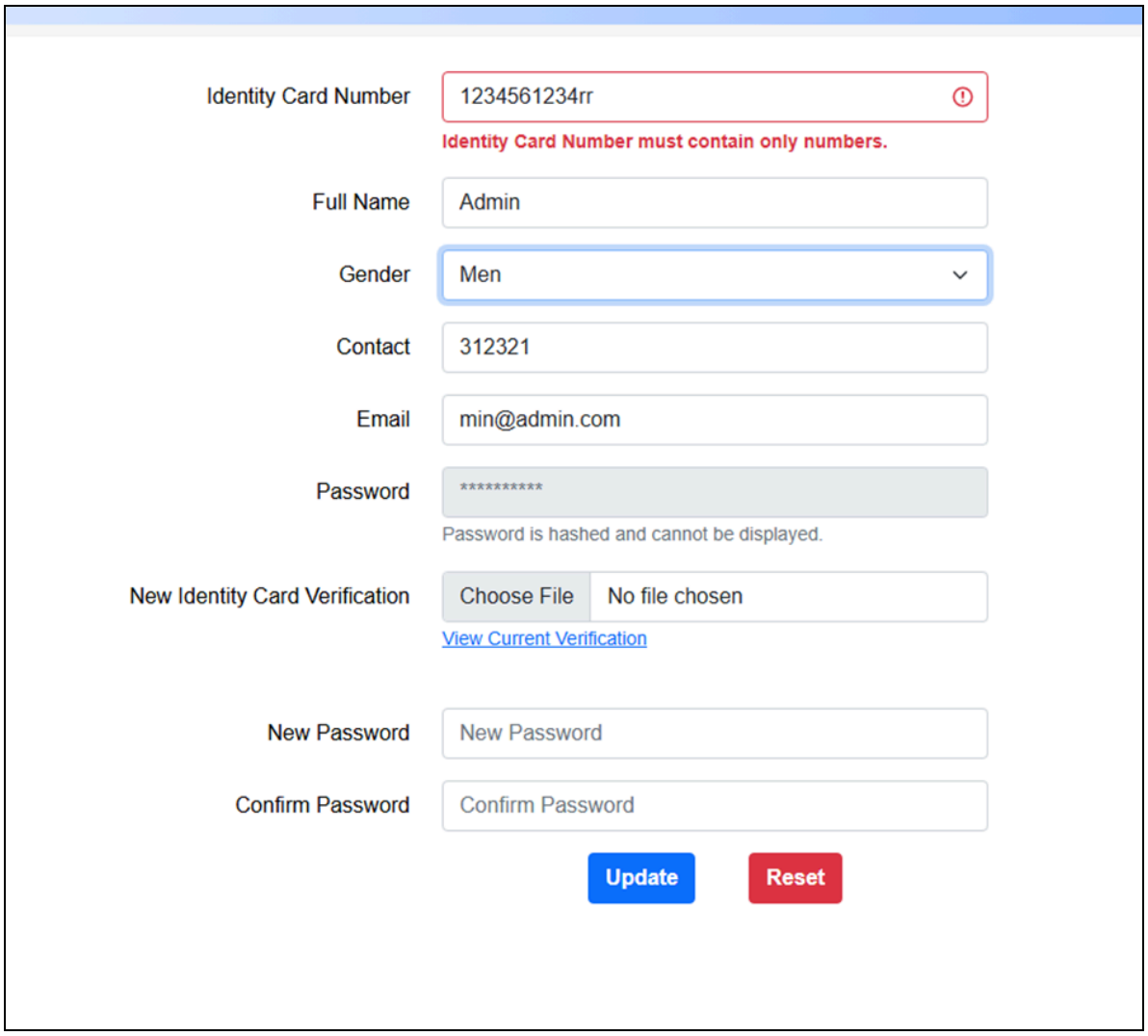
Before Implementation



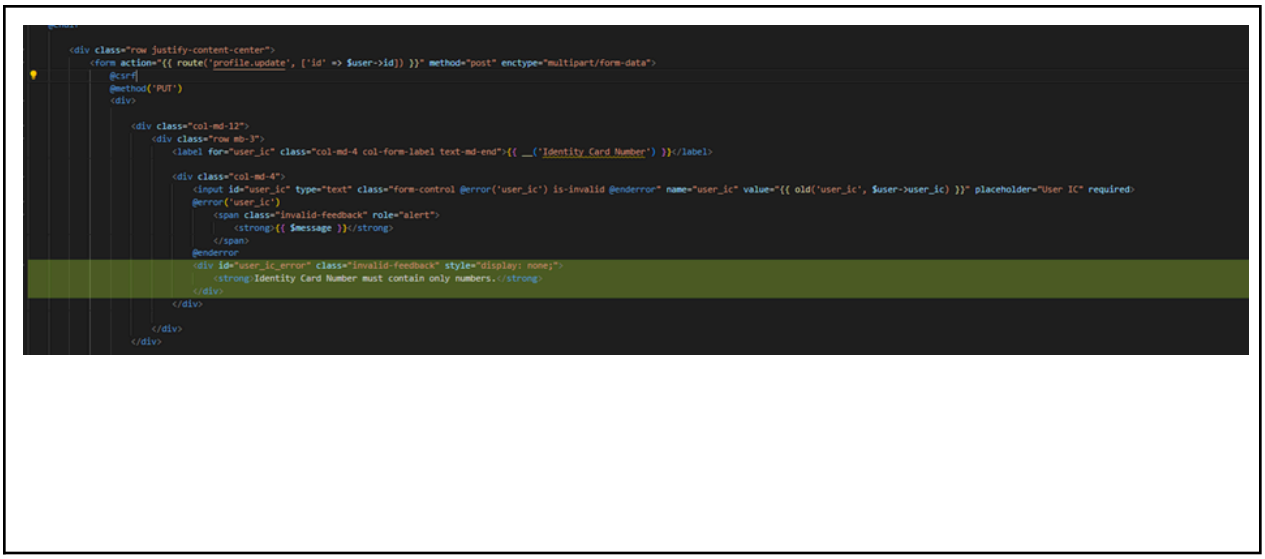



```
<div class="row justify-content-center">
    <form action="{{ route('profile.update', ['id' => $user->id]) }}" method="post" enctype="multipart/form-data">
        @csrf
        @method('PUT')
        <div>

            <div class="col-md-12">
                <div class="row mb-3">
                    <label for="user_ic" class="col-md-4 col-form-label text-md-end">{{ __('Identity Card Number') }}</label>

                    <div class="col-md-4">
                        <input id="user_ic" type="text" class="form-control @error('user_ic') is-invalid @enderror" name="user_ic" value="{{ old('user_ic', $user->user_ic) }}" placeholder="User IC" required>
                        @error('user_ic')
                            <span class="invalid-feedback" role="alert">
                                <strong>{{ $message }}</strong>
                            </span>
                        @enderror



                    </div>

                </div>
            </div>
```

After Implementation




```
<script>
document.addEventListener('DOMContentLoaded', function() {
    const userIcInput = document.getElementById('user_ic');
    const errorDiv = document.getElementById('user_ic_error');
    const form = userIcInput.closest('form');

    userIcInput.addEventListener('input', function() {
        const value = this.value;
        const hasLetters = /[a-zA-Z]/.test(value);

        if (hasLetters) {
            this.classList.add('is-invalid');
            errorDiv.style.display = 'block';
        } else {
            this.classList.remove('is-invalid');
            errorDiv.style.display = 'none';
        }
    });

    // Prevent form submission if IC contains letters
    if (form) {
        form.addEventListener('submit', function(e) {
            const value = userIcInput.value;
            const hasLetters = /[a-zA-Z]/.test(value);

            if (hasLetters) {
                e.preventDefault();
                e.stopPropagation();
                userIcInput.classList.add('is-invalid');
                errorDiv.style.display = 'block';
                userIcInput.focus();

                // Show additional alert to make it clear
                alert('Please enter only numbers in the Identity Card Number field.');
                return false;
            }
        });
    }
});
</script>

@endsection
```



```
<div class="row justify-content-center">
    <form action="{{ route('profile.update', ['id' => $user->id]) }}" method="post" enctype="multipart/form-data">
        @csrf
        @method('PUT')
        <div>

            <div class="col-md-12">
                <div class="row mb-3">
                    <label for="user_ic" class="col-md-4 col-form-label text-md-end">{{ __('Identity Card Number') }}</label>

                    <div class="col-md-4">
                        <input id="user_ic" type="text" class="form-control @error('user_ic') is-invalid @enderror" name="user_ic" value="{{ old('user_ic', $user->user_ic) }}" placeholder="User IC" required>
                        @error('user_ic')
                            <span class="invalid-feedback" role="alert">
                                <strong>{{ $message }}</strong>
                            </span>
                        @enderror
                        <div id="user_ic_error" class="invalid-feedback" style="display: none;">
                            <strong>Identity Card Number must contain only numbers.</strong>
                        </div>
                    </div>

                </div>
            </div>
```

Type of Maintenance Chosen 2

2.3.1 Perfective Maintenance

This change was made to enhance the user experience by allowing users to upload a profile picture during registration. The feature was not originally part of the system, but its addition improves personalization, trust, and visual identity within the KAFA Management System. The change does not correct a fault or adapt to external requirements, but rather improves usability and functionality.

2.3.2 Change Request Form

CHANGE REQUEST FORM

Project Description				
Project Name	KAFA Management System			
Project ID Number				
Change Description				
Change Request Submitter Name	BAGABER ABDULRAHEEM MOAHMMED			
Change Request Name	Implementation of Profile Picture Upload in Registration			
Type of CR	☑ Enhancement ☐ Corrective			
Description of Change	Add a new image upload field to the user registration form, allowing users to upload a profile picture during the account creation process.			
Reason of Change	To improve user personalization and system usability by associating user records with an actual image, making account management and verification easier for both admins and users.			
Date of Request	2025-06-10			
Priority of Change☑	☐ Low	☑Medium	☐ High	☐ Critical
Approval Description				
Decision ☑	☑ Approved		☐ Approved with Conditions	☐ Rejected
Decision Date	2025-06-11			
Approval Signature			Approval Date : 2025-06-11	

2.3.3 Proposed Change

The proposed enhancement to the Manage Account module introduces the ability for users to **upload a profile picture during the registration process**. This feature supports improved user identification and personalization, making the KAFA Management System more user-friendly and visually engaging.

Previously, the registration process only collected basic personal information such as IC number, name, email, and password. There was no way to visually distinguish one user from another, which made it harder for administrators and teachers to manage large user lists, especially when users shared similar names or details.

With this change, the registration form now includes an image upload field labeled “Upload Profile Picture.” Users are able to select and upload an image file (JPEG, PNG) from their device. The system then stores the image securely in the designated server directory and links it to the user’s profile record.

To maintain consistency and data integrity, several validation measures were introduced:

- Only image files (e.g., .jpg, .jpeg, .png) are accepted
- File size is limited to 2MB
- Real-time preview of the selected image is shown before submission
- Upload is optional but encouraged for enhanced user experience

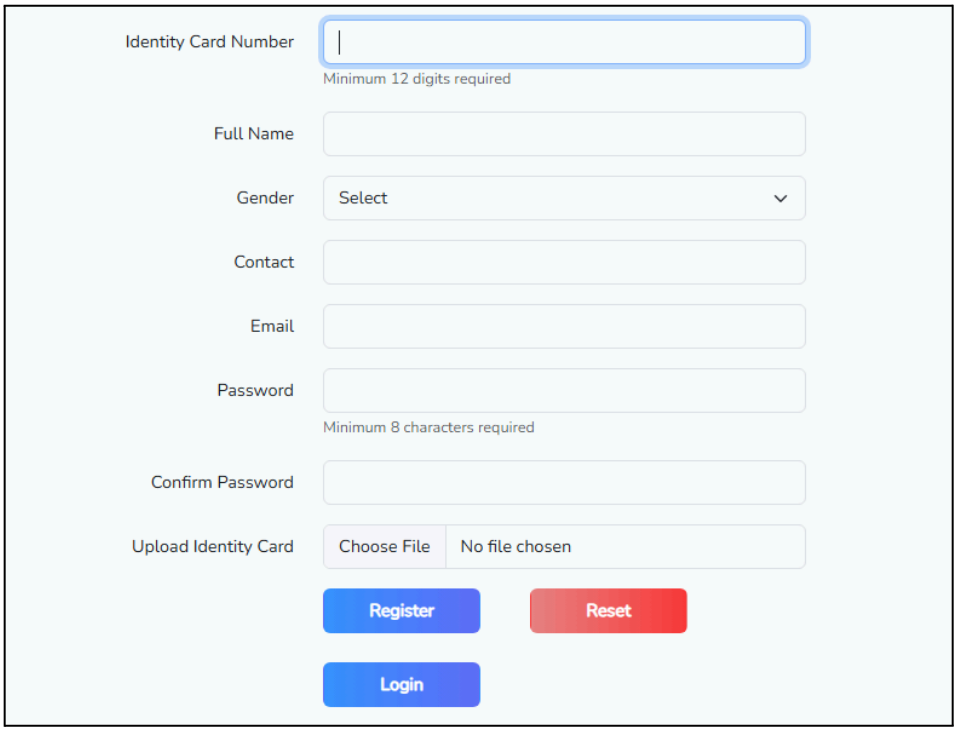
This feature required updates to:

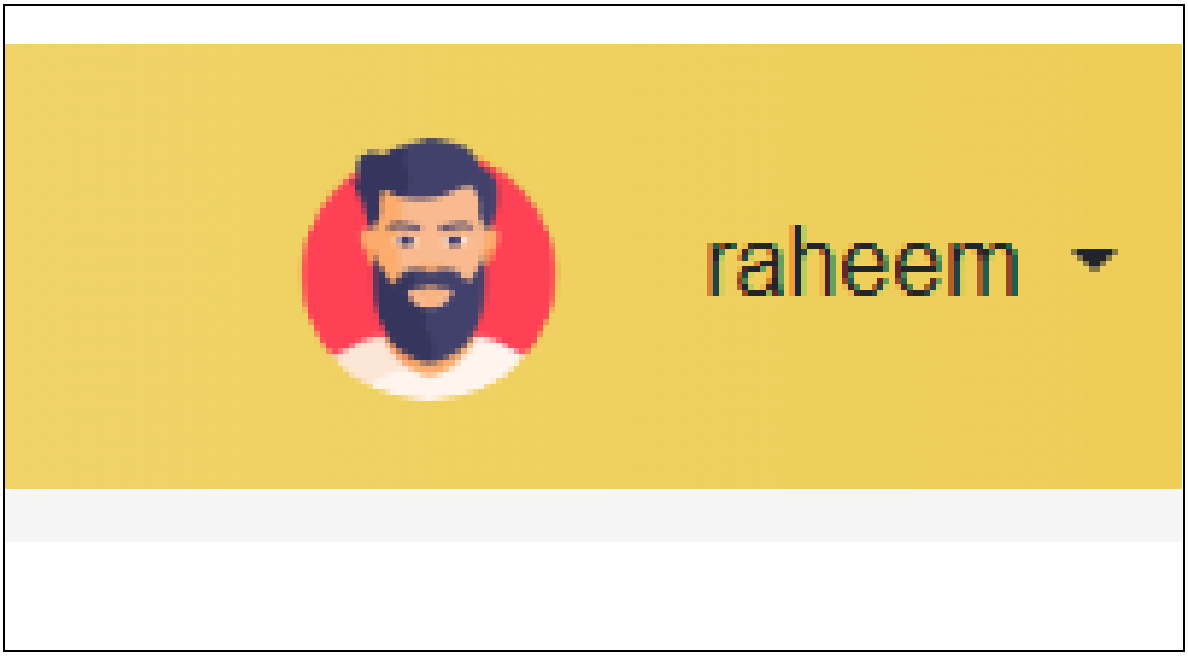
- The **frontend** registration form layout and input handling
- The **backend** logic to handle file storage and update user data models
- The **database schema** to include a profile_image field referencing the file path

By implementing this feature, the system enhances the visual recognition of users and supports administrative workflows such as verification, moderation, and profile review. This improvement also aligns with modern usability standards in web systems, where visual identifiers are key to improving user experience and trust.

Ultimately, the implementation of profile picture uploads reflects the system’s evolution toward a more **interactive**, **personalized**, and **professional** interface. It not only improves functionality but also adds value to user engagement with the system.

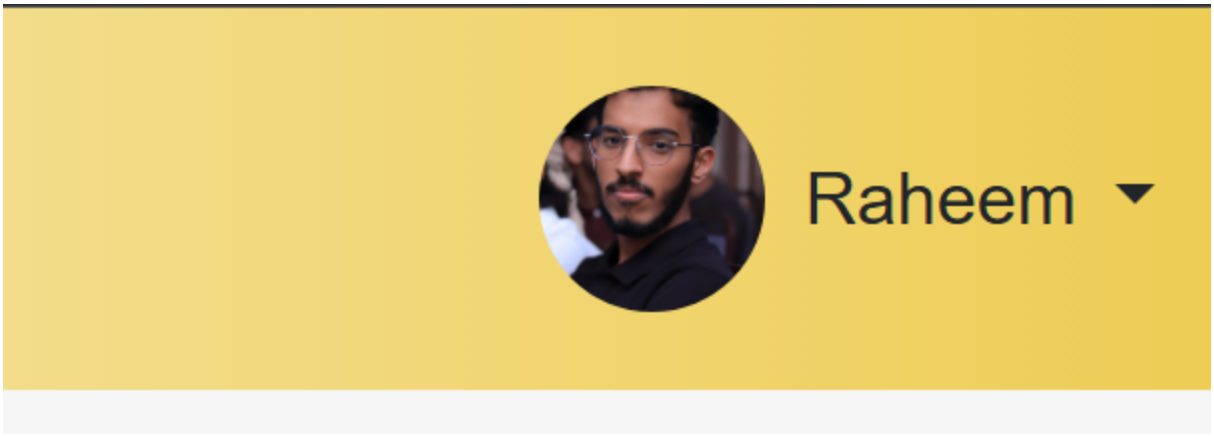
2.3.4 Implemented Change

Before Implementation





```
    return Validator::make($data, [
        'icnumber' => ['required', 'string', 'size:12'],
        'name' => ['required', 'string', 'max:255'],
        'gender' => ['required', 'string', 'max:255', 'in:Men,Women'],
        'contact' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
        'ic_docs' => ['required', 'pdf', 'max:10240'],

    ]);
```

After Implementation



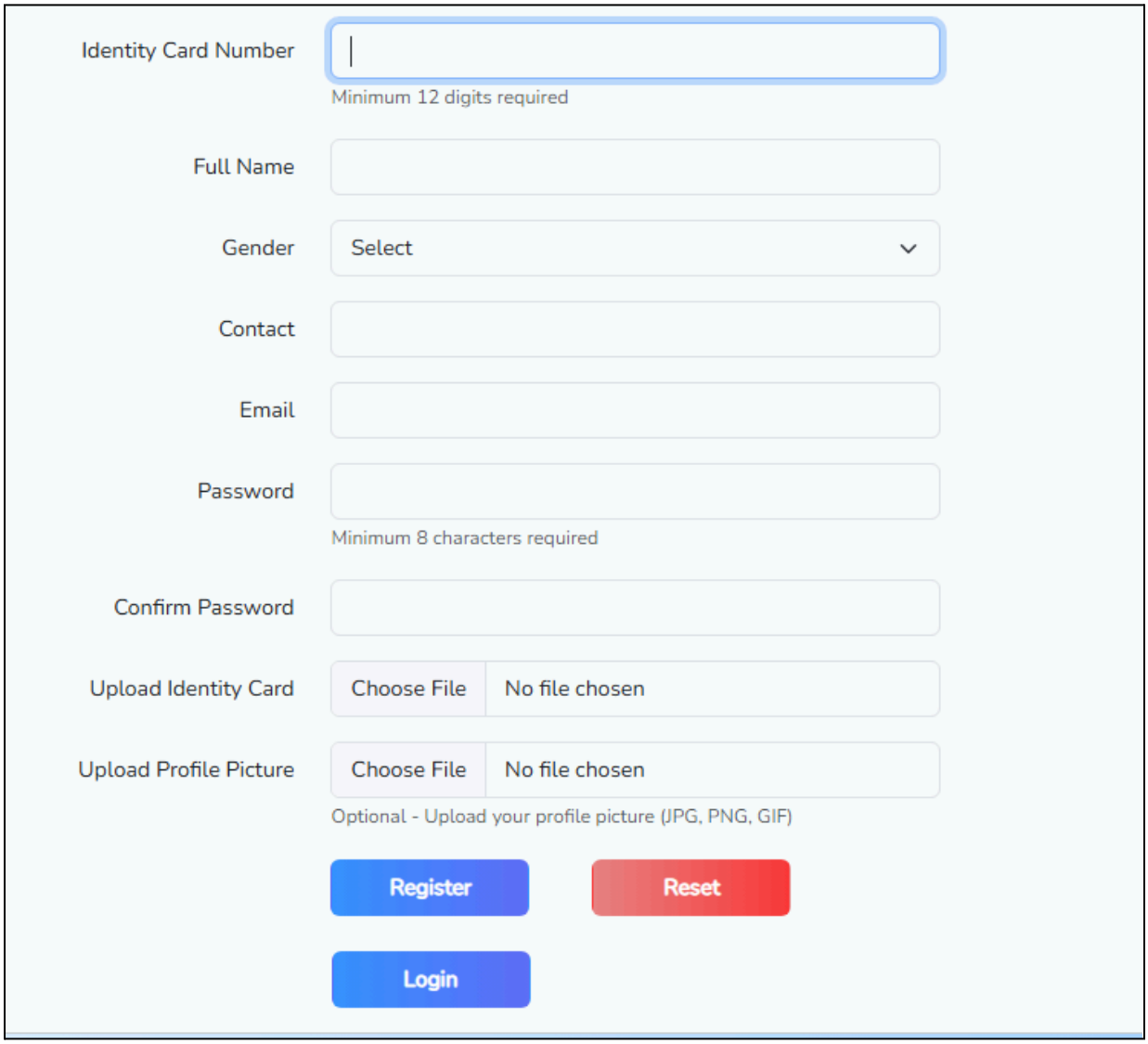

```php
protected function validator(array $data)
{
    return Validator::make($data, [
        'icnumber' => ['required', 'string', 'size:12'],
        'name' => ['required', 'string', 'max:255'],
        'gender' => ['required', 'string', 'max:255', 'in:Men,Women'],
        'contact' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
        'ic_docs' => ['required', 'mimes:pdf', 'max:10240'],
        'profile_picture' => ['nullable', 'image', 'mimes:jpeg,png,jpg,gif', 'max:5120'],
    ]);
}
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	role_id	bigint(20)		UNSIGNED	No	None			Change Drop More
3	user_name	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
4	user_ic	varchar(12)	utf8mb4_unicode_ci		No	None			Change Drop More
5	email	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
6	email_verified_at	timestamp			Yes	NULL			Change Drop More
7	password	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
8	user_gender	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
9	user_contact	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
10	user_verification	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
11	profile_picture	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
12	remember_token	varchar(100)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
13	created_at	timestamp			Yes	NULL			Change Drop More
14	updated_at	timestamp			Yes	NULL			Change Drop More

```php
        // Handle IC document upload
        if (request()->hasFile('ic_docs')) {
            $file = request()->file('ic_docs');
            $fileName = $file->getClientOriginalName();
            $path = $file->storeAs('Parent Verification', $fileName, 'public');
            $user->user_verification = $path;
        }

        // Handle profile picture upload
        if (request()->hasFile('profile_picture')) {
            $profilePicture = request()->file('profile_picture');
            $fileName = time() . '_' . $profilePicture->getClientOriginalName();
            $profilePath = $profilePicture->storeAs('profile_pictures', $fileName, 'public');
            $user->profile_picture = $profilePath;
        }

        $user->save();
        return $user;
    }
}
```

```php
    // Profile picture logic - use uploaded picture if available, otherwise fallback to gender-based default
    $profilePicture = null;
    if ($userRole->profile_picture && file_exists(public_path('storage/' . $userRole->profile_picture))) {
        $profilePicture = asset('storage/' . $userRole->profile_picture);
    } else {
        $imageprofile = $userRole->user_gender === 'Men' ? 'man.png' : ($userRole->user_gender === 'Women' ? 'woman.png' : 'mantantersakiti.jpeg');
        $profilePicture = asset('default_image/' . $imageprofile);
    }
@endphp
```

```
</div>

<div class="row mb-3">
    <label for="profile_picture" class="col-md-4 col-form-label text-md-end">{{ __('Upload Profile Picture') }}</label>
    <div class="col-md-6">
        <input id="profile_picture" type="file" class="form-control @error('profile_picture') is-invalid @enderror" name="profile_picture" accept="image/*">
        @error('profile_picture')
            <span class="invalid-feedback" role="alert">
                <strong>{{ $message }}</strong>
            </span>
        @enderror
        <small class="text-muted">Optional - Upload your profile picture (JPG, PNG, GIF)</small>
    </div>
</div>
```



2.3. Manage

2.3.1 Type of Maintenance Chosen

2.3.2 Change Request Form

2.3.3 Proposed Change

2.3.4 Implemented Change

2.4. Manage Result

2.4.1 Type of Maintenance Chosen

2.4.2 Change Request Form

2.4.3 Proposed Change

2.4.4 Implemented Change

2.5. Manage Bulletin (Nur Alya Syakirah Binti Nasarudin | CB22141)

2.5.1 Type of Maintenance Chosen 1

The type of software maintenance chosen in the Manage Bulletin module is enhancement maintenance. Enhancement maintenance is a form of software maintenance that focuses on adding new features or improving existing ones based on evolving user needs and feedback. The main objective is to enhance the system's usability, functionality and overall user satisfaction without necessarily fixing bugs or errors. It responds to user demands by introducing new capabilities that improve system interaction and information accessibility.

In this case, the enhancement was implemented by introducing a filter button and displaying the poster's name and posting date on each bulletin. The filtering feature allows users to sort bulletin notices by date such as "Today" or "Yesterday" making it easier to locate recent or relevant announcements. Displaying the poster's name and posting time adds clarity and accountability allowing users to know the source and timing of the information being shared. These enhancements improve navigation and transparency within the system.

Previously, the bulletin board lacked both the filtering capability and identification of the poster, which made it difficult for users to identify the source of a notice or determine when it was posted. Without these features, users had to manually scroll through all bulletin posts without a way to prioritize or filter them leading to wasted time and reduced efficiency especially when trying to locate recent announcements.

With the new features in place, the software becomes more user-focused and intuitive. It improves communication within the KAFA school system by ensuring that important information is organized, easy to find and clearly attributed. This aligns with the goals of enhancement maintenance, which is to meet user expectations by improving the system's usefulness, accessibility and user experience.

2.5.2 Change Request Form

Project Description				
Project Name	KAFA Management System			
Project ID Number	PR-KMS-400			
Change Description				
Change Request Submitter Name	Nur Alya Syakirah Binti Nasarudin			
Change Request Name	Manage Bulletin: Addition of filter and poster info on bulletin board			
Type of CR	☑ Enhancement ☐ Corrective			
Description of Change	The bulletin board was enhanced to include a filter button that allows users to sort bulletins by specific date ranges. Additionally, each bulletin post now displays the name of the notice and the date it was published. This involved updates to both the user interface and the backend to support filtering functionality and display enhancements.			
Reason of Change	The change improves user experience by allowing users to identify the poster and date of each bulletin and to filter notices by date. This makes it easier to find relevant announcements quickly and enhances overall bulletin navigation.			
Date of Request	19 May 2025			
Priority of Change	☐ Low	☐ Medium	☑ High	☐ Critical
Approval Description				
Decision ☑	☑ Approved	☐ Approved with Conditions		☐ Rejected
Decision Date	29 May 2025			
Approval Signature			Approval Date : 1 June 2025	

2.5.3 Proposed Change

The proposed change within the Manage Bulletin module involved adding a filter button and displaying the poster's name and posting date. Before this maintenance, users were able to view all bulletin notices but there was no way to know who posted them or when they were published. Additionally, the absence of a filtering function made it difficult to locate specific or recent notices efficiently. This lack of clarity and navigation made it time-consuming for users to browse through all bulletin content especially when searching for relevant announcements.

To address this issue, filtering functionality was implemented along with the display of the poster's name and post date. With these enhancements, users can now filter bulletin notices by specific date ranges and view important context such as who posted the notice and when. This allows users to quickly identify and access relevant information without having to manually go through every post.

These improvements significantly enhance the user's ability to navigate and manage bulletin content. By adding filter options and visual indicators of the poster and date, the system reduces time spent locating notices and increases overall transparency. This not only streamlines communication across the platform but also improves system usability for more effective school-wide information management.

2.5.4 Implemented Change

The implemented changes address issues in the Manage Bulletin module by adding a filter button and displaying the poster's name and posting date. These enhancements improve bulletin navigation and information clarity for all affected users including Teachers, KAFA Admins, MUIP Admins and Parents.

Before Implementation

```
{
    //Display all notices
    public function allnotices(Request $request)
    {
        $status = $request->notice_status;

        if ($status === 'Pending' || $status === 'Approved' || $status === 'Rejected') {
            $notices = Notice::where('notice_status', $status)->get();

            return view('ManageBulletin.all_notices', compact('notices', 'status'));
        } else {
            $notices = Notice::all(); // fetch all notices

            $status = "null";

            return view('ManageBulletin.all_notices', compact('notices', 'status'));
        }
    }
```

Figure 2.1 Bulletin Board code before filter button implementation

Figure 2.1 displays the Bulletin Board code before implementation. The system did not include a filter button to sort or view notices by date.

```
<div class="container">
    @foreach($notices as $notice)
    @if($notice->notice_status == 'Approved')
    <div class="row">
        <div class="col">
            <div style="border: 1px solid black; background-color:
#f2f2f2; padding: 10px; margin-bottom: 10px;">
                <div style="padding-left: 10px; padding-right: 10px; margin-top: 10px;">
                    <b style="text-transform: uppercase;">{{ $notice->notice_title }}</b>
                    <br>
                    <div style="margin-top: 10px;">
                        {{ $notice->notice_text }}
                    </div>
                    <div style="text-align: right; margin-top: 10px; margin-bottom: 10px;">
                        <a href="{{ route('selectednotices', ['id' => $notice->id]) }}" class="btn btn-primary" style="margin-right: 10px;">
                            {{ __('View') }}
                        </a>
                    </div>
                </div>
            </div>
        </div>
    </div>
    @endif
    @endforeach
```

Figure 2.2 Bulletin Board code before notice's name and date implementation

Figure 2.2 displays the Bulletin Board code before implementation, where the system only showed bulletin notices posted by Teachers, KAFA Admins or MUIP Admins without displaying the notice's name or the date.

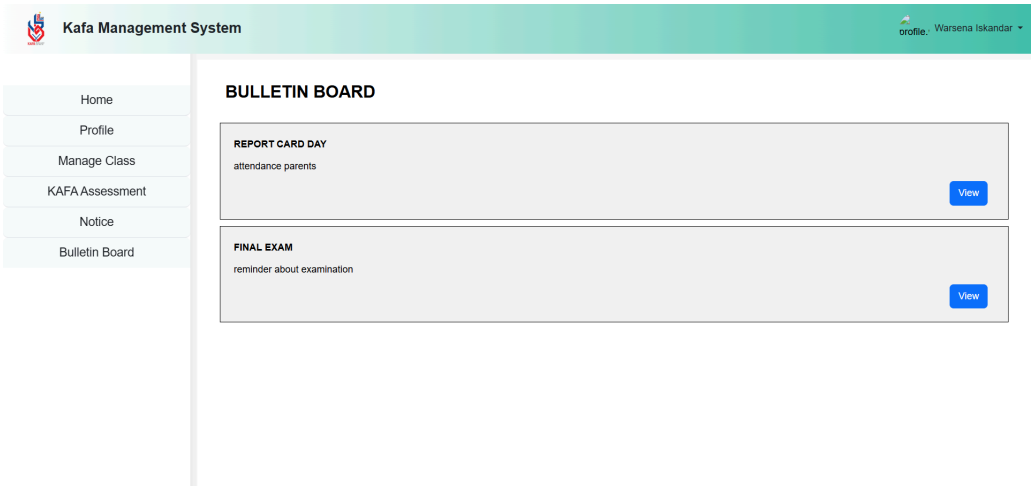


Figure 2.3 Bulletin Board interface before the implementation of the filter button and notice's name and date (Teacher dashboard)

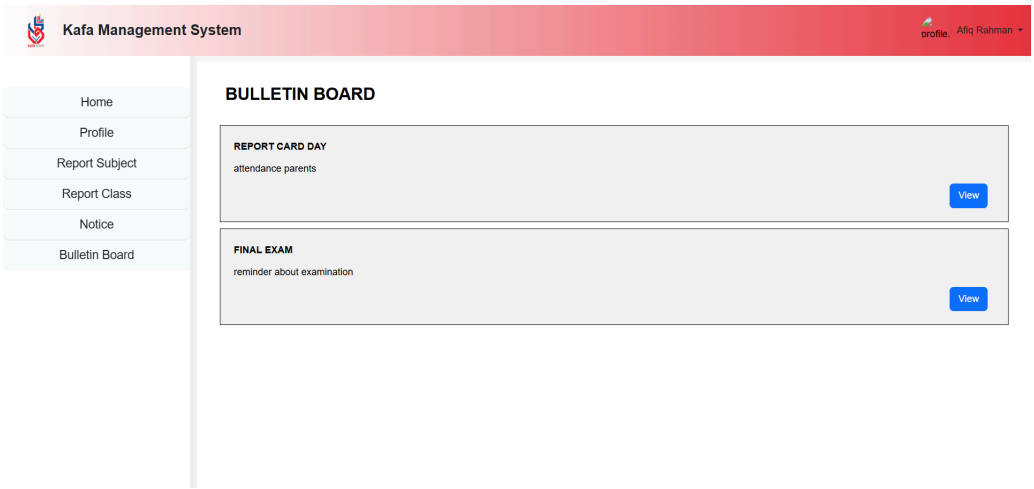


Figure 2.4 Bulletin Board interface before the implementation of the filter button and notice's name and date (MUIP Admin dashboard)

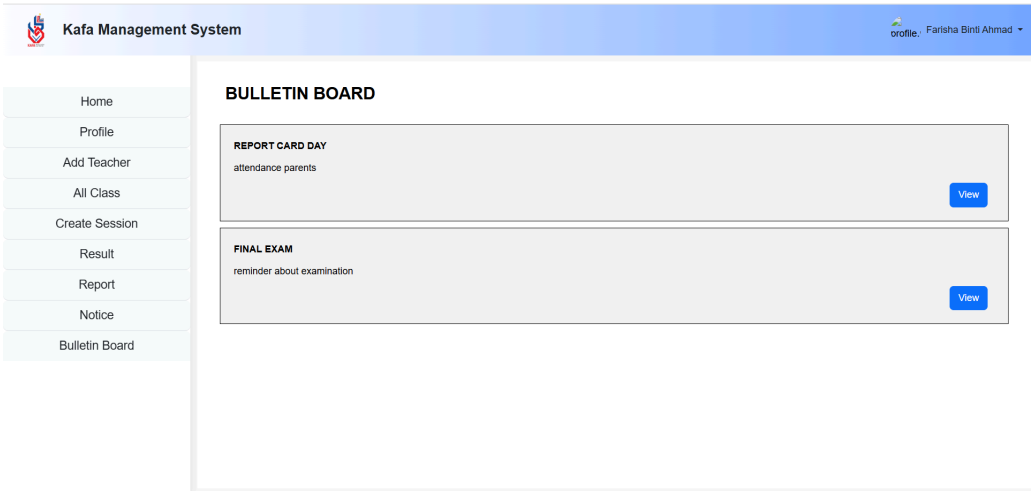


Figure 2.5 Bulletin Board interface before the implementation of the filter button and notice's name and date (KAFA Admin dashboard)

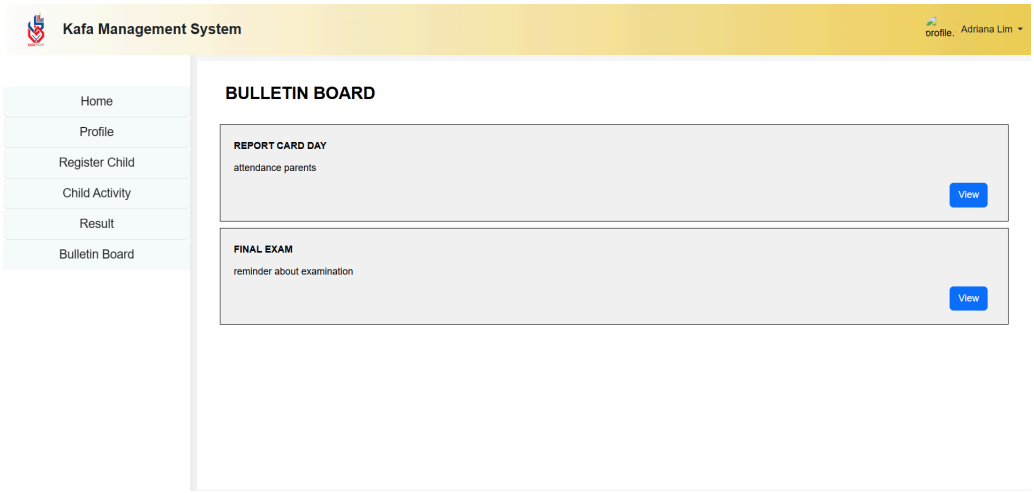


Figure 2.6 Bulletin Board interface before the implementation of the filter button and notice's name and date (Parent dashboard)

Figures 2.3 to 2.6 show the interface of the Bulletin Board before implementation where there was no filter button to sort notices by date and the system did not display the notice's name or the date it was published.

After Implementation

```
14      //Display all notices with filtering options
15      public function allnotices(Request $request)
16      {
17          $status = $request->notice_status;
18          $dateFilter = $request->date_filter;
19          $customStartDate = $request->start_date;
20          $customEndDate = $request->end_date;
21
22          // Start building the query
23          $query = Notice::query();
24
25          // Apply status filter
26          if ($status === 'Pending' || $status === 'Approved' || $status === 'Rejected') {
27              $query->where('notice_status', $status);
28          }
29
30          // Apply date filters
31          if ($dateFilter) {
32              $now = Carbon::now();
33
34              switch ($dateFilter) {
35                  case 'today':
36                      $query->whereDate('notice_submission_date', $now->toDateString());
37                      break;
38
39                  case 'yesterday':
40                      $query->whereDate('notice_submission_date', $now->subDay()->toDateString());
41                      break;
```

Figure 2.7 Bulletin Board code after filter button implementation

```
31          if ($dateFilter) {
34              switch ($dateFilter) {
39                  case 'yesterday':
43                  case 'this_week':
44                      $query->whereBetween('notice_submission_date', [
45                          $now->startOfWeek()->toDateString(),
46                          $now->endOfWeek()->toDateString()
47                      ]);
48                      break;
49
50                  case 'last_week':
51                      $startOfLastWeek = $now->subWeek()->startOfWeek();
52                      $endOfLastWeek = $now->endOfWeek();
53                      $query->whereBetween('notice_submission_date', [
54                          $startOfLastWeek->toDateString(),
55                          $endOfLastWeek->toDateString()
56                      ]);
57                      break;
58
59                  case 'this_month':
60                      $query->whereMonth('notice_submission_date', $now->month)
61                            ->whereYear('notice_submission_date', $now->year);
62                      break;
63
64                  case 'last_month':
65                      $lastMonth = $now->subMonth();
66                      $query->whereMonth('notice_submission_date', $lastMonth->month)
67                            ->whereYear('notice_submission_date', $lastMonth->year);
68                      break;
```

Figure 2.8 Bulletin Board code after filter button implementation

Figure 2.7 and 2.8 displays the Bulletin Board code after implementation. The changes, specifically the addition of the filter button, were made between line 17 and 105 in Visual Studio Code. This implementation allows users to filter bulletin notices by time ranges such as Yesterday, This Week, Last Week, Last Month, This Year, and Last Year. Based on the selected date range, the system will display the relevant notices on the bulletin board accordingly.

```
82  <!-- Notices Display Section -->
83  <div class="container">
84      @if($notices->count() > 0)
85          @foreach($notices as $notice)
86          @if($notice->notice_status == 'Approved')
87          <div class="row">
88              <div class="col">
89                  <div style="border: 1px solid black; background-color: #f2f2f2; padding: 10px; margin-bottom: 10px;">
90                      <div style="padding-left: 10px; padding-right: 10px; margin-top: 10px;">
91                          <b style="text-transform: uppercase;">{{ $notice->notice_title }}</b>
92                          <br>
93                          <small style="color: #6c757d; font-style: italic;">
94                              Posted on: {{ \Carbon\Carbon::parse($notice->notice_submission_date)->format('M d, Y \a\t g:i A') }}
95                              by {{ $notice->user->user_name ?? 'Unknown' }}
96                          </small>
97                          <div style="margin-top: 10px;">
98                              {{ $notice->notice_text }}
99                          </div>
100                         <div style="text-align: right; margin-top: 10px; margin-bottom: 10px;">
101                             <a href="{{ route('selectednotices', ['id' => $notice->id]) }}" class="btn btn-primary" style="margin-right: 10px;">
102                                 {{ __('View') }}
103                             </a>
104                             @if(Auth::check() && Auth::user()->role_id == 2)
105                                 <button type="button" class="btn btn-success" onclick="downloadNotice({{ $notice->id }}, '{{ addslashes($notice->n
106                                     <i class="fas fa-download"></i> {{ __('Download') }}
107                                 </button>
108                             @endif
109                         </div>
110                     </div>
111                 </div>
```

Figure 2.9 Bulletin Board code after notice's name and date implementation

Figure 2.9 displays the Bulletin Board code after implementation. The changes which added the poster's name and the notice date, were made between lines 82 and 111 in Visual Studio Code. With this update, users can now view who posted each notice and the date it was published.

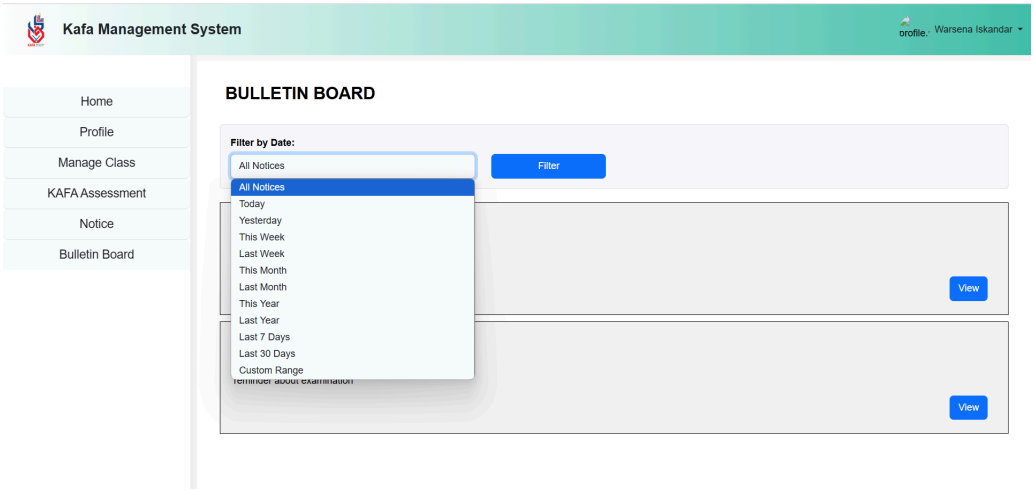


Figure 2.10 Bulletin Board interface after the implementation of the filter button (Teacher dashboard)

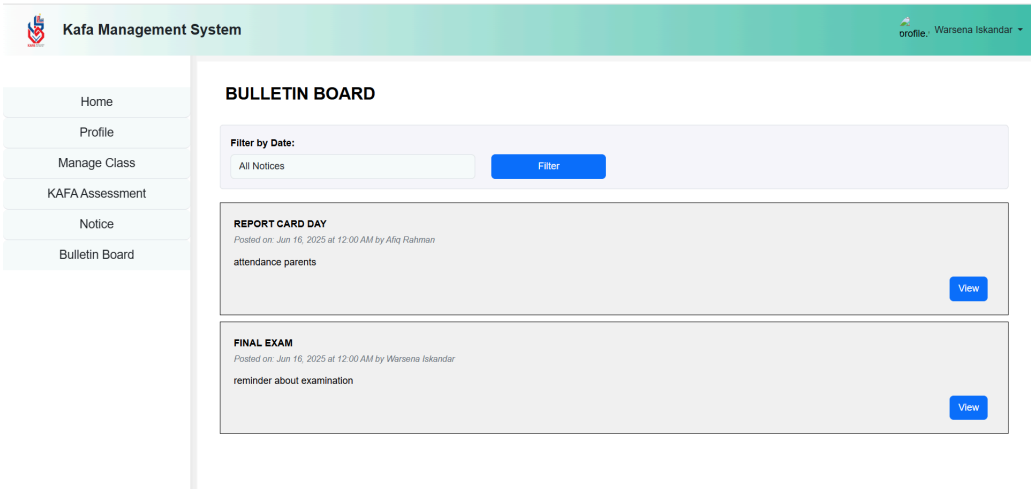


Figure 2.11 Bulletin Board interface after the implementation of the notice's name and date (Teacher dashboard)

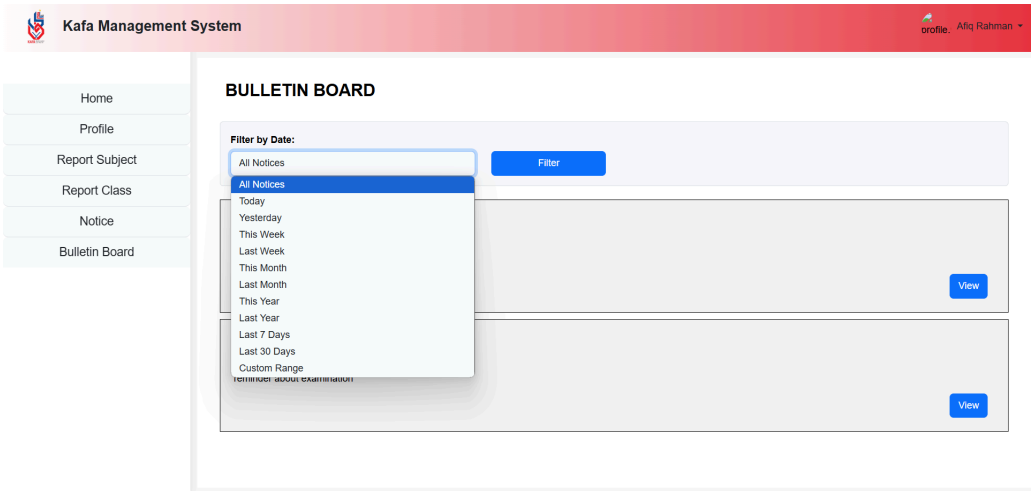


Figure 2.12 Bulletin Board interface after the implementation of the filter button (MUIP Admin dashboard)

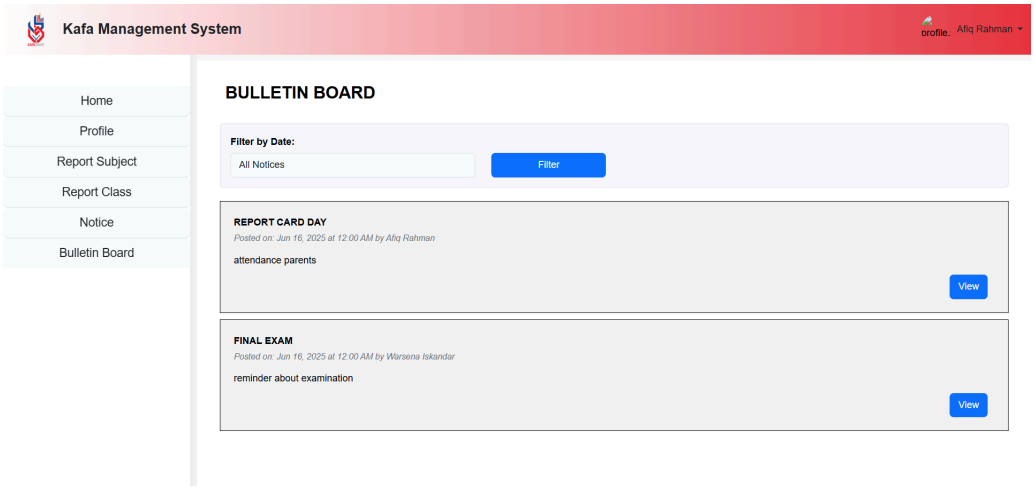


Figure 2.13 Bulletin Board interface after the implementation of the notice's name and date (MUIP Admin dashboard)

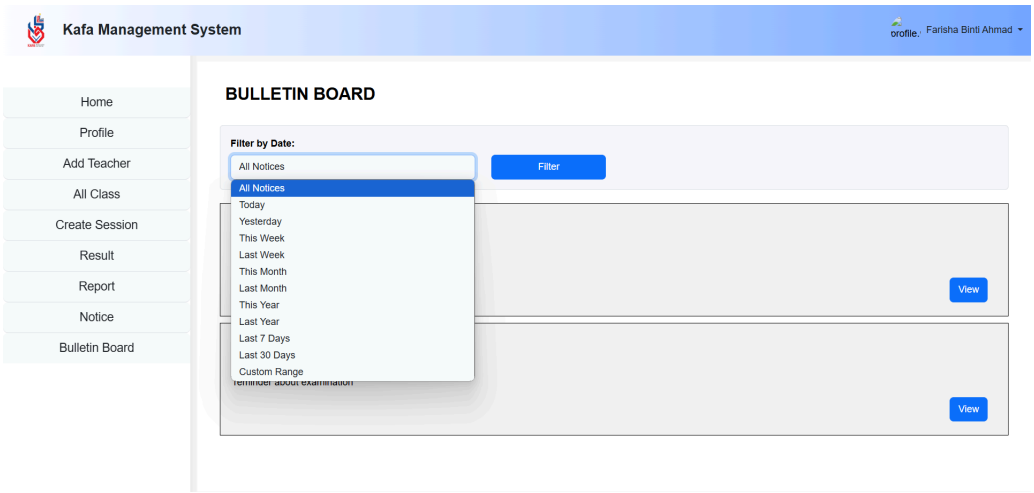


Figure 2.14 Bulletin Board interface after the implementation of the filter button (KAFA Admin dashboard)

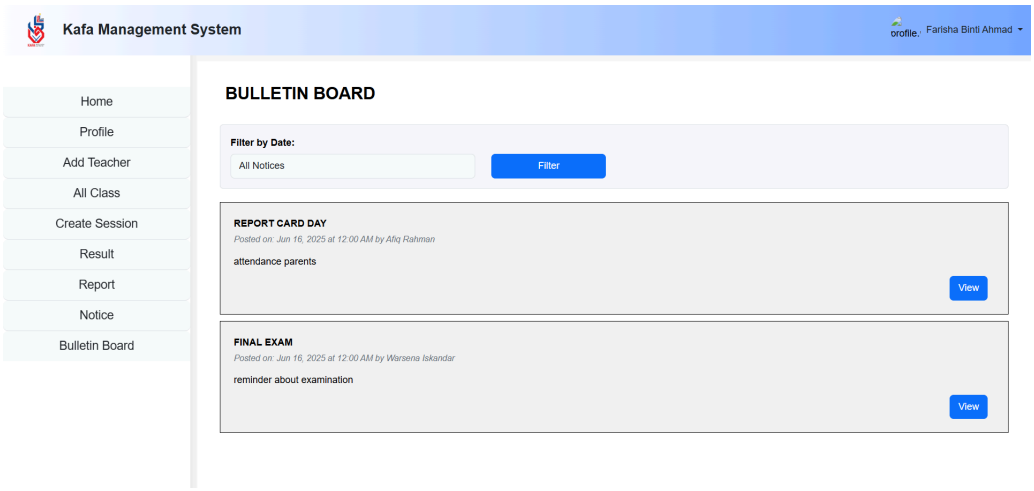


Figure 2.15 Bulletin Board interface after the implementation of the notice's name and date (KAFA Admin dashboard)

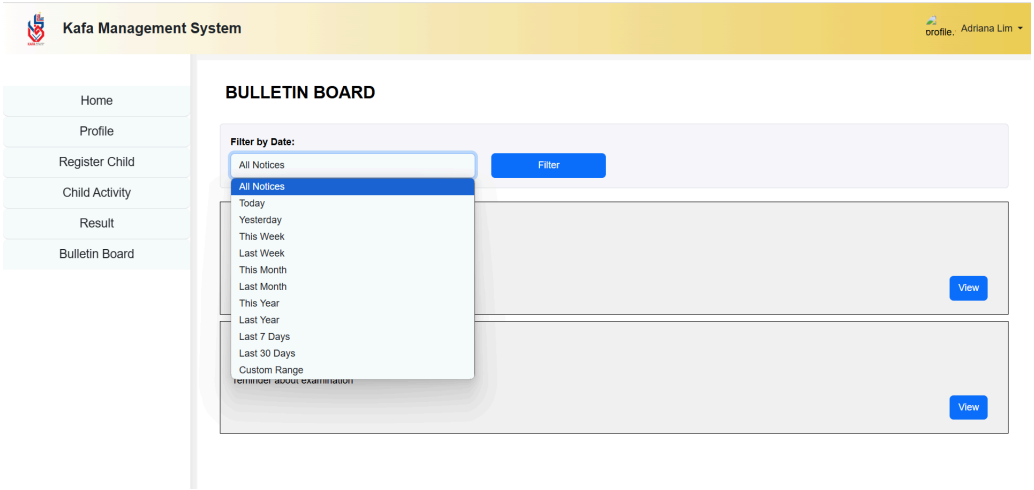


Figure 2.16 Bulletin Board interface after the implementation of the filter button (Parent dashboard)

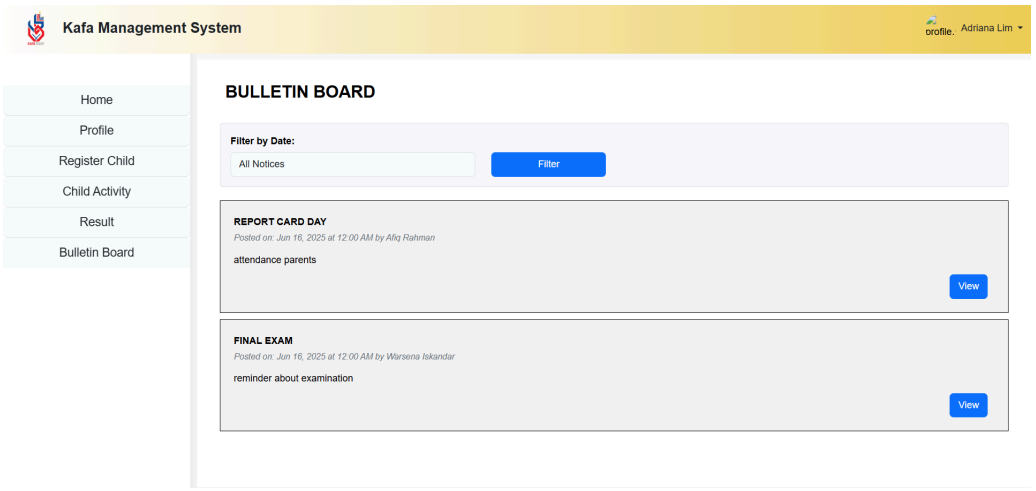


Figure 2.17 Bulletin Board interface after the implementation of the notice's name and date (Parent dashboard)

Figures 2.10 to 2.17 show the notice display on the Bulletin Board page after implementation, which includes the filter button, the notice's name and the date it was published. These improvements are visible to all user roles including Teachers, MUIP Admins, KAFA Admins and Parents enhancing clarity and navigation for every user group.

2.6. Manage Bulletin (Nur Alya Syakirah Binti Nasarudin | CB22141)

2.6.1 Type of Maintenance Chosen 2

The type of software maintenance chosen in the Manage Bulletin module is enhancement maintenance. Enhancement maintenance focuses on extending the system's functionality or introducing new features in response to user needs. Unlike corrective maintenance, which addresses defects or errors, enhancement maintenance improves the overall system experience by making it more practical, efficient and user-friendly.

In this case, the enhancement was implemented by adding a delete button that allows Teachers and MUIP Admins to remove their own notices from the Bulletin Board. Teachers can delete both approved and pending notices giving them more flexibility and control over the content they publish. This feature was introduced to improve content management and provide users with better oversight of their announcements.

Previously, the system did not provide any option for deleting notices once submitted which limited user control and made it difficult to manage incorrect or outdated announcements. If a user wanted to retract or correct a post, they had to rely on KAFA Admin intervention which slowed down workflows and reduced efficiency.

By enabling users to delete their own notices, the system becomes more responsive to real-time content management needs. It increases flexibility and supports better information governance. This change aligns with the purpose of enhancement maintenance as it upgrades existing functionality to better serve users and improve the overall system experience.

2.6.2 Change Request Form

Project Description				
Project Name	KAFA Management System			
Project ID Number	PR-KMS-401			
Change Description				
Change Request Submitter Name	Nur Alya Syakirah Binti Nasarudin			
Change Request Name	Manage Bulletin: Addition of delete button for bulletin notices			
Type of CR	☑ Enhancement ☐ Corrective			
Description of Change	The bulletin board was enhanced by adding a delete button allowing Teachers and MUIP Admins to remove their own notices. Teachers are permitted to delete both approved and pending posts. This change required updates to the interface and system logic to support controlled content removal.			
Reason of Change	The change improves content management by giving users control over their own notices. It helps manage outdated or incorrect posts more efficiently and reduces reliance on KAFA Admin intervention, streamlining workflows and improving overall system usability.			
Date of Request	19 May 2025			
Priority of Change	☐ Low	☐ Medium	☑ High	☐ Critical
Approval Description				
Decision ☑	☑ Approved	☐ Approved with Conditions		☐ Rejected
Decision Date	29 May 2025			
Approval Signature			Approval Date : 1 June 2025	

2.6.3 Proposed Change

The proposed change within the Manage Bulletin module involved adding a delete button that allows Teachers and MUIP Admins to remove their own notices. Before this maintenance, users had no control over their submitted notices once published. This limitation made it difficult to correct errors or remove outdated announcements and users had to rely on KAFA Admin support for content removal. As a result, it slowed down workflows and reduced efficiency in managing bulletin content.

To resolve this issue, a delete function was implemented enabling Teachers to delete both approved and pending notices while MUIP Admins can remove their own notices as needed. This change provides greater flexibility and control allowing users to manage their bulletin notices more effectively without requiring intervention from the KAFA Admin.

These improvements significantly enhance content management and reduce dependency on administrative actions. By allowing users to directly manage their notices, the system promotes better accuracy, reduces clutter from outdated posts and supports faster updates. This enhancement ultimately improves the overall usability of the bulletin board and strengthens communication across the platform.

2.6.4 Implemented Change

The implemented changes address issues in the Manage Bulletin module by adding a delete button allowing Teachers and MUIP Admins to remove their own notices. This improves content control and streamlines bulletin management.

Before Implementation

```
@if($userRole->role_id == 4 || $userRole->role_id == 2)
<div class="col text-center d-flex align-items-center justify-content-center"
    style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
    {{ $notice->notice_status }}
</div>
@endif
@if($userRole->role_id == 2)
<div class="col text-center d-flex align-items-center justify-content-center"
    style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
    @if ($notice->notice_status == "Approved")
    <button class="btn btn-warning btn-sm" disabled>
        <i class="bi bi-eye"></i>
    </button>
    @elseif ($notice->notice_status == "Rejected")
    <button class="btn btn-warning btn-sm" disabled>
        <i class="bi bi-eye"></i>
    </button>
    @else
    <a href="{{ route('formapproval', ['id' => $notice->id]) }}" class="btn btn-warning btn-sm">
        <i class="bi bi-eye"></i>
    </a>
    @endif
      
    <form action="{{ route('deletenotice', $notice->id) }}" method="POST"
        onsubmit="return confirm('Are you sure you want to delete this notice?');">
        @csrf
        @method('DELETE')
        <button type="submit" class="btn btn-danger btn-sm">
```

Figure 2.18 Notice page code before delete button implementation

Figure 2.18 displays the Notice page code before delete button implementation. Teachers could not delete pending or approved notices and MUIP Admins had no option to delete their notices.

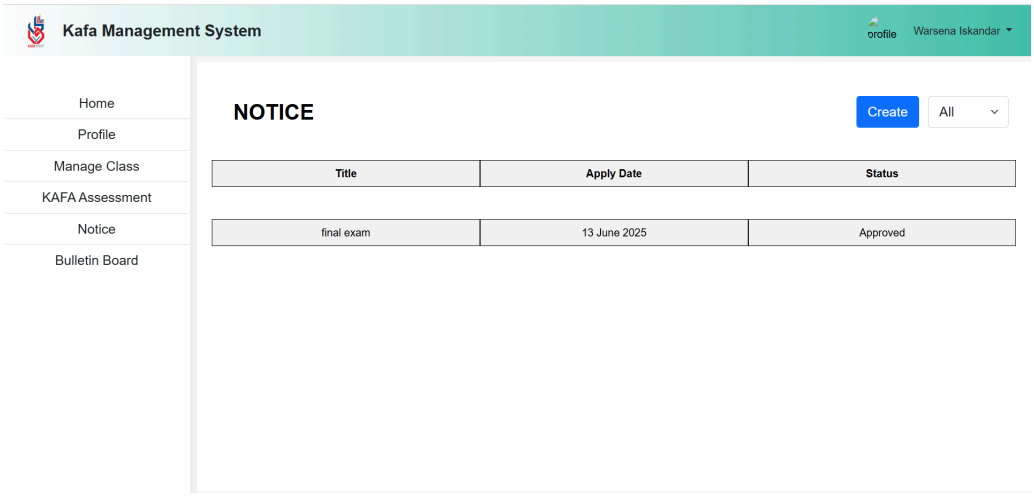


Figure 2.19 Notice page interface before the implementation of the delete button (Teacher dashboard)

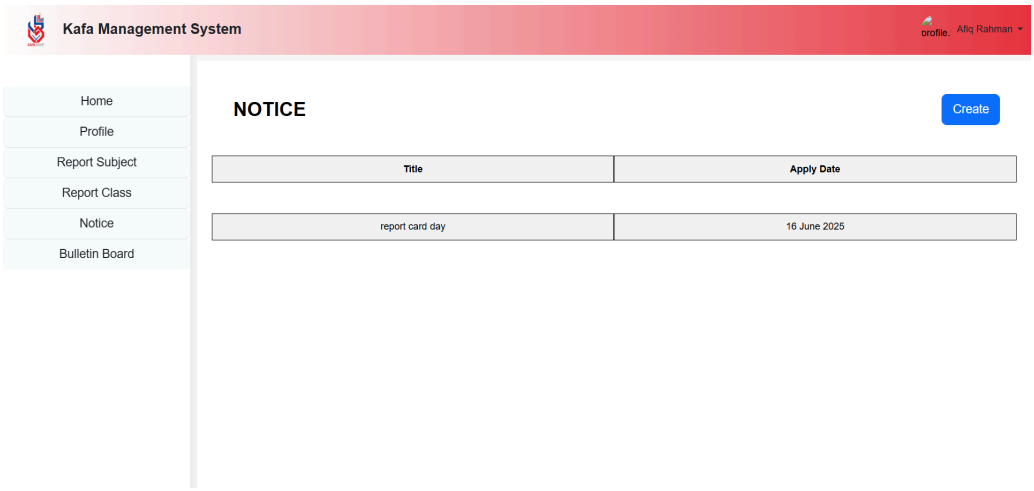


Figure 2.20 Notice page interface before the implementation of the delete button (MUIP Admin)

Figures 2.19 and 2.20 show the Notice page before implementation, where Teachers and MUIP Admins did not have the option to delete the notices they created.

After Implementation

```
89              @endif
90              <div class="col text-center d-flex align-items-center justify-content-center"
91                  style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
92                  {{ \Carbon\Carbon::parse($notice->notice_submission_date)->format('j F Y') }}
93              </div>
94              @if($userRole->role_id == 4 || $userRole->role_id == 2)
95              <div class="col text-center d-flex align-items-center justify-content-center"
96                  style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
97                  {{ $notice->notice_status }}
98              </div>
99              @endif
100             <div class="col text-center d-flex align-items-center justify-content-center"
101                 style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
102                 @if($userRole->role_id == 2)
103                 @if ($notice->notice_status == "Approved")
104                 <button class="btn btn-warning btn-sm" disabled>
105                     <i class="bi bi-eye"></i>
106                 </button>
107                 @elseif ($notice->notice_status == "Rejected")
108                 <button class="btn btn-warning btn-sm" disabled>
109                     <i class="bi bi-eye"></i>
110                 </button>
111                 @else
112                 <a href="{{ route('formapproval', ['id' => $notice->id]) }}" class="btn btn-warning btn-sm">
113                     <i class="bi bi-eye"></i>
114                 </a>
115                 @endif
116                   
```

Figure 2.21 Notice page code after delete button implementation

```
116                   
117                 @endif
118                 <form action="{{ route('deletenotice', $notice->id) }}" method="POST"
119                     onsubmit="return confirm('Are you sure you want to delete this notice?');">
120                     @csrf
121                     @method('DELETE')
122                     <button type="submit" class="btn btn-danger btn-sm">
123                         <i class="bi bi-trash"></i>
124                     </button>
```

Figure 2.22 Notice page code after delete button implementation

Figures 2.21 and 2.22 display the Notice page code after implementation. The changes, specifically the addition of the delete button, were made between lines 90 and 124 in Visual Studio Code. This implementation allows Teachers and MUIP Admins to delete their own notices directly from the bulletin board giving them better control over the content they publish.

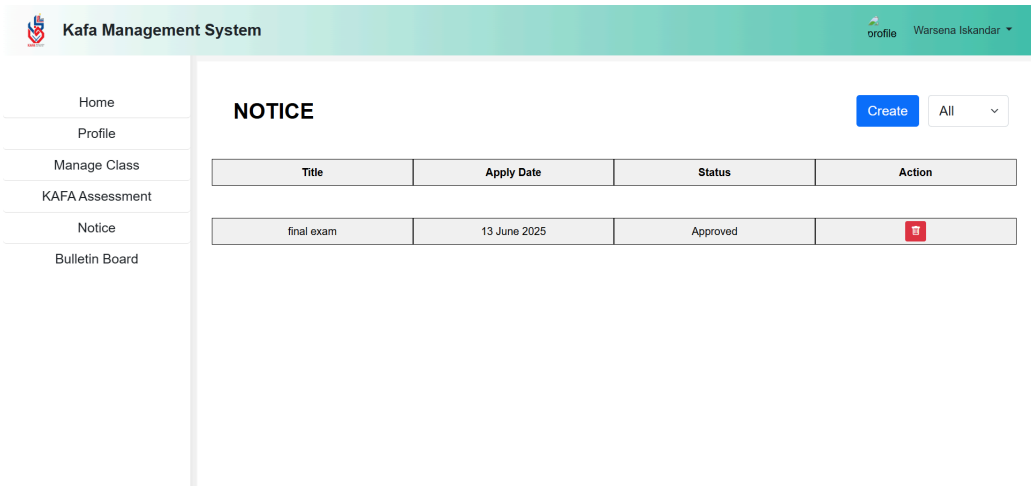


Figure 2.23 Notice page interface after the implementation of the delete button (Teacher dashboard)

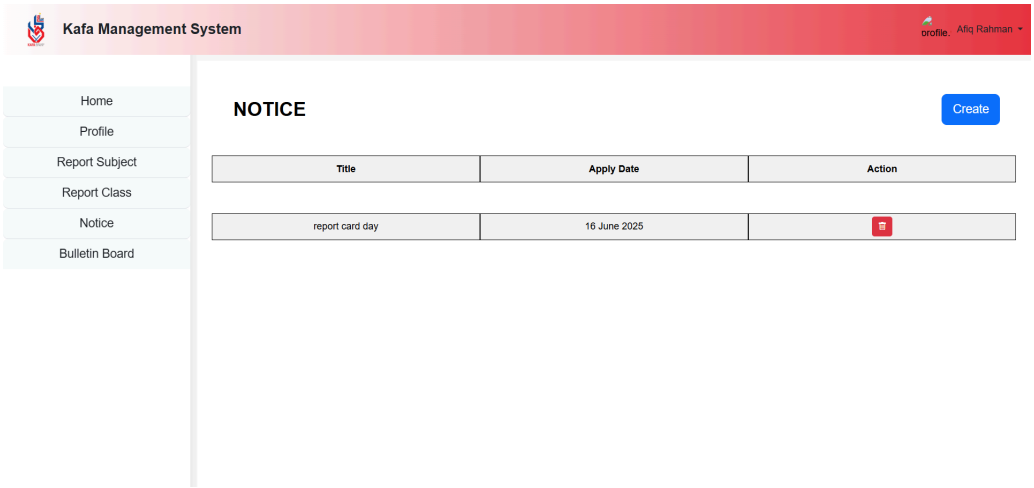


Figure 2.24 Notice page interface after the

implementation of the delete button (MUIP Admin dashboard)

Figures 2.23 and 2.24 show the notice display on the Notice page after implementation, which includes the action form with a delete button allowing Teachers and MUIP Admins to remove their own notices, enhancing control and content management for relevant user roles.

2.7. Manage Bulletin (Nur Alya Syakirah Binti Nasarudin | CB22141)

2.7.1 Type of Maintenance Chosen 3

The type of software maintenance chosen in the Manage Bulletin module is enhancement maintenance. Enhancement maintenance focuses on extending the system's functionality or introducing new features in response to user needs. Unlike corrective maintenance, which fixes errors or bugs, enhancement maintenance improves the overall system experience by making it more efficient, practical and user-friendly.

In this case, the enhancement was implemented by adding a download button on the KAFA Admin Bulletin dashboard, allowing bulletin notices to be downloaded in PDF format. This feature provides KAFA Admins with a simple way to save or print notices for reference, documentation or administrative purposes.

Previously, bulletin notices could only be viewed within the system interface with no option to download or store them externally. This limitation made it harder to keep official records or access notices outside of the system especially when preparing reports or reviewing past communications.

By enabling the download feature, the system becomes more flexible and supportive of KAFA Admin workflows. It improves information accessibility, supports better record-keeping and aligns with the goal of enhancement maintenance by upgrading the system to meet evolving administrative needs.

2.7.2 Change Request Form

Project Description				
Project Name	KAFA Management System			
Project ID Number	PR-KMS-402			
Change Description				
Change Request Submitter Name	NUR ALYA SYAKIRAH BINTI NASARUDIN			
Change Request Name	Manage Bulletin: Addition of download button for bulletin notices			
Type of CR	☑ Enhancement ☐ Corrective			
Description of Change	The bulletin board was enhanced by adding a download button on the KAFA Admin Bulletin dashboard allowing bulletin notices to be downloaded as PDF files. This change required updates to both the interface and backend to support file generation and download functionality.			
Reason of Change	The change improves accessibility and record-keeping by allowing KAFA Admins to download notices for external reference. It supports better documentation, easier reporting and provides flexibility in how bulletin information is stored and shared.			
Date of Request	19 May 2025			
Priority of Change	☐ Low	☐ Medium	☑ High	☐ Critical
Approval Description				
Decision ☑	☑ Approved	☐ Approved with Conditions		☐ Rejected
Decision Date	29 May 2025			
Approval Signature			Approval Date : 1 June 2025	

2.7.3 Proposed Change

The proposed change within the Manage Bulletin module involved adding a download button that allows KAFA Admins to download bulletin notices in PDF format. Before this maintenance, notices could only be viewed within the system, with no built-in option to export or save them externally. This limitation made it difficult for KAFA Admins to keep formal records, share notices outside the system or reference them offline when needed.

To address this issue, a download function was implemented specifically on the KAFA Admin Bulletin dashboard. This feature enables KAFA Admins to generate and save bulletin notices as PDF files with a single click improving accessibility and flexibility in managing official communications.

These improvements enhance the system's practicality by supporting better record-keeping, reporting and offline access to important notices. By introducing this capability, the enhancement contributes to more efficient administration and aligns with the goal of improving overall system usability and functionality.

2.7.4 Implemented Change

The implemented changes address issues in the Manage Bulletin module by adding a download button allowing KAFA Admins to export bulletin notices as PDF. This enhances record-keeping and improves access to bulletin information.

Before Implementation

```
<!-- JavaScript for Custom Date Range Toggle -->
<script>
function toggleCustomDateRange() {
    const dateFilter = document.getElementById('dateFilter').value;
    const customDateRange = document.getElementById('customDateRange');

    if (dateFilter === 'custom') {
        customDateRange.style.display = 'block';
    } else {
        customDateRange.style.display = 'none';
        // Clear custom date inputs when not using custom filter
        document.getElementById('start_date').value = '';
        document.getElementById('end_date').value = '';
    }
}

// Auto-submit form when date filter changes (except for custom)
document.getElementById('dateFilter').addEventListener('change', function() {
    if (this.value !== 'custom' && this.value !== '') {
        document.getElementById('filterForm').submit();
    }
});
</script>

@endsection
```

Figure 2.25 KAFA Admin Bulletin Dashboard code before download button implementation

Figure 2.25 displays the KAFA Admin Bulletin dashboard code before download button implementation. At this stage, KAFA Admins had no option to download bulletin notices for external reference or record-keeping.

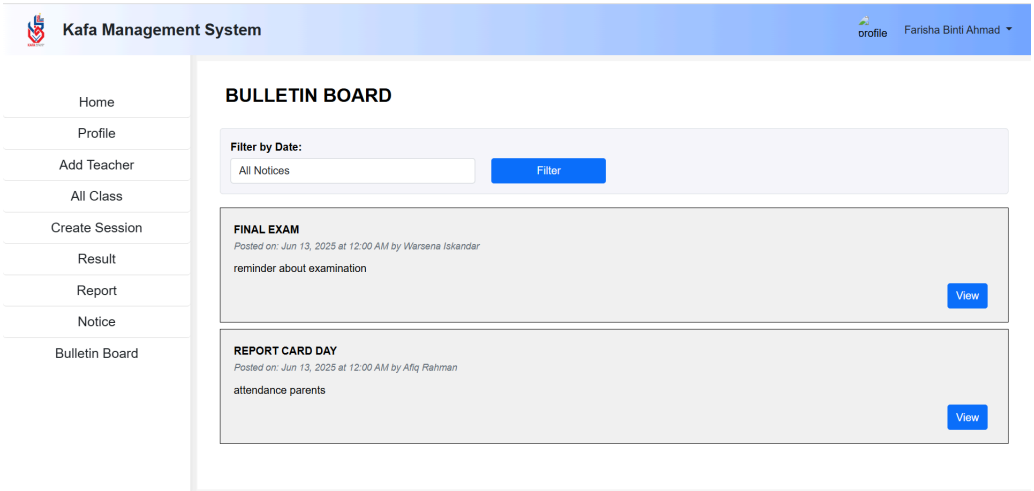


Figure 2.26 KAFA Admin Bulletin dashboard interface before the implementation of the download button

Figure 2.26 shows the bulletin interface on the KAFA Admin dashboard before the implementation, where there was no option for KAFA Admins to download bulletin notices for reference or record-keeping.

After Implementation

```
<!-- Load jsPDF library from CDN -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>

<!-- JavaScript for Custom Date Range Toggle and Download Function -->
<script>
function toggleCustomDateRange() {
    const dateFilter = document.getElementById('dateFilter').value;
    const customDateRange = document.getElementById('customDateRange');

    if (dateFilter === 'custom') {
        customDateRange.style.display = 'block';
    } else {
        customDateRange.style.display = 'none';
        // Clear custom date inputs when not using custom filter
        document.getElementById('start_date').value = '';
        document.getElementById('end_date').value = '';
    }
}
```

Figure 2.27 KAFA Admin Bulletin Dashboard code after download button implementation

```
// Download notice as PDF function
function downloadNotice(noticeId, title, content, author, date) {
    // Initialize jsPDF
    const { jsPDF } = window.jspdf;
    const doc = new jsPDF();

    // Set up fonts and styling
    doc.setFont("helvetica", "bold");
    doc.setFontSize(18);

    // Add title
    doc.text("BULLETIN BOARD NOTICE", 20, 20);

    // Add a line separator
    doc.setLineWidth(0.5);
    doc.line(20, 25, 190, 25);

    // Notice title
    doc.setFontSize(16);
    doc.setFont("helvetica", "bold");
    const titleLines = doc.splitTextToSize(title.toUpperCase(), 170);
    let currentY = 40;
    doc.text(titleLines, 20, currentY);
    currentY += titleLines.length * 7;

    // Notice details
    doc.setFontSize(10);
    doc.setFont("helvetica", "normal");
    currentY += 10;
    doc.text(`Posted on: ${date}`, 20, currentY);
    currentY += 6;
```

Figure 2.28 KAFA Admin Bulletin Dashboard code after download button implementation

```
212                 currentY = 20;
213             }
214             doc.text(contentLines[i], 20, currentY);
215             currentY += 6;
216         }
217
218         // Add footer
219         currentY += 20;
220         if (currentY > 270) {
221             doc.addPage();
222             currentY = 20;
223         }
224         doc.setLineWidth(0.3);
225         doc.line(20, currentY, 190, currentY);
226         currentY += 10;
227         doc.setFontSize(8);
228         doc.setFont("helvetica", "italic");
229         doc.text("Generated from Bulletin Board System", 20, currentY);
230         doc.text(`Generated on: ${new Date().toLocaleString()}`, 20, currentY + 5);
231
232         // Save the PDF
233         const filename = `Notice_${noticeId}_${title.replace(/[^a-z0-9]/gi, '_').toLowerCase()}.pdf`;
234         doc.save(filename);
235     }
236     </script>
237
238     @endsection
```

Figure 2.29 KAFA Admin Bulletin Dashboard code after download button implementation

Figures 2.27, 2.28 and 2.29 display the KAFA Admin Bulletin dashboard code after implementation. The changes, specifically the addition of the download button were made between lines 136 and 238 in Visual Studio Code. This implementation allows KAFA Admins to export bulletin notices as PDF files, providing easier access for reference and improving official record-keeping.

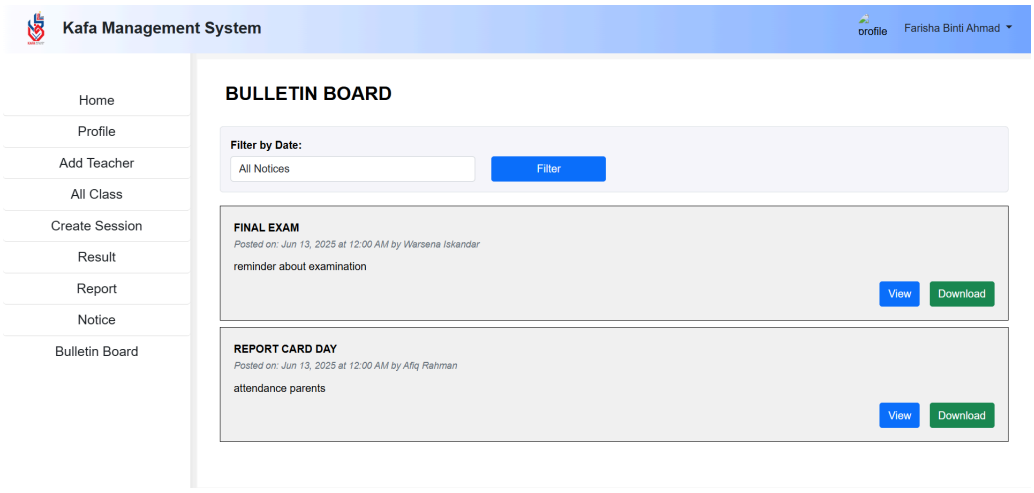


Figure 2.30 KAFA Admin Bulletin dashboard interface after the implementation of the download button

Figure 2.30 shows the updated interface of the KAFA Admin Bulletin dashboard where a download button has been added. This new feature enables KAFA Admins to generate and save bulletin notices in PDF format supporting more efficient documentation.

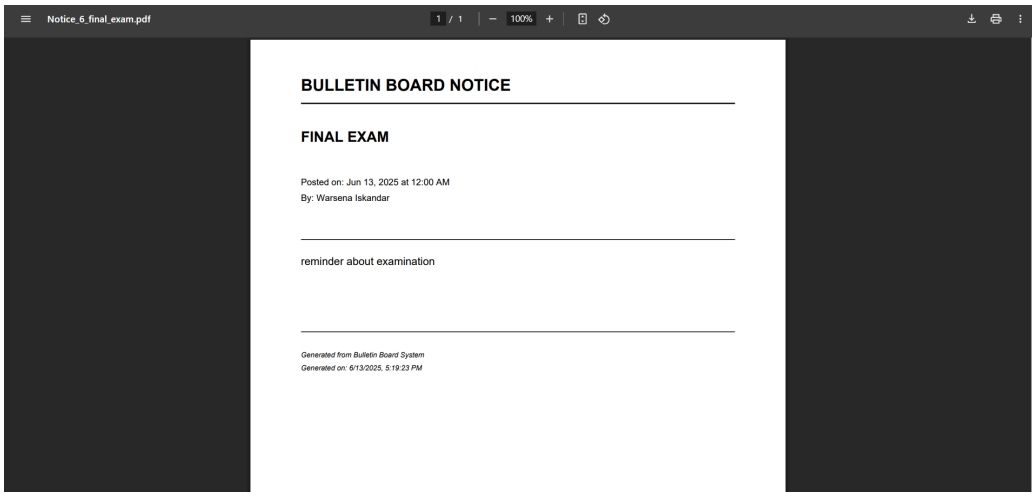


Figure 2.31 Downloaded PDF of Bulletin Notice

Figure 2.31 shows the PDF file generated from the bulletin download feature, illustrating how the notice content appears after being exported by the KAFA Admin for reference or official use.

3. Project Implementation

3.1. Project Tool Description

Several tools were used throughout the software development life cycle to ensure the successful development and deployment of the KAFA Management System (KMS). These tools supported various stages of the process including coding, testing, debugging and version control. The main tools used by the development team were Visual Studio Code (VS Code), Laravel, Laragon and GitHub.

Visual Studio Code served as the primary code editor during the development of KMS. It provided essential features such as IntelliSense (code autocompletion), syntax highlighting, integrated terminal access and Git integration. The development team used VS Code extensively to write code for Laravel controllers, Blade view templates, routes, models and request validation files. The Blade templates allowed for seamless integration of HTML, CSS, JavaScript and PHP logic, enabling efficient construction of dynamic, role-based views for different user interfaces such as KAFA Admin, Teacher, Parent and MUIP Admin.

Laragon was used as the local development environment, offering a portable and fast setup for running PHP applications. It includes Apache, MySQL, PHP and Composer in a pre-configured environment, allowing the team to host and test the KAFA Management System locally. Laragon's user-friendly interface made it easy to manage databases, run Laravel commands and view the system in a browser without complex configuration. This allowed real-time testing and debugging before deployment to the production server.

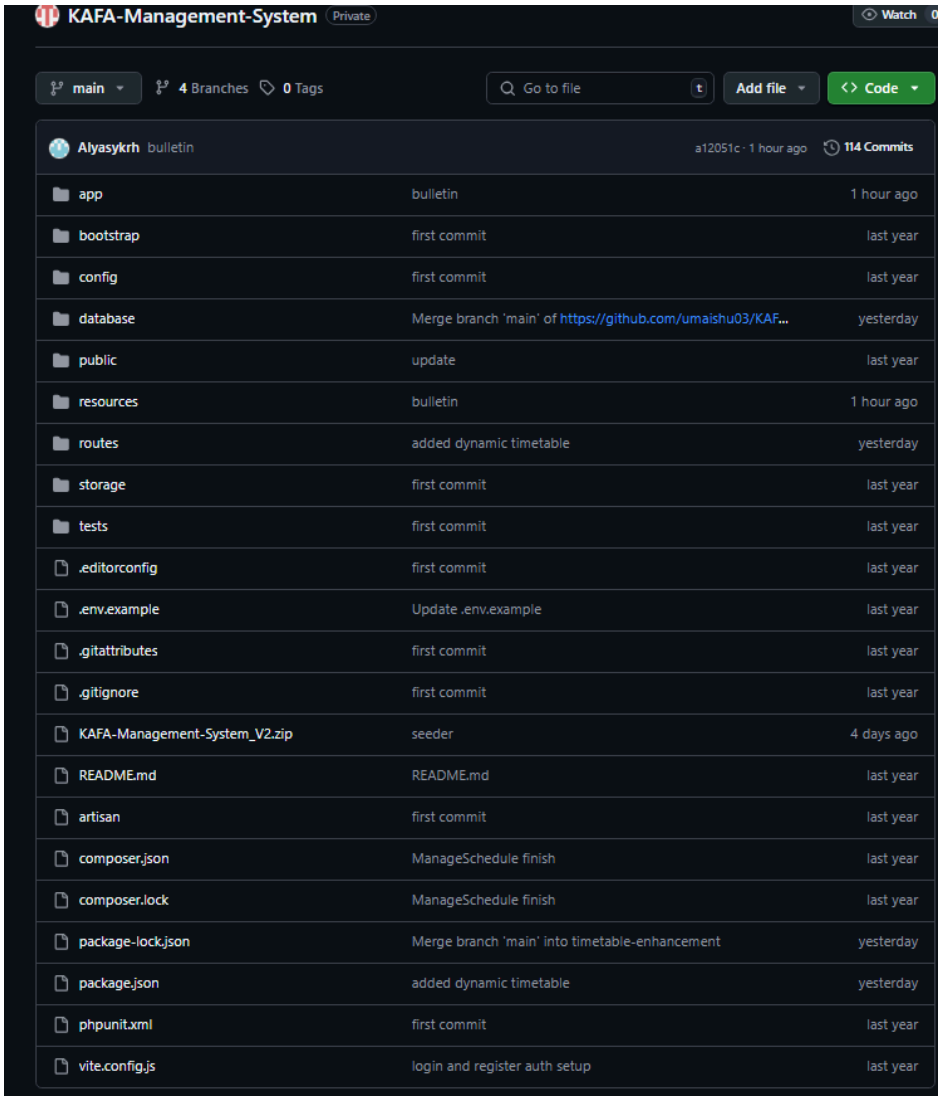
Laravel, the PHP framework used in this project followed the Model-View-Controller (MVC) architecture. It offered structured routing, built-in security features, Eloquent ORM for database interaction and convenient request validation. The use of Laravel helped the team implement modules such as Manage Account, Manage Schedule, Manage Result, Manage Report and Manage Bulletin with efficient back-end logic and clear separation of concerns.

GitHub was used for version control and team collaboration. It enabled the team to track changes, manage branches and merge updates efficiently. By using GitHub, multiple developers could work on different modules concurrently, resolve conflicts and maintain a comprehensive development history. It also served as a centralized and secure repository for all project files.

By utilizing these tools effectively, the development team ensured that KMS was robust and maintainable. Each tool played a key role in different phases of the software life cycle, contributing to a smooth and organized development workflow from initial design to system deployment.

3.2. Project Structure Implementation (screenshot evidence for each criterion)

Step 1: The main view of a project with a repository typically includes a few essential components that help in the efficient management, tracking, and collaboration of developers working on the project. The name of the repository, usually representative of the project's name. The figure below shows how we clone the project into a repository.

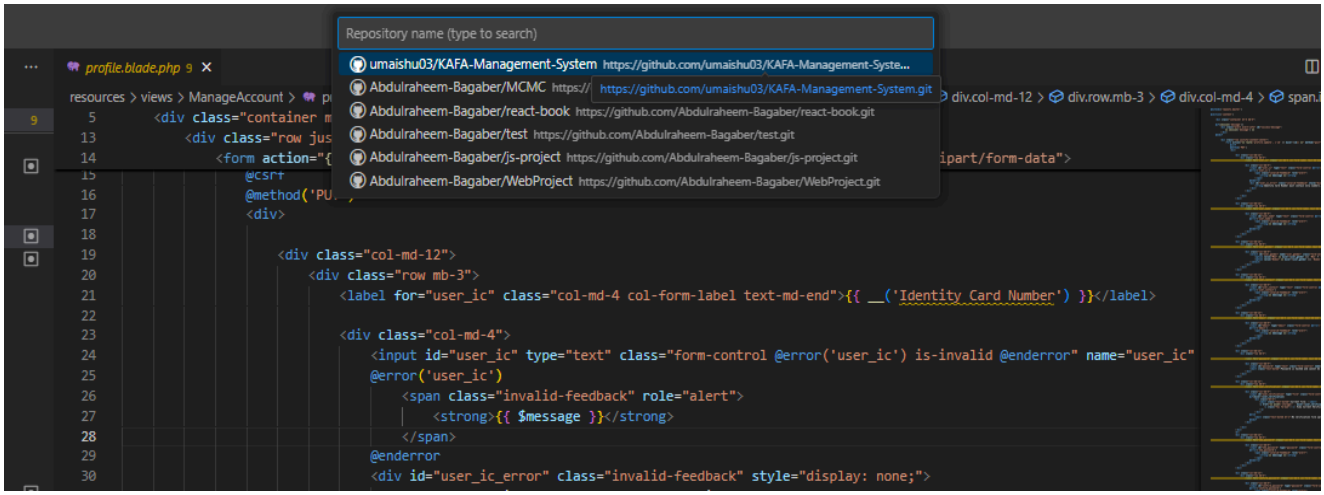


Step 2: Cloning the Repository into Visual Studio Code

To begin maintaining the KAFA Management System, the first step involves cloning the project repository from GitHub into Visual Studio Code (VS Code). As shown in the screenshot, the GitHub repository URL

https://github.com/umaishu03/KAFA-Management-System.git was selected and cloned using the built-in Git integration in VS Code.

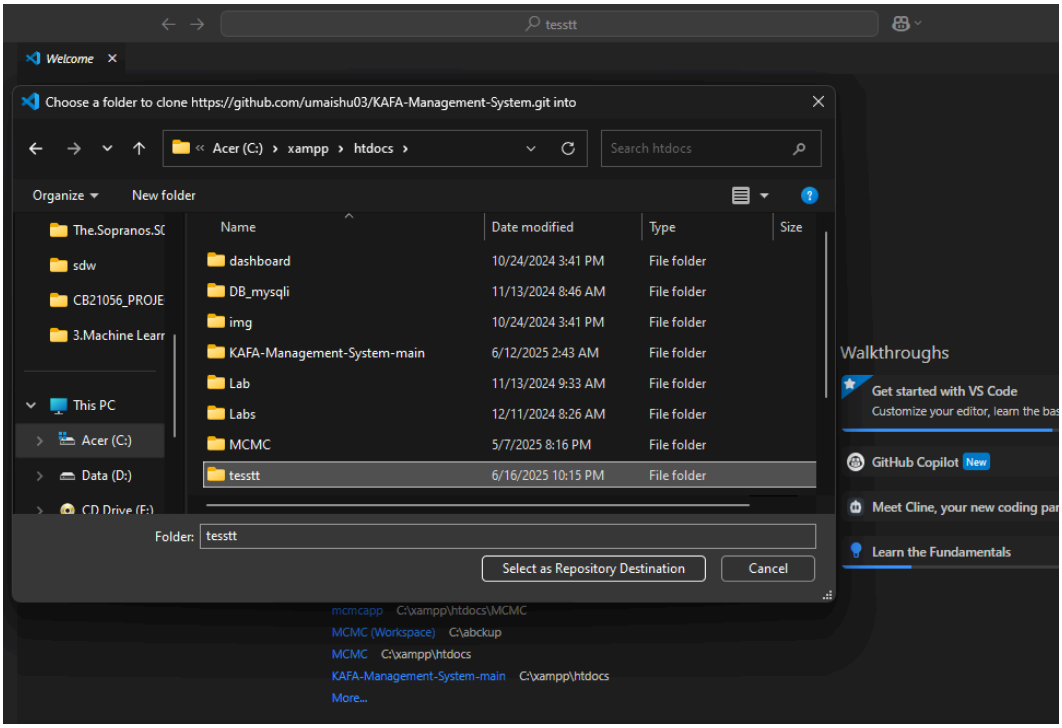
This step initializes a local copy of the source code, allowing team members to access all files and begin reviewing or modifying specific modules. Cloning ensures that the project structure, codebase, and version history are fully available for collaborative development and maintenance tasks.



Step 3: Selecting the Destination Folder for Cloning

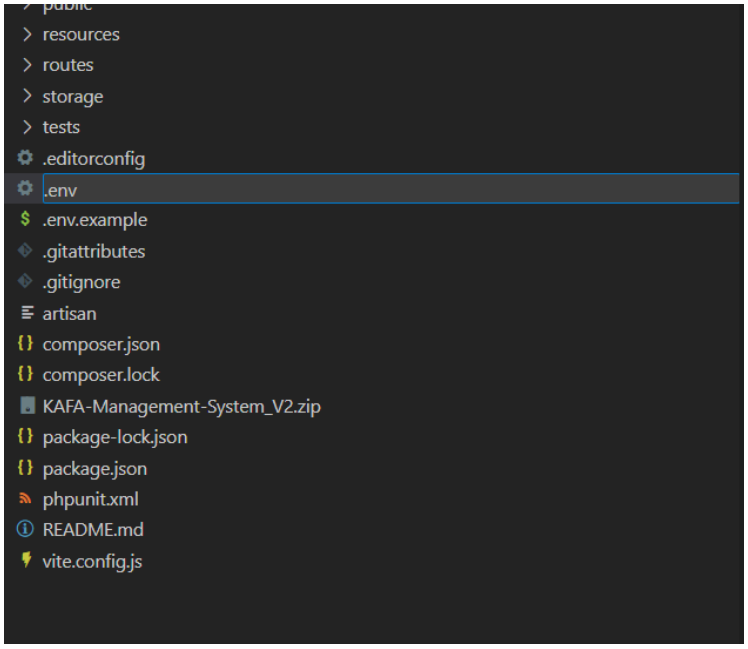
After initiating the repository clone process, the next step is to choose the local directory where the KAFA Management System project will be stored. As shown in the screenshot, the folder path C:\xampp\htdocs\tesstt was selected as the destination.

This step is crucial for setting up the project in a working environment, particularly for PHP-based systems running on local servers like XAMPP. By placing the cloned repository within the htdocs directory, the system can be executed and tested locally via a browser using http://localhost/.



Step 4: Creating the .env File for Environment Configuration

After cloning the project into the local directory, the next step involves setting up the **.env file**, which contains essential environment-specific configurations for running the KAFA Management System. In this step, a new .env file is created by copying the existing .env.example file provided in the project root. This file is then renamed to .env and customized with the appropriate local environment variables

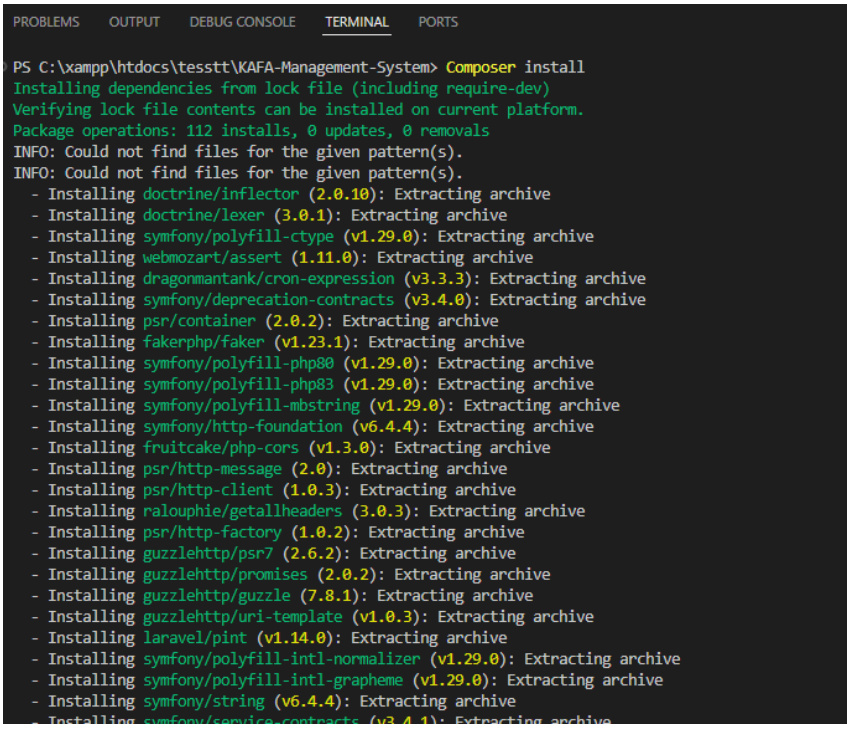


Step 5: Installing Project Dependencies Using Composer

As shown in the terminal output, Composer begins resolving and installing all necessary PHP packages defined in the composer.lock file. These include core libraries such as Laravel framework components, Symfony packages, Guzzle HTTP client, and other third-party dependencies required for the KAFA Management System to function.

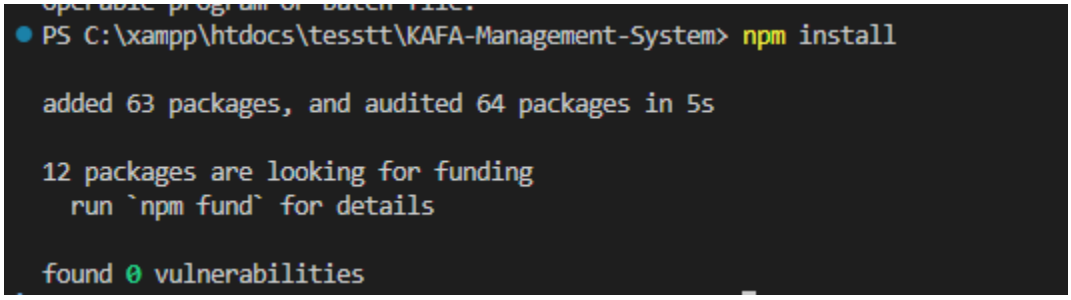
This process ensures the local environment has all the backend dependencies needed to run the application, perform authentication, handle routing, manage sessions, and interact with the database.

Successfully completing this step is essential before the system can be launched or further development and testing can be done.



Step 6: Installing Frontend Dependencies Using NPM

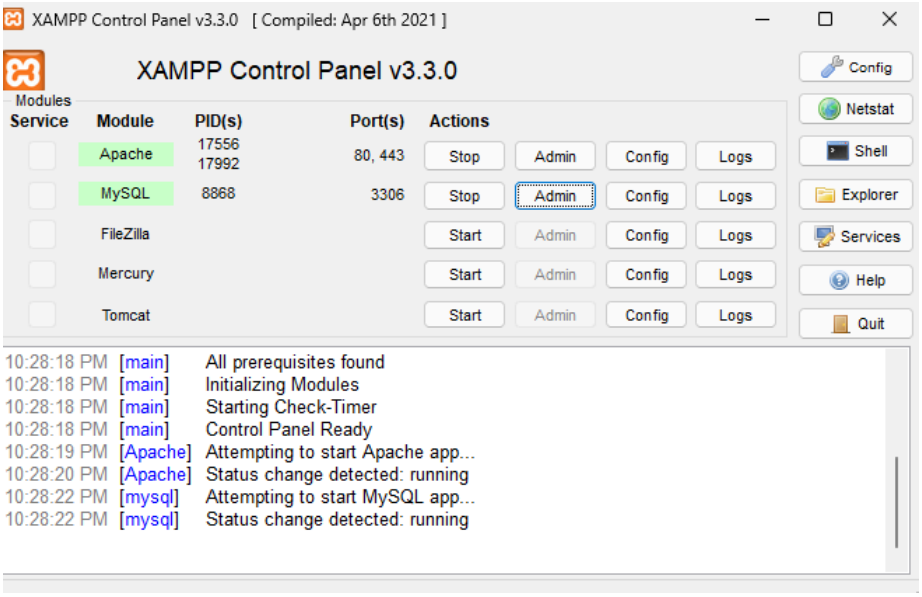
In addition to backend packages, the KAFA Management System also relies on frontend assets such as JavaScript libraries, CSS preprocessors, and build tools. To install these, the command:



Step 7: Running Apache and MySQL Using XAMPP

To host and run the KAFA Management System locally, the necessary services must be activated through XAMPP Control Panel. In this step, both Apache (port 80, 443) and MySQL (port 3306) services are started successfully.

- Apache is used to serve the PHP backend and render Blade templates in the browser.
- MySQL is the database engine used to store system data such as users, profiles, schedules, and results.



Step 8: Running Database Migrations

With the environment configured and services running, the next step is to prepare the database schema using Laravel's migration system. This is done by executing the command:

```
  migrate:status

PS C:\xampp\htdocs\tesstt\KAFA-Management-System> php artisan migrate

   INFO  Running migrations.

  2014_10_12_000000_create_users_table ..................................... 28ms DONE
  2014_10_12_100000_create_password_reset_tokens_table ..................... 6ms DONE
  2014_10_12_100000_create_password_resets_table ........................... 28ms DONE
  2019_08_19_000000_create_failed_jobs_table ............................... 18ms DONE
  2019_12_14_000001_create_personal_access_tokens_table .................... 30ms DONE
  2024_04_03_071431_create_classrooms_table ................................ 6ms DONE
  2024_04_03_071432_create_students_table .................................. 43ms DONE
  2024_04_03_071506_create_activities_table ................................ 39ms DONE
  2024_04_03_071518_create_roles_table ..................................... 4ms DONE
  2024_04_03_071533_create_notices_table ................................... 5ms DONE
  2024_04_03_071540_create_results_table ................................... 5ms DONE
  2024_04_03_071547_create_feedback_table .................................. 39ms DONE
  2024_05_04_132530_create_examinations_table .............................. 5ms DONE
  2024_05_07_103751_create_subjects_table .................................. 6ms DONE
  2025_06_14_193458_add_profile_picture_to_users_table ..................... 15ms DONE

PS C:\xampp\htdocs\tesstt\KAFA-Management-System>
```

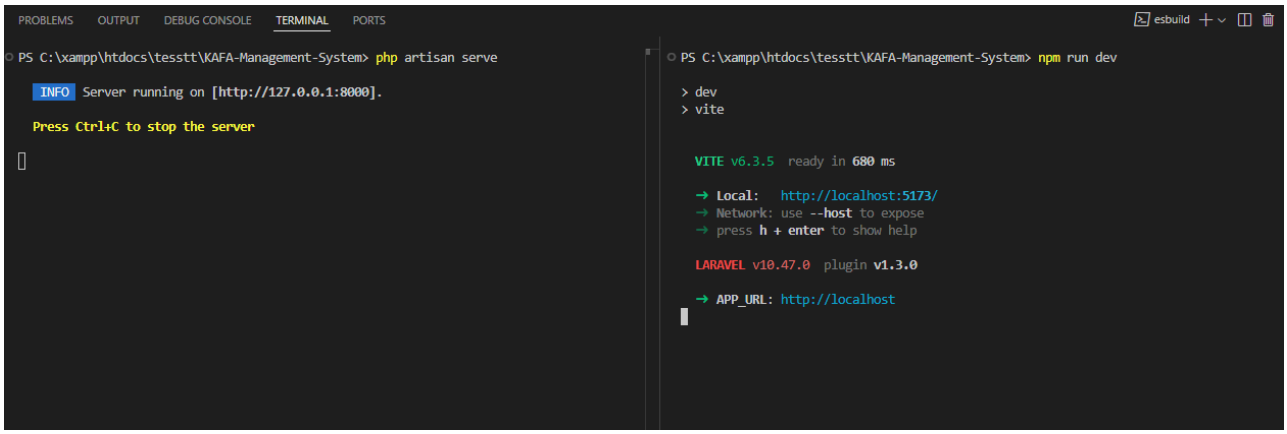
	Table	Action	Rows	Type	Collation	Size	Overhead
☐	activities	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
☐	classrooms	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	examinations	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
☐	feedback	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	migrations	Browse Structure Search Insert Empty Drop	15	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	notices	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	password_resets	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
☐	password_reset_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	personal_access_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
☐	results	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	roles	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	students	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
☐	subjects	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	users	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
	15 tables	Sum	15	InnoDB	utf8mb4_general_ci	352.0 KiB	0 B

Step 9: Running the Laravel and Frontend Development Servers

To launch the KAFA Management System locally, both the Laravel backend and Vite frontend development servers must be started.

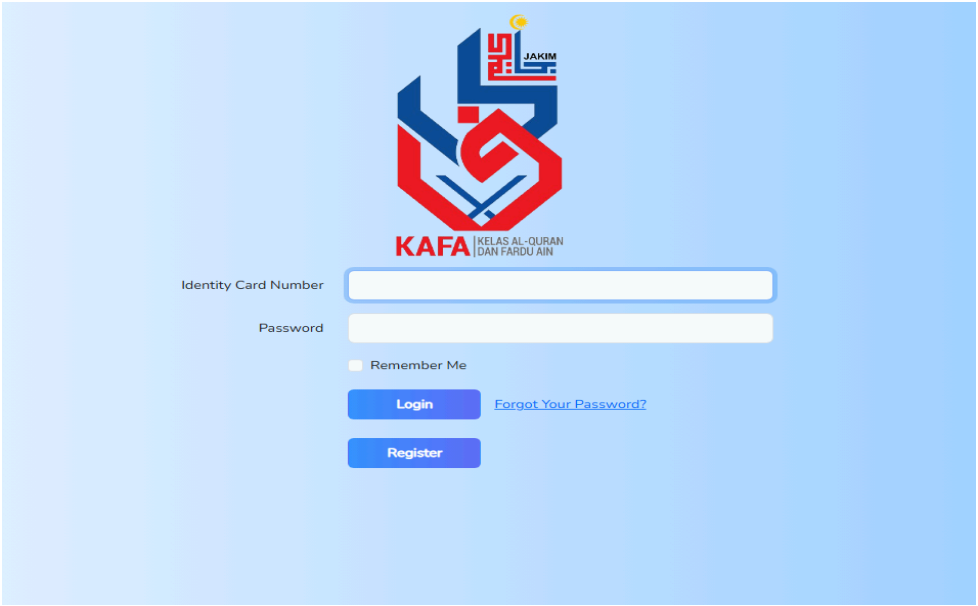
- The command php artisan serve was executed to start the Laravel backend server, which successfully began listening on http://127.0.0.1:8000. This allows backend routes, authentication, and API endpoints to function.

- In parallel, the command npm run dev was executed to start the Vite frontend build tool, which compiles and serves the system's frontend assets such as CSS, JavaScript, and Vue components. The Vite server started successfully and served assets via http://localhost:5173/.

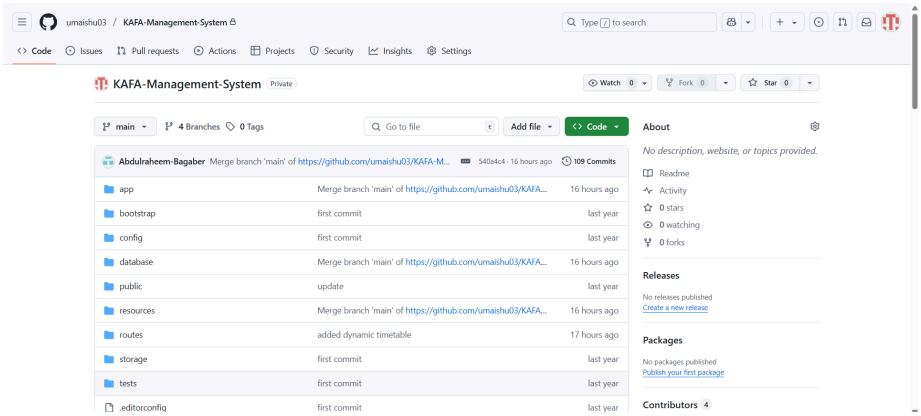


Step 10: Verifying the Application Interface in Browser

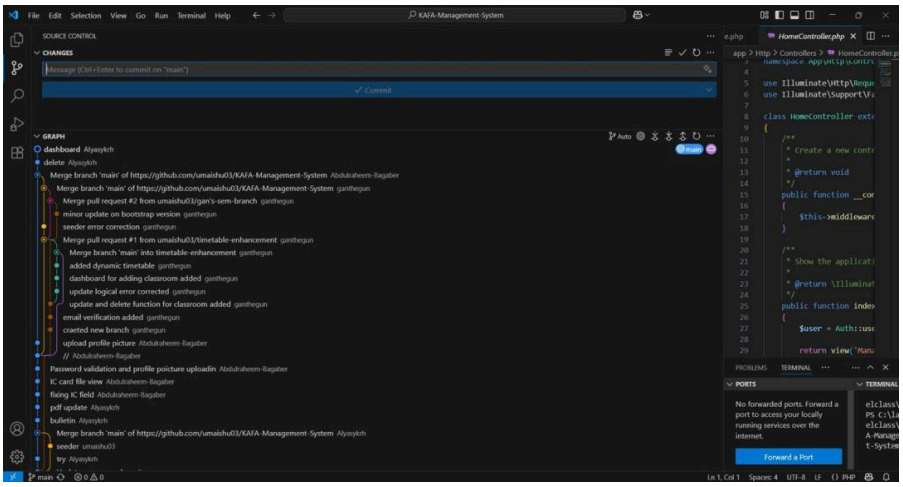
After starting both the Laravel backend server and the Vite frontend development server, the KAFA Management System can now be accessed via the browser using the link:http://127.0.0.1:8000



3.3. The main view of the project with a repository



Project Repository - Github

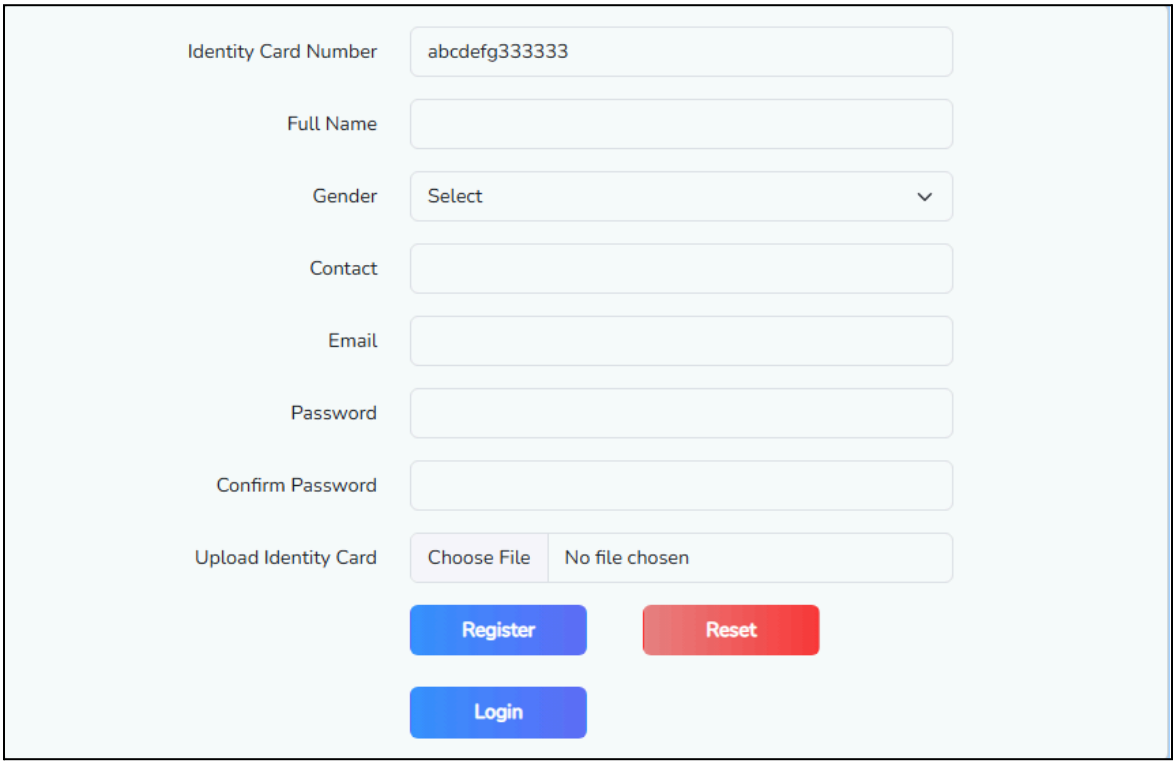


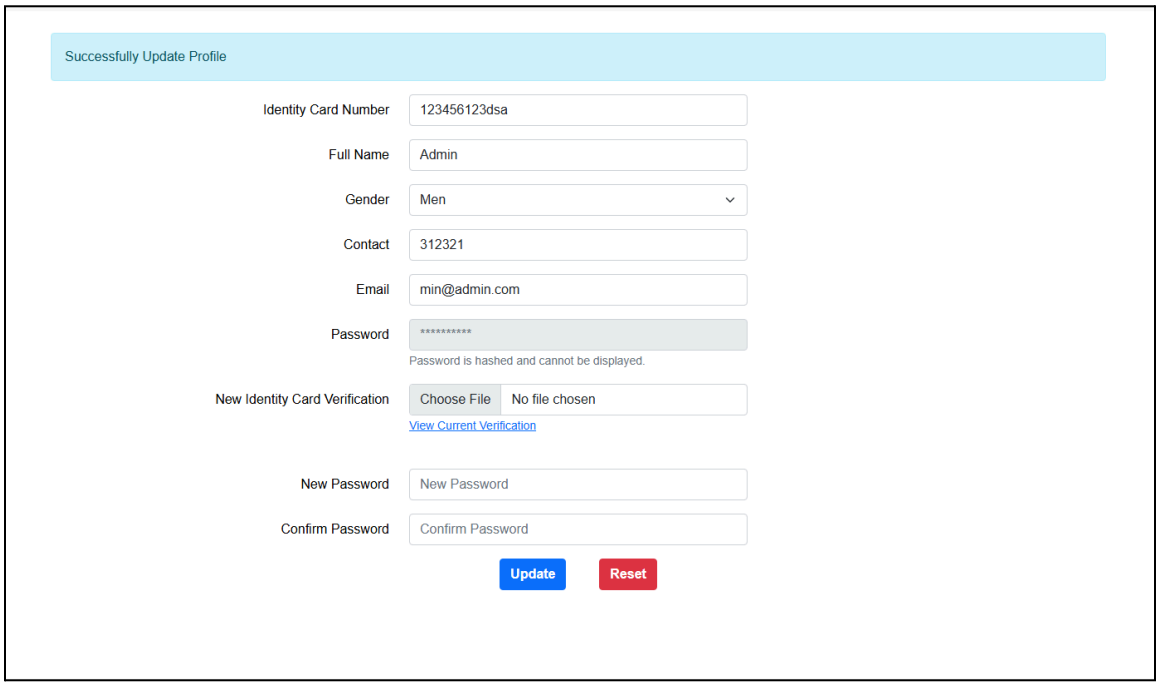
Project Environment - Visual Studio Code

3.3.1. View of Project (based on each module: before and after implementation)

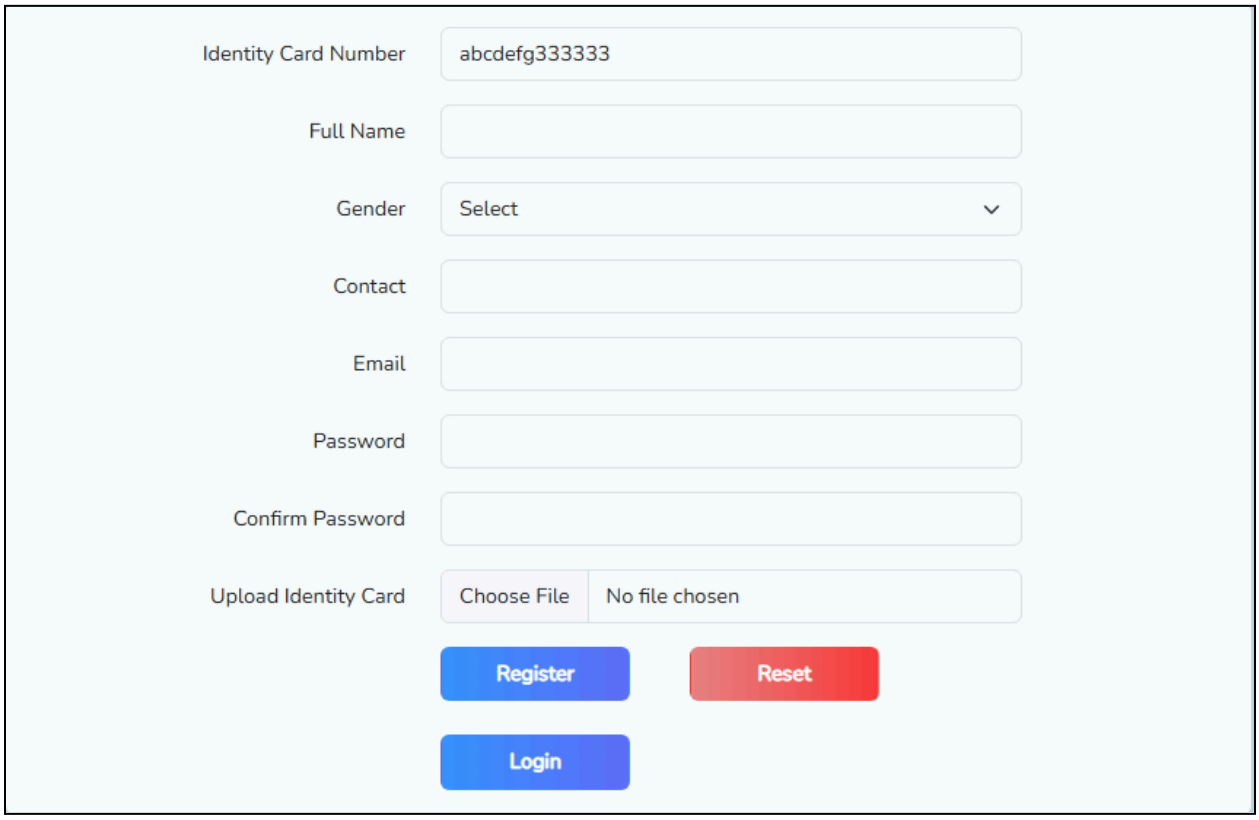
3.3.1.1. Manage Profile

Before Implmenetation (IC Number Field Validation Correction) : IC number field accepting letters with no instructions of minimum digits or numbers only

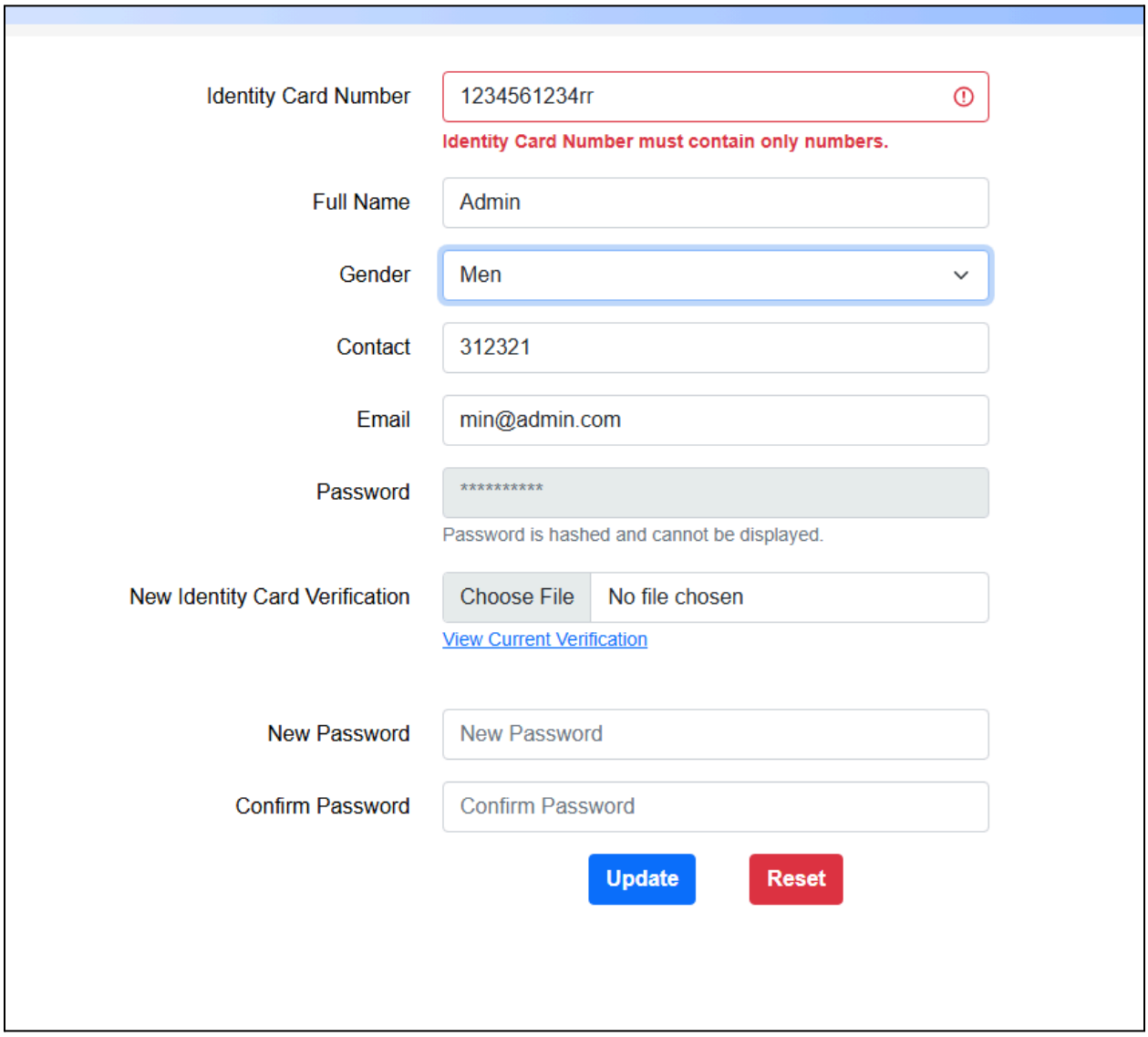


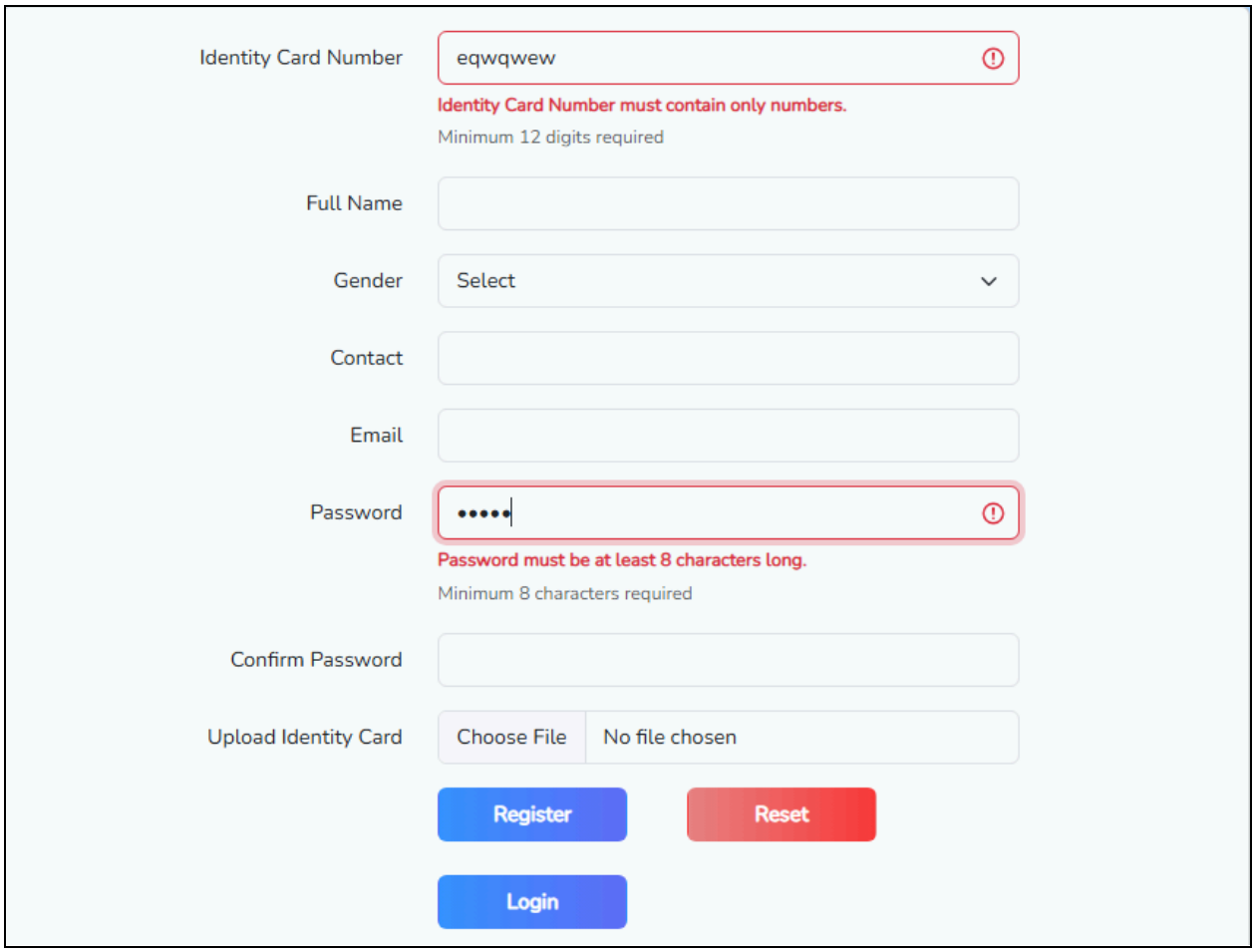


Before Implmenetation (Implementation of Profile Picture Upload in Registration): Users have a default generated profile picture and no option to upload a profile picture

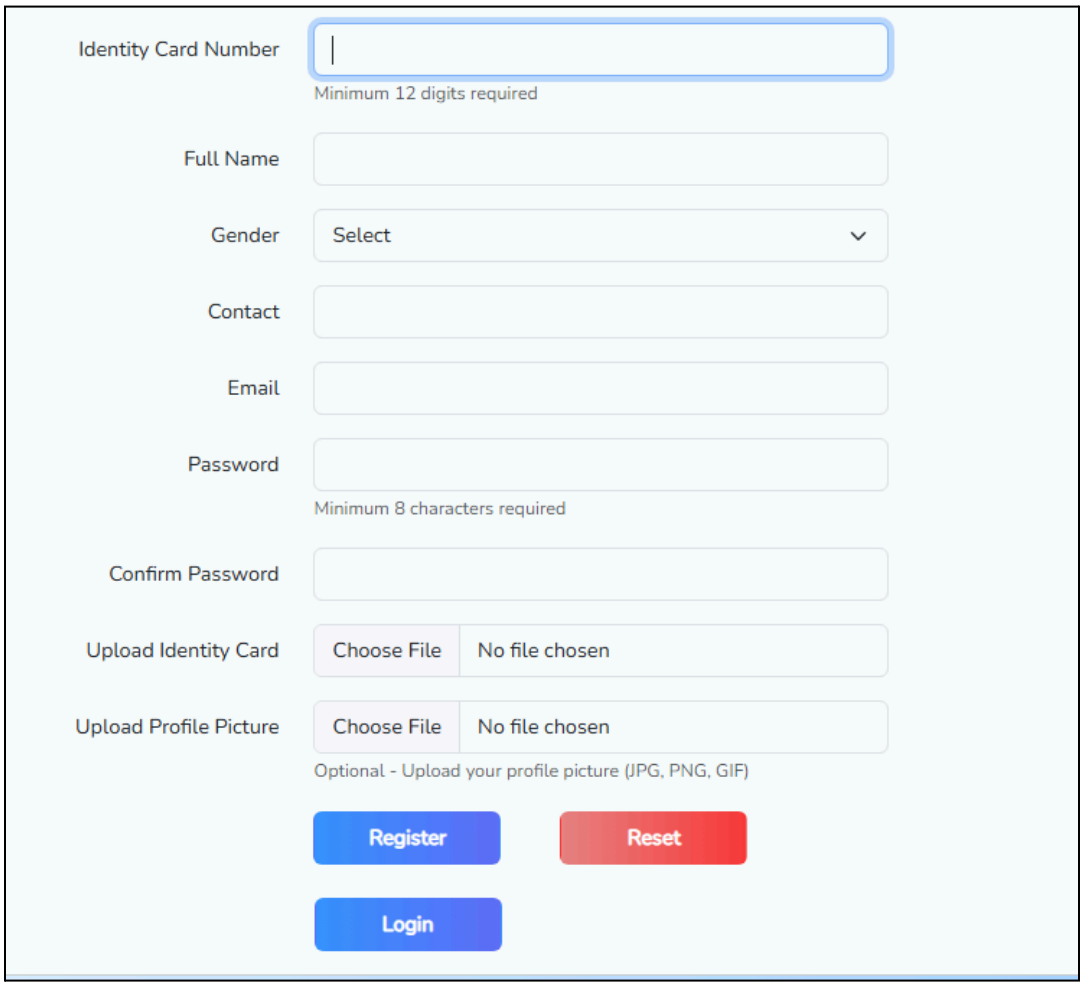


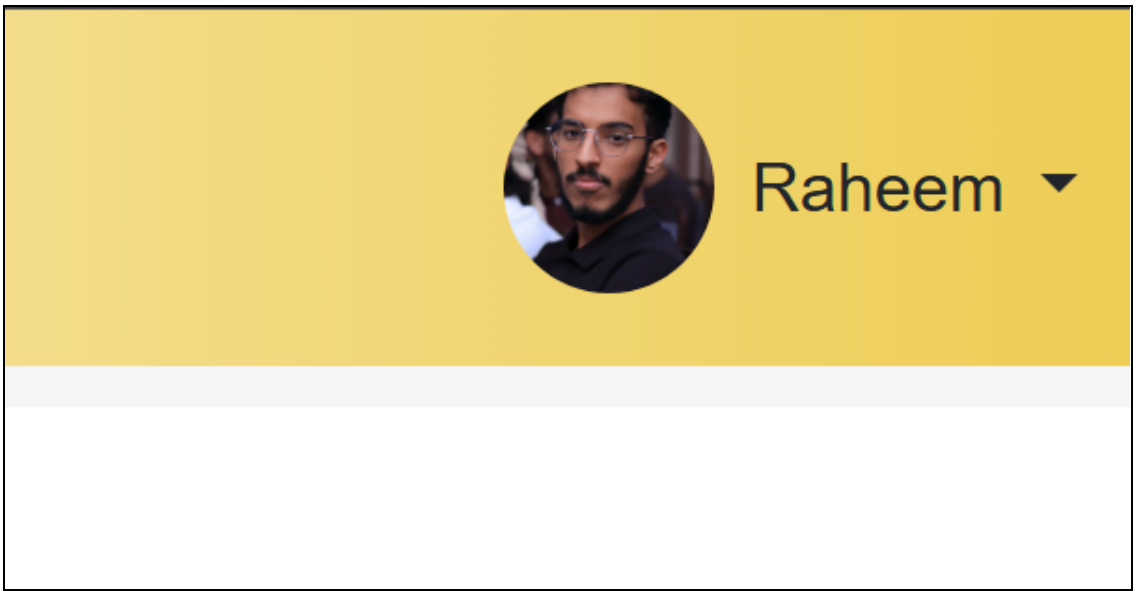
After Implmentation (IC Number Field Validation Correction**):** The system dose not allow any other type of input other than numbers, and should exceed 11 numbers. Updated the same field for IC number for registration page



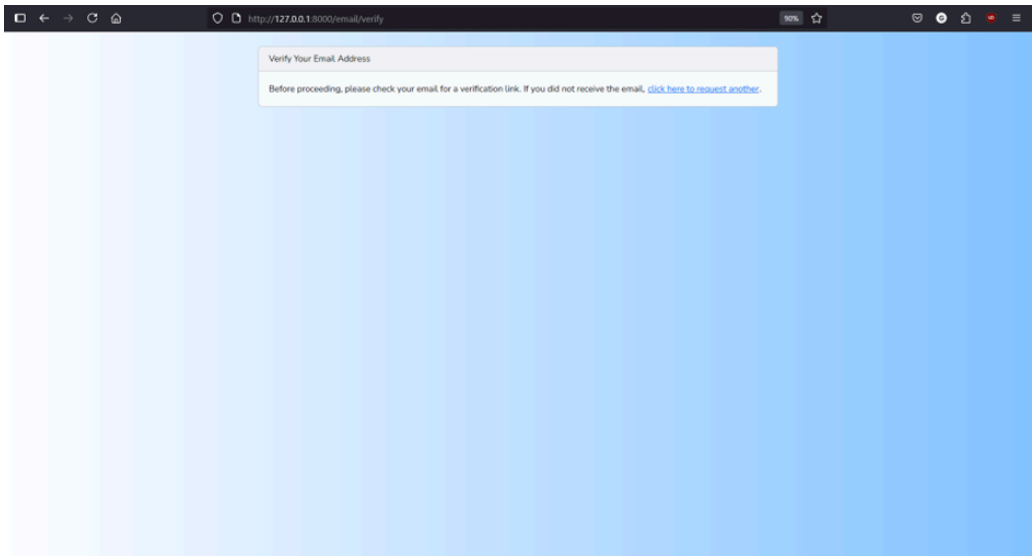

Identity Card Number
eqwqwew
Identity Card Number must contain only numbers.
Minimum 12 digits required
Full Name
Gender
Select
Contact
Email
Password
Password must be at least 8 characters long.
Minimum 8 characters required
Confirm Password
Upload Identity Card
Choose File
No file chosen
Register
Reset
Login

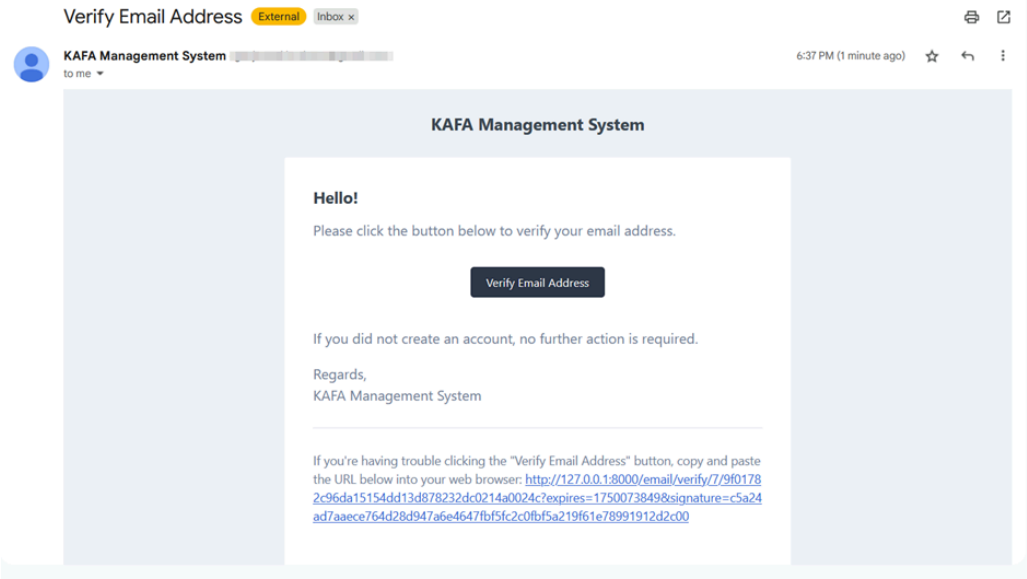
After Implmentation (Implementation of Profile Picture Upload in Registration): Users can upload a personal profile picture during registration form to be displayed on the profile





3.3.1.2. Manage Schedule

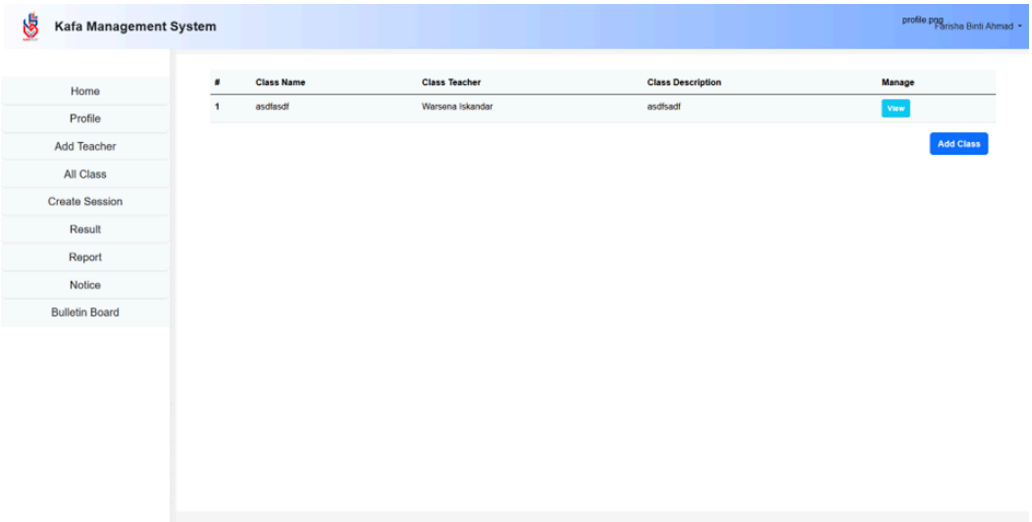
Type of Maintenance Chosen 1 Manage Profile: Email Verification
<u>Before Implementation:</u> No interface is available for email verification.
<u>After Implementation :</u>    User is redirected to the verification page after registration.



User is receives an email for account's email verification.

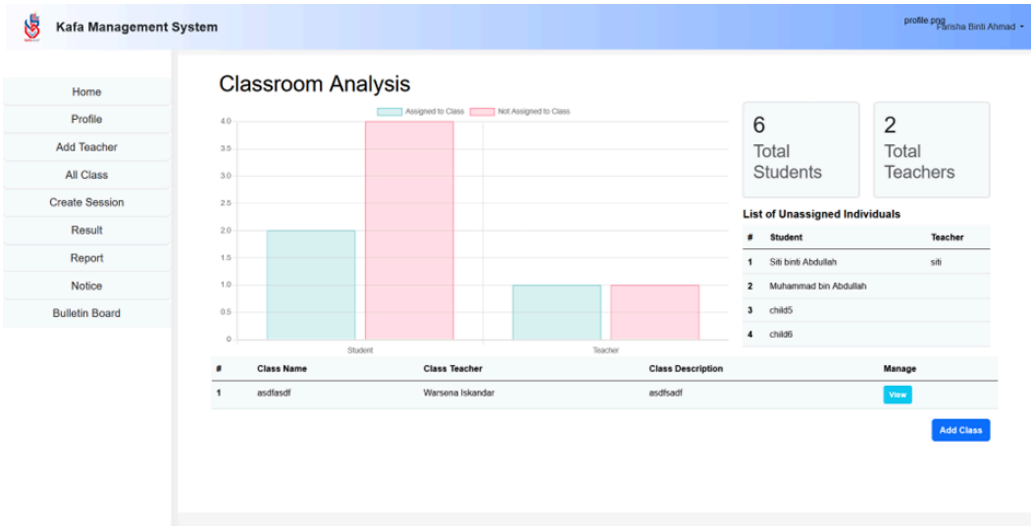
Type of Maintenance Chosen 2
Manage Schedule: Analytical Dashboard

Before Implementation:



KAFA admin cannot know how many teachers and student they have not assign easily.

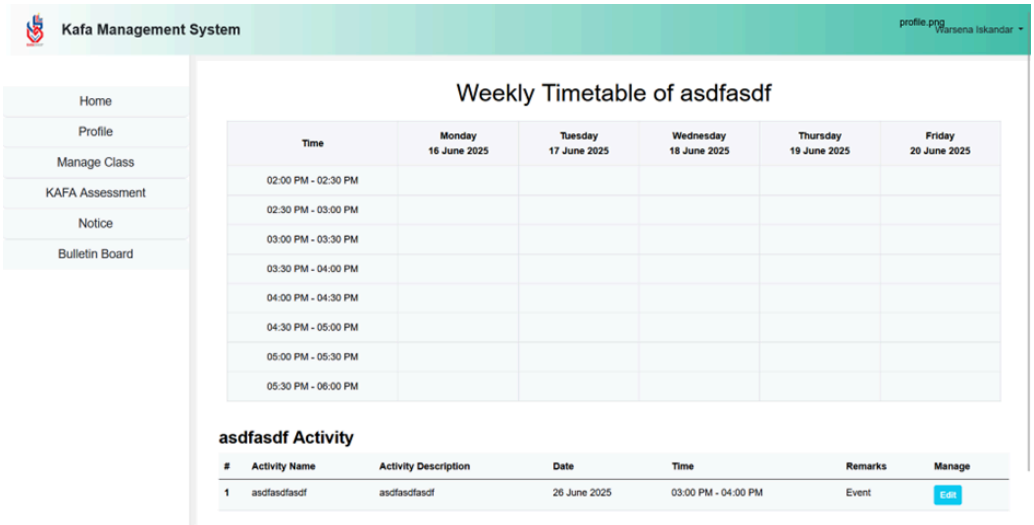
After Implementation :



KAFA admin can easily know the exact number of unassigned teachers and students.

Type of Maintenance Chosen 3
Manage Schedule: Intuitive Timetable Navigator Button

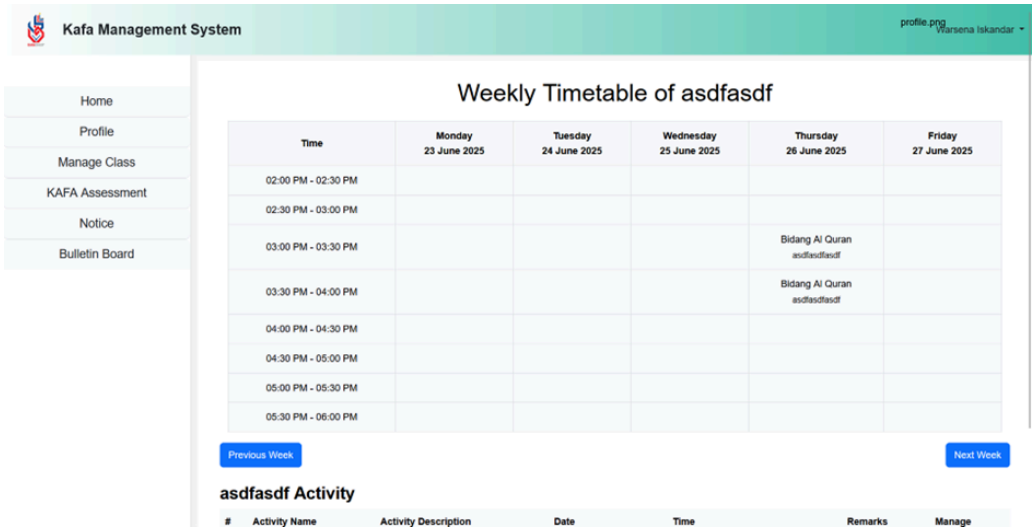
<u>Before Implementation:</u>



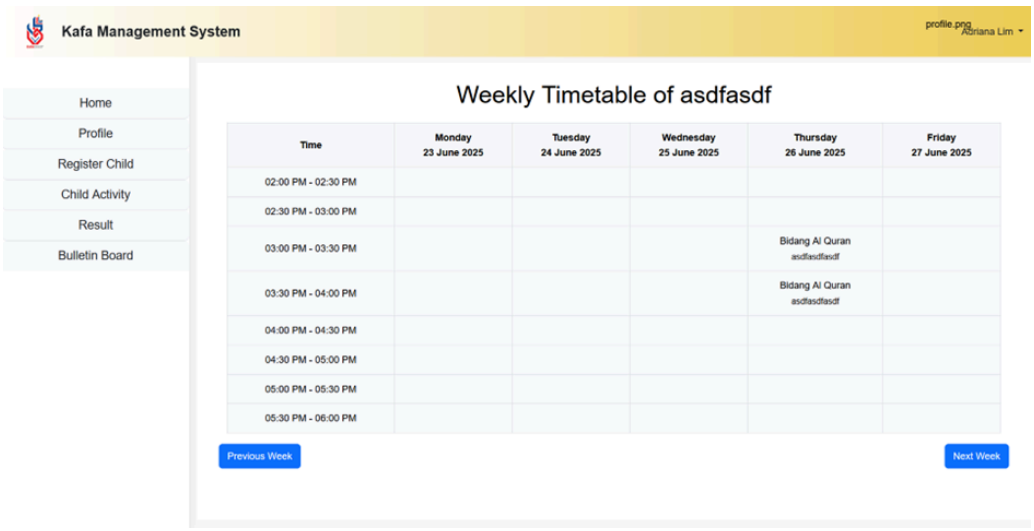
Teacher cannot see the timetable for the next and previous week easily.

Students cannot see their timetable for the next and previous week easily.

After Implementation :



Teachers can see the timetable for the next and previous week easily by using the next and previous button to display the desired week of timetable.

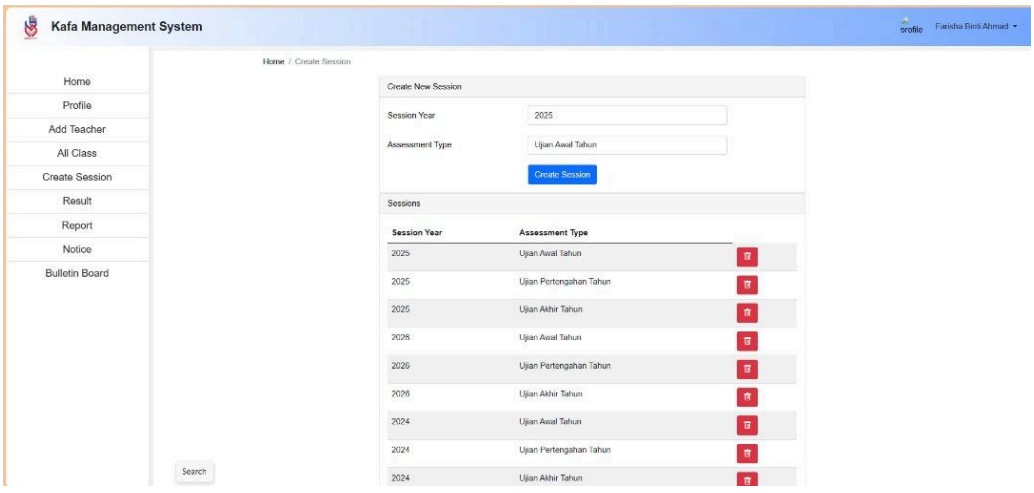


Students can see their timetable for the next and previous week easily by using the next and previous button to display the desired week of timetable.

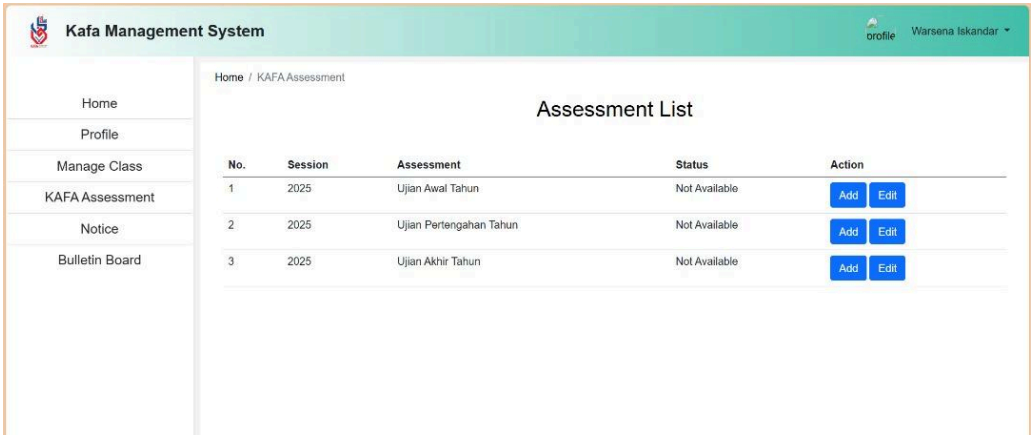
3.3.1.3. Manage Result (Umairah Shuhada Binti Ahmad | CB22090)

Type of Maintenance Chosen 1
Manage Result: Correction the business logic flow

Before Implementation:

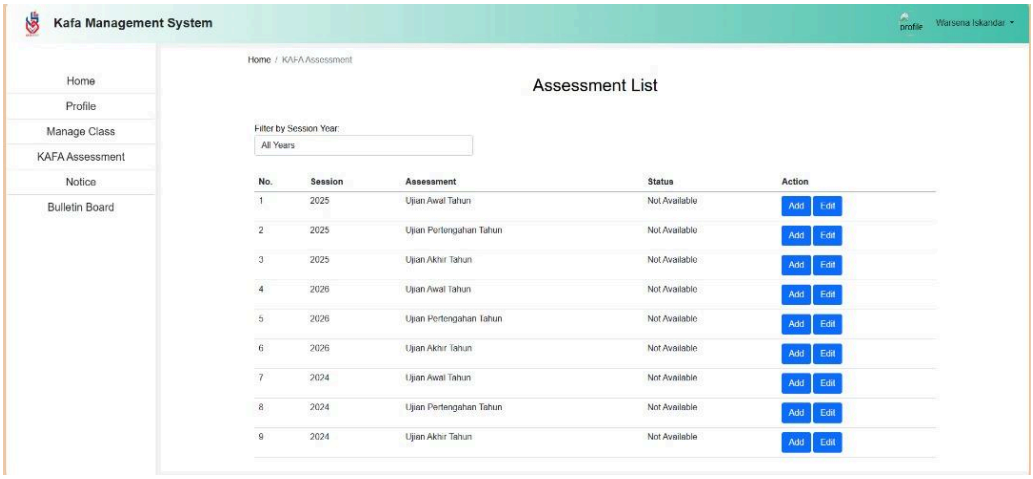


Create session interface show all the list added session year and assessment - KAFA Admin



KAFA Assessment interface did not tie the added session year with the Teacher module at creation by the KAFA Admin - Teacher

<u>After Implementation :</u>

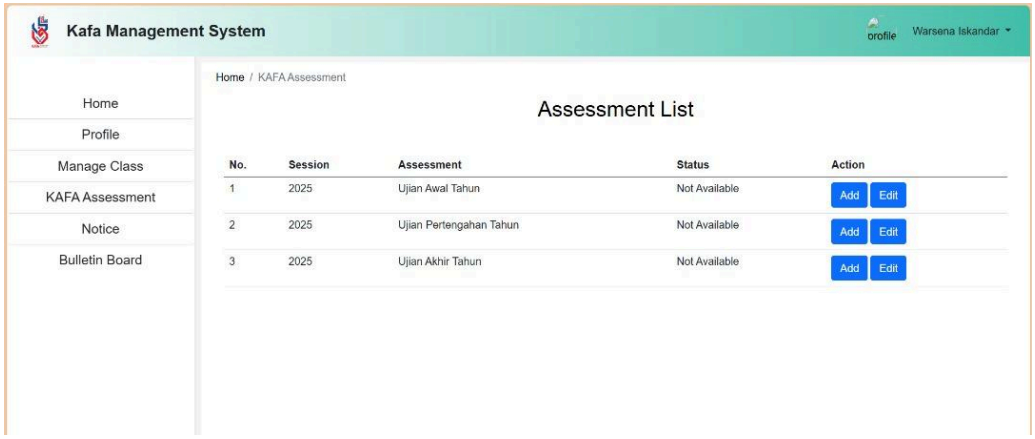


Kafa Assessment interface after correct the business logic flow - Teacher

Type of Maintenance Chosen 2

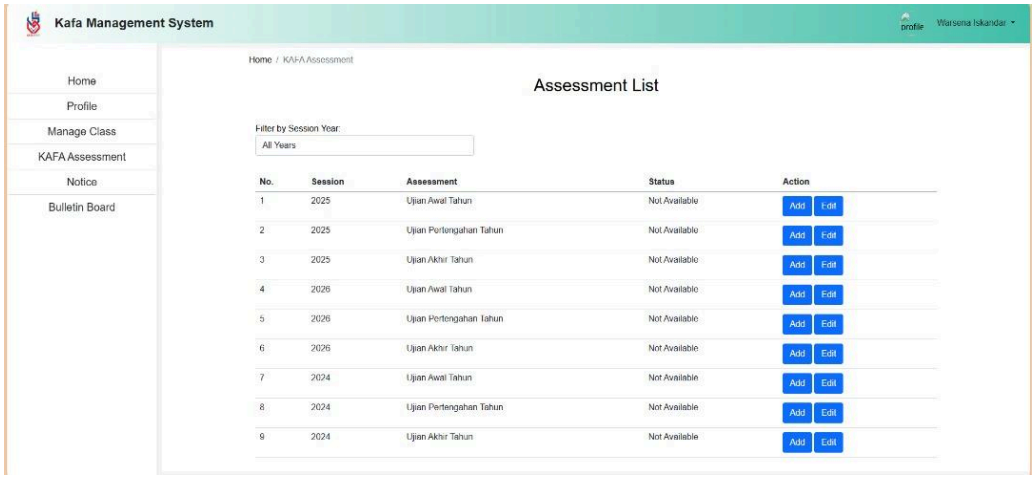
Manage Result: Addition of filter button academic session year at assessment list.

<u>Before Implementation:</u>

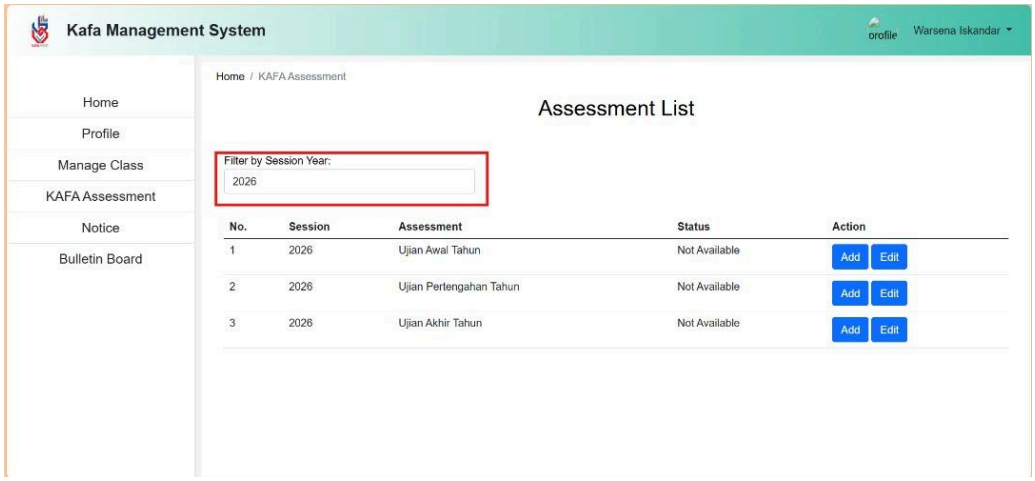


KAFA Assessment interface before the implementation of the filter button -Teacher

<u>After Implementation :</u>



KAFA Assessment interface after the implementation of the filter session year -Teacher

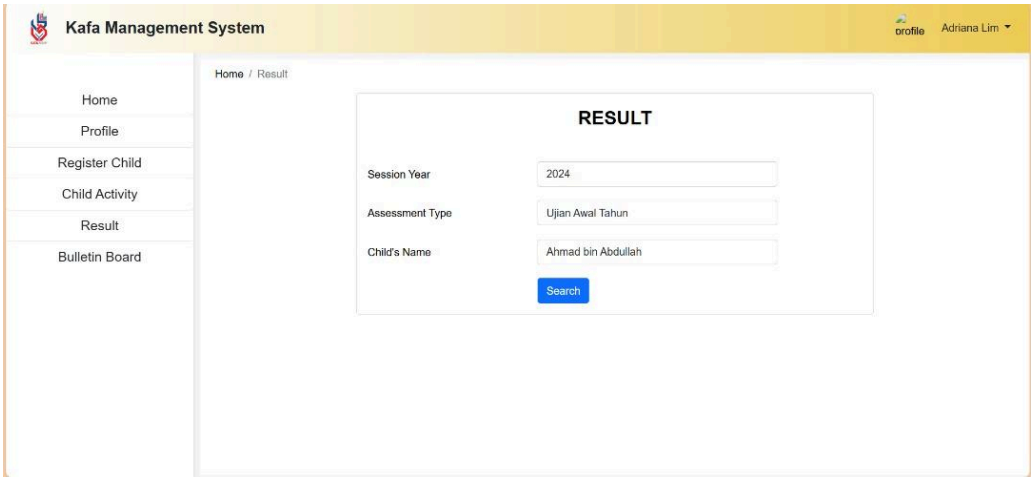


KAFA Assessment interface show the listed assessment based on session year

after the implementation of the filter button -Teacher

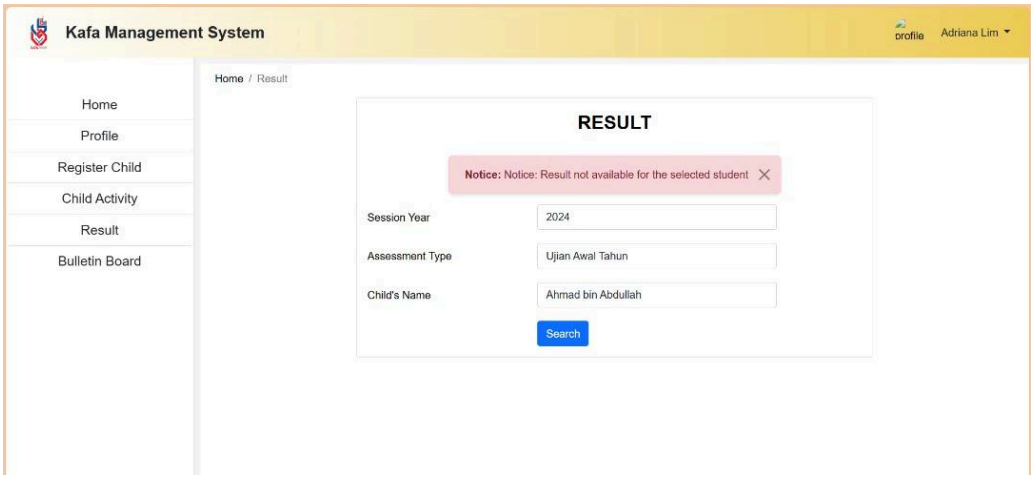
Type of Maintenance Chosen 3
Manage Result: Addition message dialog for missing results

Before Implementation:



Result interface before implement the error message when the user fill the wrong form or data did not available - Parent

After Implementation :



Result interface show after implement the error message - Parent

3.3.1.4. Manage Bulletin (Nur Alya Syakirah Binti Nasarudin | CB22141)

Type of Maintenance Chosen 1
Manage Bulletin: Addition of filter and poster info on bulletin board

Before Implementation:

```
{
    //Display all notices
    public function allnotices(Request $request)
    {
        $status = $request->notice_status;

        if ($status === 'Pending' || $status === 'Approved' || $status === 'Rejected') {
            $notices = Notice::where('notice_status', $status)->get();

            return view('ManageBulletin.all_notices', compact('notices', 'status'));
        } else {
            $notices = Notice::all(); // fetch all notices

            $status = "null";

            return view('ManageBulletin.all_notices', compact('notices', 'status'));
        }
    }

```

Figure 2.32 Bulletin Board code before filter button implementation

```
<div class="container">
    @foreach($notices as $notice)
    @if($notice->notice_status == 'Approved')
    <div class="row">
        <div class="col">
            <div style="border: 1px solid black; background-color:
#f2f2f2; padding: 10px; margin-bottom: 10px;">
                <div style="padding-left: 10px; padding-right: 10px; margin-top: 10px;">
                    <b style="text-transform: uppercase;">{{ $notice->notice_title }}</b>
                    <br>
                    <div style="margin-top: 10px;">
                        {{ $notice->notice_text }}
                    </div>
                    <div style="text-align: right; margin-top: 10px; margin-bottom: 10px;">
                        <a href="{{ route('selectednotices', ['id' => $notice->id]) }}" class="btn btn-primary" style="margin-right: 10px;">
                            {{ __('View') }}
                        </a>
                    </div>
                </div>
            </div>
        </div>
    </div>
    @endif
    @endforeach
```

Figure 2.33 Bulletin Board code before notice's name and date implementation

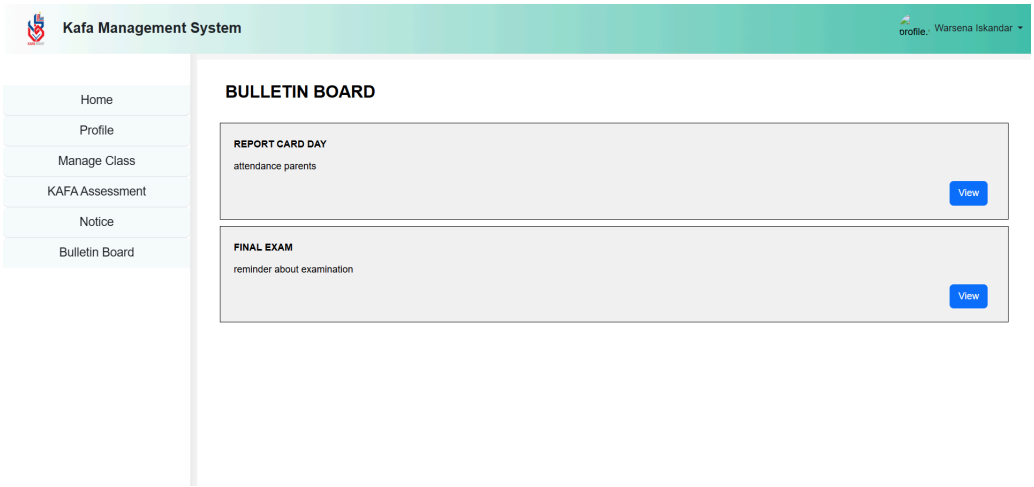


Figure 2.34 Bulletin Board interface before the implementation of the filter button and notice's name and date (Teacher dashboard)

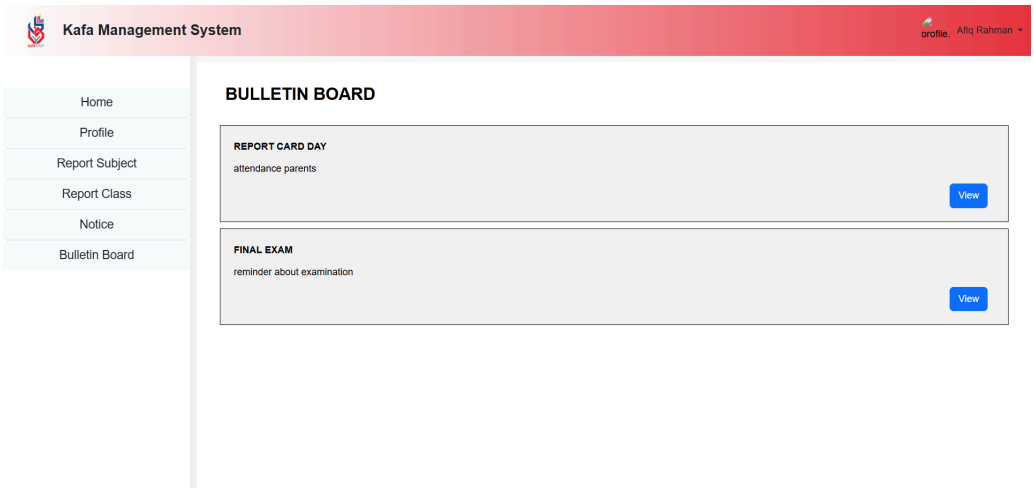


Figure 2.35 Bulletin Board interface before the implementation of the filter button and notice's name and date (MUIP Admin dashboard)

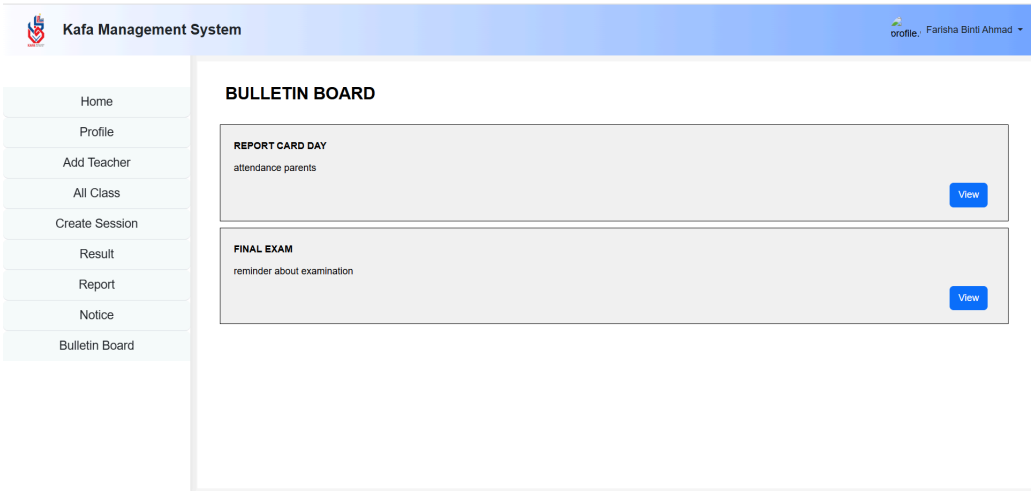


Figure 2.36 Bulletin Board interface before the implementation of the filter button and notice's name and date (KAFA Admin dashboard)

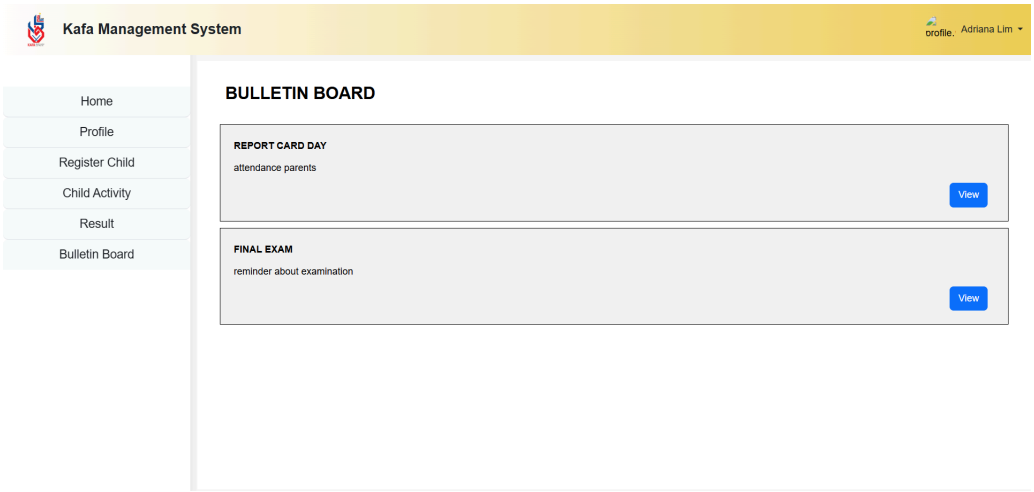


Figure 2.37 Bulletin Board interface before the implementation of the filter button and notice's name and date (Parent dashboard)

After Implementation :

```php
    //Display all notices with filtering options
    public function allnotices(Request $request)
    {
        $status = $request->notice_status;
        $dateFilter = $request->date_filter;
        $customStartDate = $request->start_date;
        $customEndDate = $request->end_date;

        // Start building the query
        $query = Notice::query();

        // Apply status filter
        if ($status === 'Pending' || $status === 'Approved' || $status === 'Rejected') {
            $query->where('notice_status', $status);
        }

        // Apply date filters
        if ($dateFilter) {
            $now = Carbon::now();

            switch ($dateFilter) {
                case 'today':
                    $query->whereDate('notice_submission_date', $now->toDateString());
                    break;

                case 'yesterday':
                    $query->whereDate('notice_submission_date', $now->subDay()->toDateString());
                    break;
```

Figure 2.38 Bulletin Board code after filter button implementation

```php
        if ($dateFilter) {
            switch ($dateFilter) {
                case 'yesterday':
                case 'this_week':
                    $query->whereBetween('notice_submission_date', [
                        $now->startOfWeek()->toDateString(),
                        $now->endOfWeek()->toDateString()
                    ]);
                    break;

                case 'last_week':
                    $startOfLastWeek = $now->subWeek()->startOfWeek();
                    $endOfLastWeek = $now->endOfWeek();
                    $query->whereBetween('notice_submission_date', [
                        $startOfLastWeek->toDateString(),
                        $endOfLastWeek->toDateString()
                    ]);
                    break;

                case 'this_month':
                    $query->whereMonth('notice_submission_date', $now->month)
                          ->whereYear('notice_submission_date', $now->year);
                    break;

                case 'last_month':
                    $lastMonth = $now->subMonth();
                    $query->whereMonth('notice_submission_date', $lastMonth->month)
                          ->whereYear('notice_submission_date', $lastMonth->year);
                    break;
```

Figure 2.39 Bulletin Board code after filter button implementation

```blade
<!-- Notices Display Section -->
<div class="container">
    @if($notices->count() > 0)
        @foreach($notices as $notice)
        @if($notice->notice_status == 'Approved')
        <div class="row">
            <div class="col">
                <div style="border: 1px solid black; background-color: #f2f2f2; padding: 10px; margin-bottom: 10px;">
                    <div style="padding-left: 10px; padding-right: 10px; margin-top: 10px;">
                        <b style="text-transform: uppercase;">{{ $notice->notice_title }}</b>
                        <br>
                        <small style="color: #6c757d; font-style: italic;">
                            Posted on: {{ \Carbon\Carbon::parse($notice->notice_submission_date)->format('M d, Y \a\t g:i A') }}
                            by {{ $notice->user->user_name ?? 'Unknown' }}
                        </small>
                        <div style="margin-top: 10px;">
                            {{ $notice->notice_text }}
                        </div>
                        <div style="text-align: right; margin-top: 10px; margin-bottom: 10px;">
                            <a href="{{ route('selectednotices', ['id' => $notice->id]) }}" class="btn btn-primary" style="margin-right: 10px;">
                                {{ __('View') }}
                            </a>
                            @if(Auth::check() && Auth::user()->role_id == 2)
                                <button type="button" class="btn btn-success" onclick="downloadNotice({{ $notice->id }}, '{{ addslashes($notice->n
                                    <i class="fas fa-download"></i> {{ __('Download') }}
                                </button>
                            @endif
                        </div>
                    </div>
                </div>
```

Figure 2.40 Bulletin Board code after notice's name and date implementation

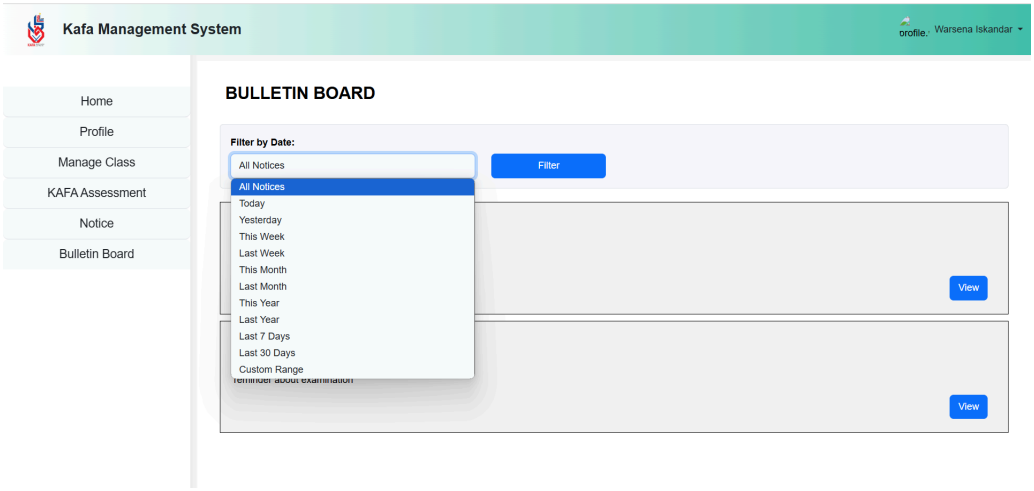


Figure 2.41 Bulletin Board interface after the implementation of the filter button (Teacher dashboard)

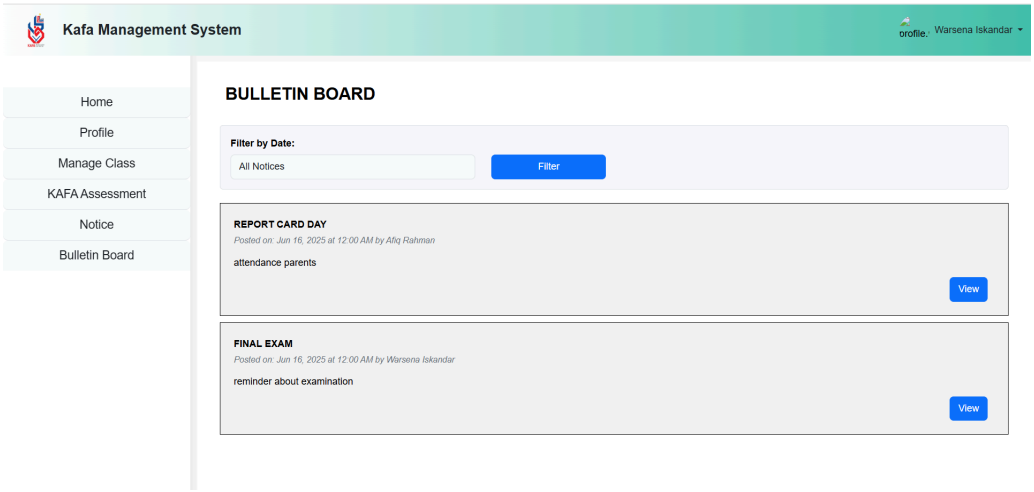


Figure 2.42 Bulletin Board interface after the implementation of the notice's name and date (Teacher dashboard)

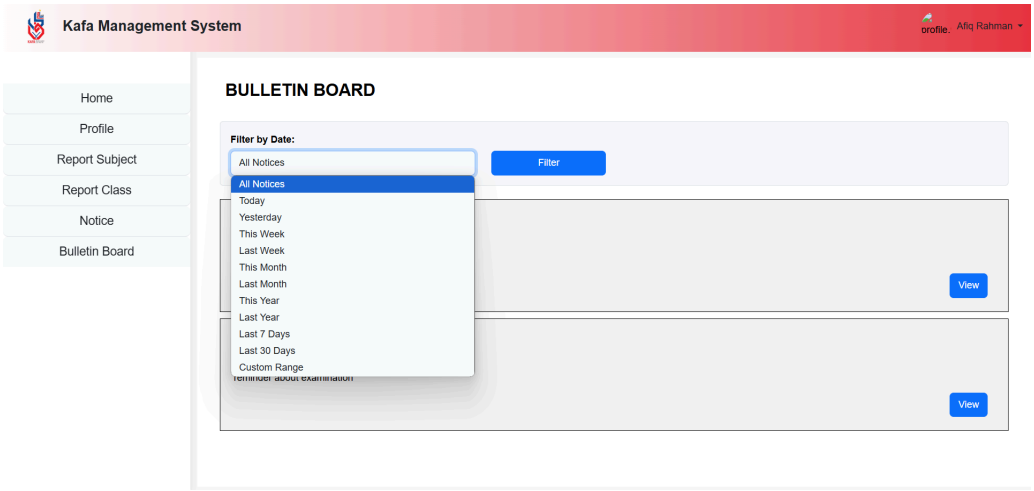


Figure 2.43 Bulletin Board interface after the implementation of the filter button (MUIP Admin dashboard)

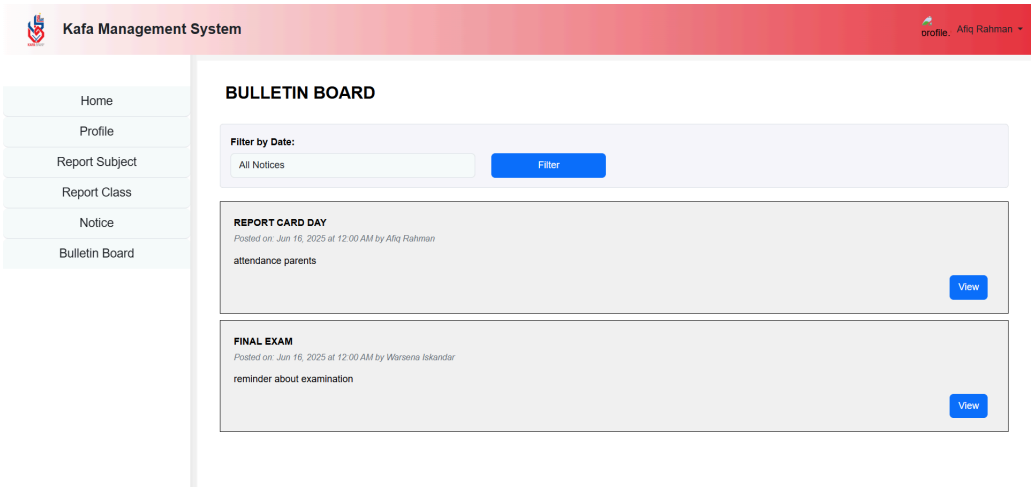


Figure 2.44 Bulletin Board interface after the implementation of the notice's name and date (MUIP Admin dashboard)

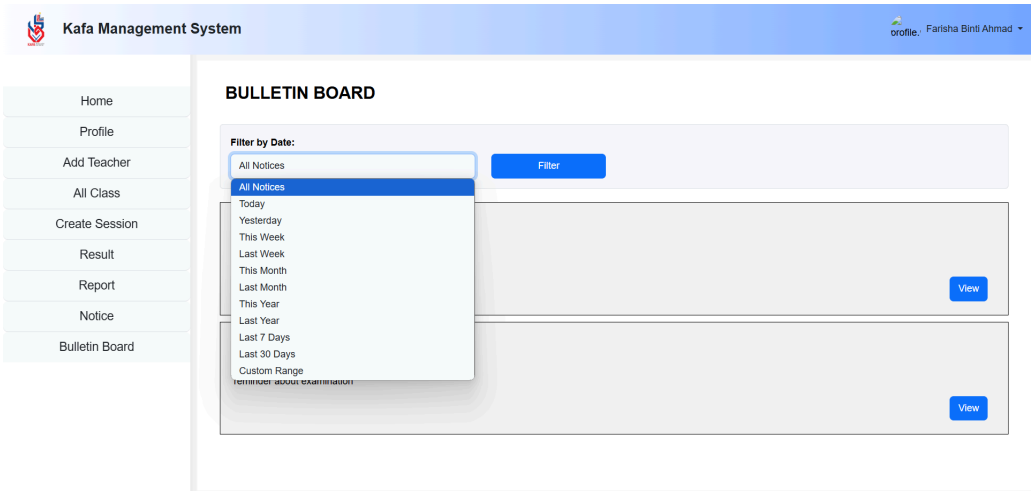


Figure 2.45 Bulletin Board interface after the implementation

of the filter button (KAFA Admin dashboard)

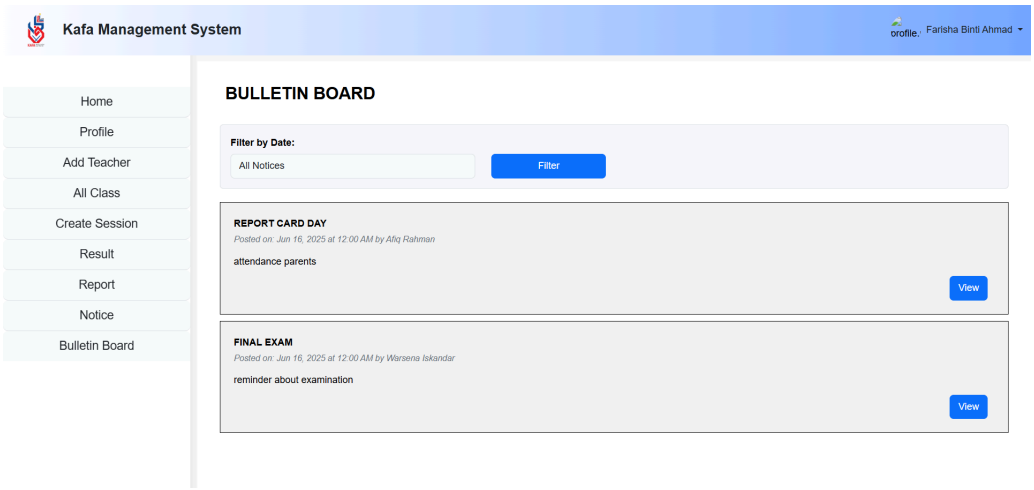


Figure 2.46 Bulletin Board interface after the implementation of the notice's name and date (KAFA Admin dashboard)

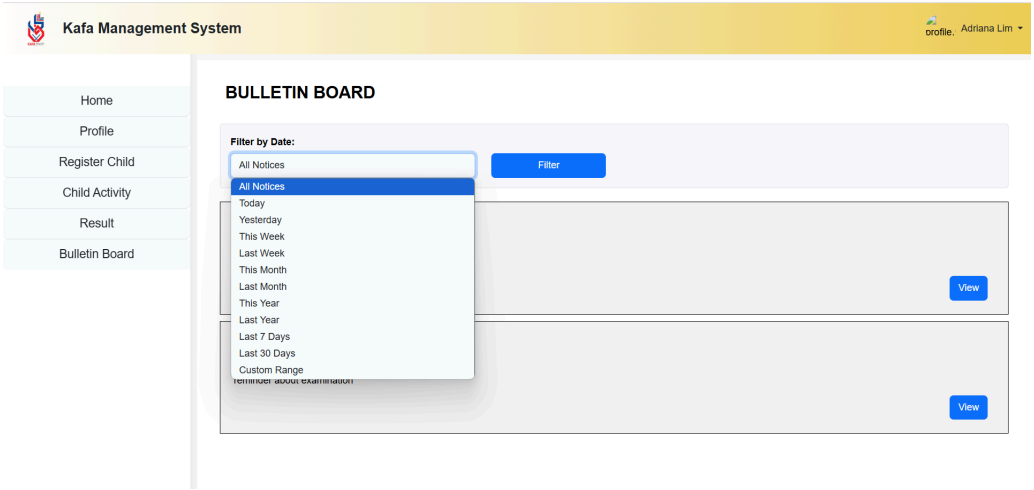


Figure 2.47 Bulletin Board interface after the implementation of the filter button (Parent dashboard)

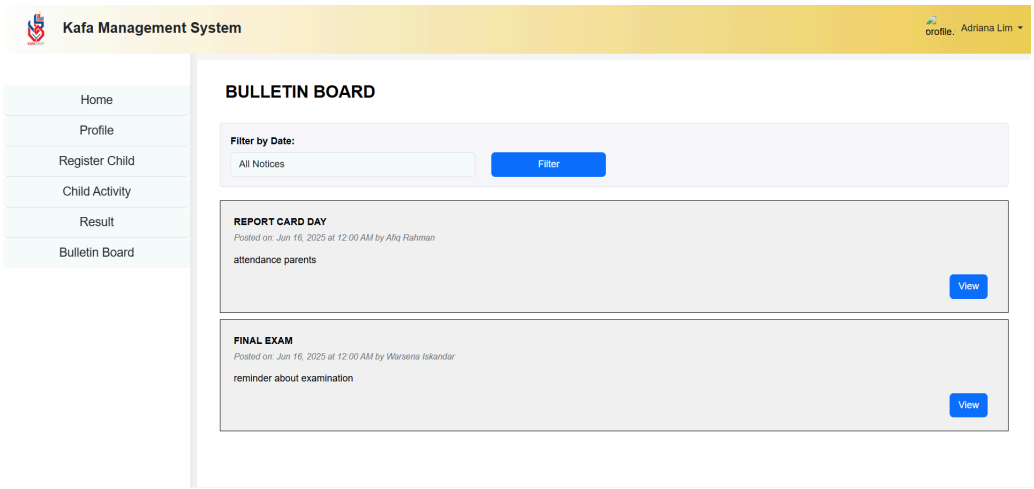


Figure 2.48 Bulletin Board interface after the implementation of the notice's name and date (Parent dashboard)

Type of Maintenance Chosen 2
Manage Bulletin: Addition of delete button for bulletin notices

Before Implementation:

```
96          @if($userRole->role_id == 4 || $userRole->role_id == 2)
97          <div class="col text-center d-flex align-items-center justify-content-center"
98              style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
99              {{ $notice->notice_status }}
100         </div>
101         @endif
102         @if($userRole->role_id == 2)
103         <div class="col text-center d-flex align-items-center justify-content-center"
104             style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
105             @if ($notice->notice_status == "Approved")
106             <button class="btn btn-warning btn-sm" disabled>
107                 <i class="bi bi-eye"></i>
108             </button>
109             @elseif ($notice->notice_status == "Rejected")
110             <button class="btn btn-warning btn-sm" disabled>
111                 <i class="bi bi-eye"></i>
112             </button>
113             @else
114             <a href="{{ route('formapproval', ['id' => $notice->id]) }}" class="btn btn-warning btn-sm">
115                 <i class="bi bi-eye"></i>
116             </a>
117             @endif
118               
119             <form action="{{ route('deletenotice', $notice->id) }}" method="POST"
120                 onsubmit="return confirm('Are you sure you want to delete this notice?');">
121                 @csrf
122                 @method('DELETE')
123                 <button type="submit" class="btn btn-danger btn-sm">
```

Figure 2.49 Notice page code before delete button implementation

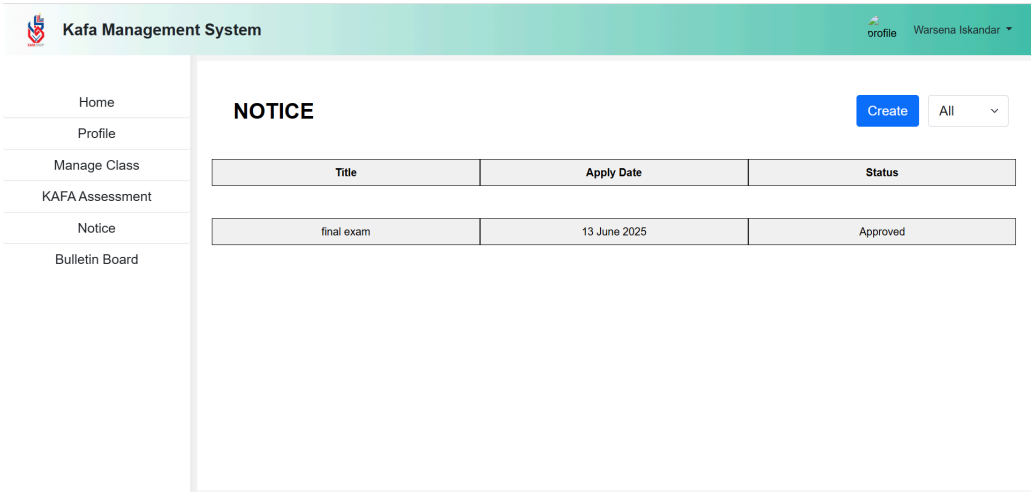


Figure 2.50 Notice page interface before the implementation of the delete button (Teacher dashboard)

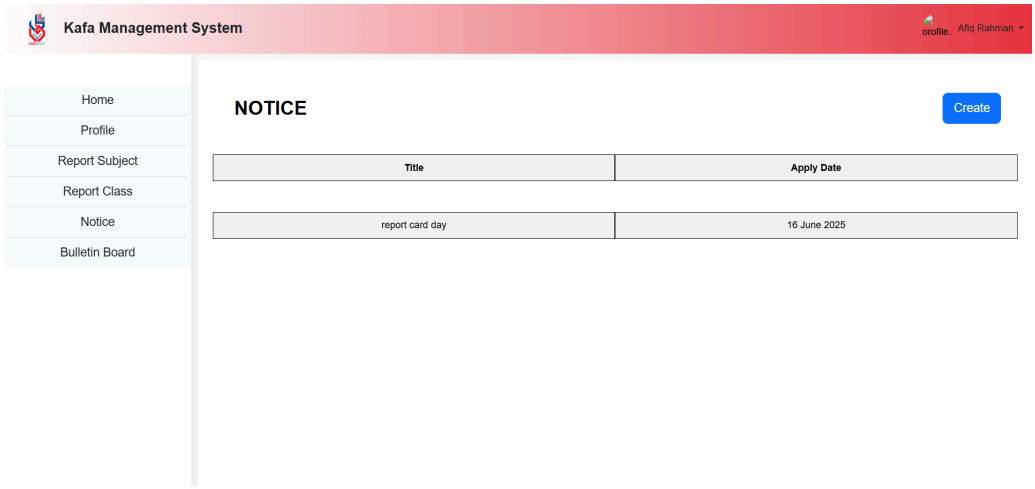


Figure 2.51 Notice page interface before the implementation of the delete button (MUIP Admin)

After Implementation :

```
89              @endif
90              <div class="col text-center d-flex align-items-center justify-content-center"
91                  style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
92                  {{ \Carbon\Carbon::parse($notice->notice_submission_date)->format('j F Y') }}
93              </div>
94              @if($userRole->role_id == 4 || $userRole->role_id == 2)
95              <div class="col text-center d-flex align-items-center justify-content-center"
96                  style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
97                  {{ $notice->notice_status }}
98              </div>
99              @endif
100             <div class="col text-center d-flex align-items-center justify-content-center"
101                 style="border: 1px solid black; height: 40px; background-color: #f2f2f2;">
102                 @if($userRole->role_id == 2)
103                 @if ($notice->notice_status == "Approved")
104                 <button class="btn btn-warning btn-sm" disabled>
105                     <i class="bi bi-eye"></i>
106                 </button>
107                 @elseif ($notice->notice_status == "Rejected")
108                 <button class="btn btn-warning btn-sm" disabled>
109                     <i class="bi bi-eye"></i>
110                 </button>
111                 @else
112                 <a href="{{ route('formapproval', ['id' => $notice->id]) }}" class="btn btn-warning btn-sm">
113                     <i class="bi bi-eye"></i>
114                 </a>
115                 @endif
116                   
```

Figure 2.52 Notice page code after delete button implementation

```
116                   
117                 @endif
118                 <form action="{{ route('deletenotice', $notice->id) }}" method="POST"
119                     onsubmit="return confirm('Are you sure you want to delete this notice?');">
120                     @csrf
121                     @method('DELETE')
122                     <button type="submit" class="btn btn-danger btn-sm">
123                         <i class="bi bi-trash"></i>
124                     </button>
```

Figure 2.53 Notice page code after delete button implementation

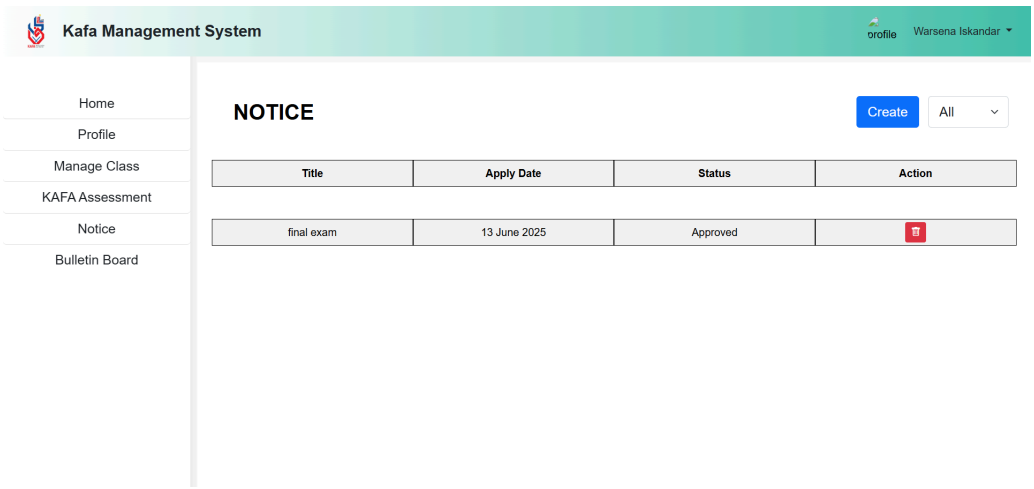


Figure 2.54 Notice page interface after the implementation of the delete button (Teacher dashboard)

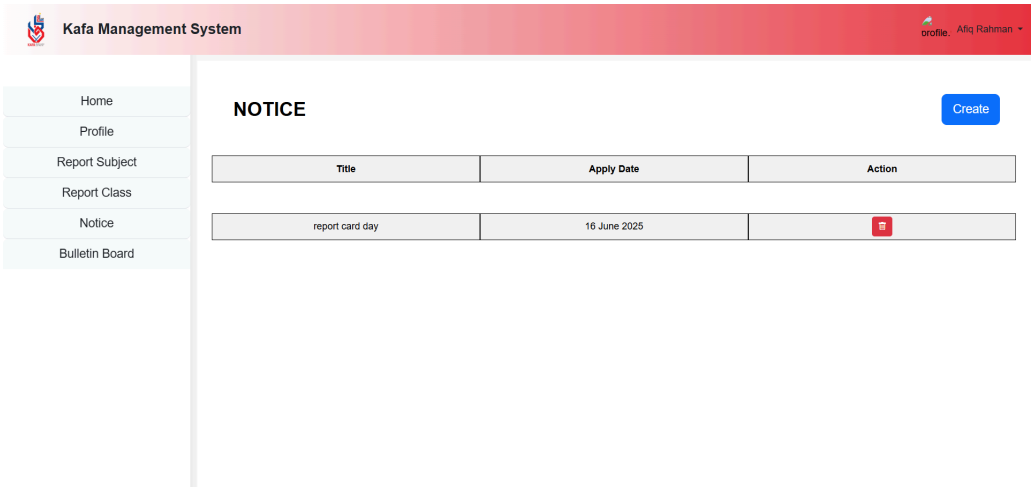


Figure 2.55 Notice page interface after the implementation of the delete button (MUIP Admin dashboard)

Type of Maintenance Chosen 3
Manage Bulletin: Addition of download button for bulletin notices

Before Implementation:

```
<!-- JavaScript for Custom Date Range Toggle -->
<script>
function toggleCustomDateRange() {
    const dateFilter = document.getElementById('dateFilter').value;
    const customDateRange = document.getElementById('customDateRange');

    if (dateFilter === 'custom') {
        customDateRange.style.display = 'block';
    } else {
        customDateRange.style.display = 'none';
        // Clear custom date inputs when not using custom filter
        document.getElementById('start_date').value = '';
        document.getElementById('end_date').value = '';
    }
}

// Auto-submit form when date filter changes (except for custom)
document.getElementById('dateFilter').addEventListener('change', function() {
    if (this.value !== 'custom' && this.value !== '') {
        document.getElementById('filterForm').submit();
    }
});
</script>

@endsection
```

Figure 2.56 KAFA Admin Bulletin Dashboard code before download button implementation

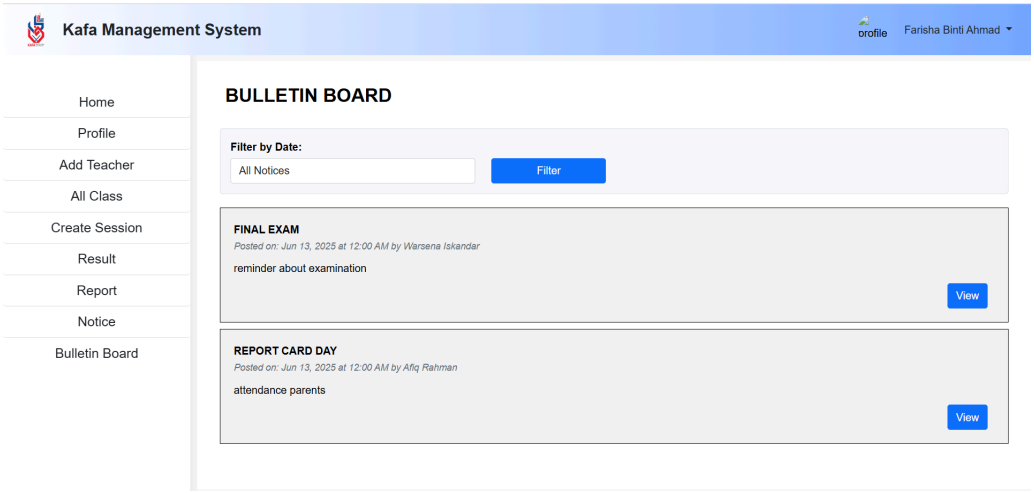


Figure 2.57 KAFA Admin Bulletin dashboard interface before the implementation of the download button

After Implementation

```
<!-- Load jsPDF library from CDN -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>

<!-- JavaScript for Custom Date Range Toggle and Download Function -->
<script>
function toggleCustomDateRange() {
    const dateFilter = document.getElementById('dateFilter').value;
    const customDateRange = document.getElementById('customDateRange');

    if (dateFilter === 'custom') {
        customDateRange.style.display = 'block';
    } else {
        customDateRange.style.display = 'none';
        // Clear custom date inputs when not using custom filter
        document.getElementById('start_date').value = '';
        document.getElementById('end_date').value = '';
    }
}
```

Figure 2.58 KAFA Admin Bulletin Dashboard code after download button inplementation

```
// Download notice as PDF function
function downloadNotice(noticeId, title, content, author, date) {
    // Initialize jsPDF
    const { jsPDF } = window.jspdf;
    const doc = new jsPDF();

    // Set up fonts and styling
    doc.setFont("helvetica", "bold");
    doc.setFontSize(18);

    // Add title
    doc.text("BULLETIN BOARD NOTICE", 20, 20);

    // Add a line separator
    doc.setLineWidth(0.5);
    doc.line(20, 25, 190, 25);

    // Notice title
    doc.setFontSize(16);
    doc.setFont("helvetica", "bold");
    const titleLines = doc.splitTextToSize(title.toUpperCase(), 170);
    let currentY = 40;
    doc.text(titleLines, 20, currentY);
    currentY += titleLines.length * 7;

    // Notice details
    doc.setFontSize(10);
    doc.setFont("helvetica", "normal");
    currentY += 10;
    doc.text(`Posted on: ${date}`, 20, currentY);
    currentY += 6;
```

Figure 2.59 KAFA Admin Bulletin Dashboard code after download button inplementation

```
                currentY = 20;
            }
            doc.text(contentLines[i], 20, currentY);
            currentY += 6;
        }

        // Add footer
        currentY += 20;
        if (currentY > 270) {
            doc.addPage();
            currentY = 20;
        }
        doc.setLineWidth(0.3);
        doc.line(20, currentY, 190, currentY);
        currentY += 10;
        doc.setFontSize(8);
        doc.setFont("helvetica", "italic");
        doc.text("Generated from Bulletin Board System", 20, currentY);
        doc.text(`Generated on: ${new Date().toLocaleString()}`, 20, currentY + 5);

        // Save the PDF
        const filename = `Notice_${noticeId}_${title.replace(/[^a-z0-9]/gi, '_').toLowerCase()}.pdf`;
        doc.save(filename);
    }
    </script>

    @endsection
```

Figure 2.60 KAFA Admin Bulletin Dashboard code after download button inplementation

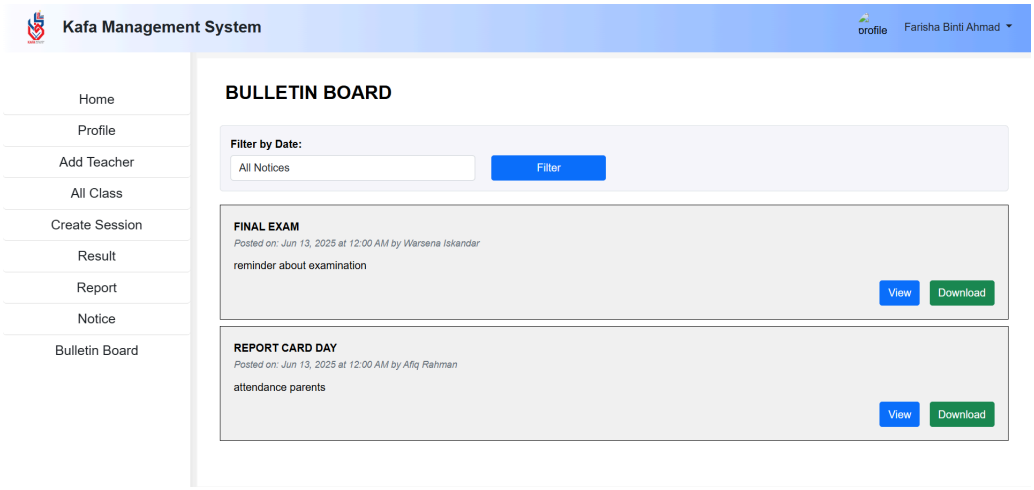


Figure 2.61 KAFA Admin Bulletin dashboard interface after the implementation of the download button

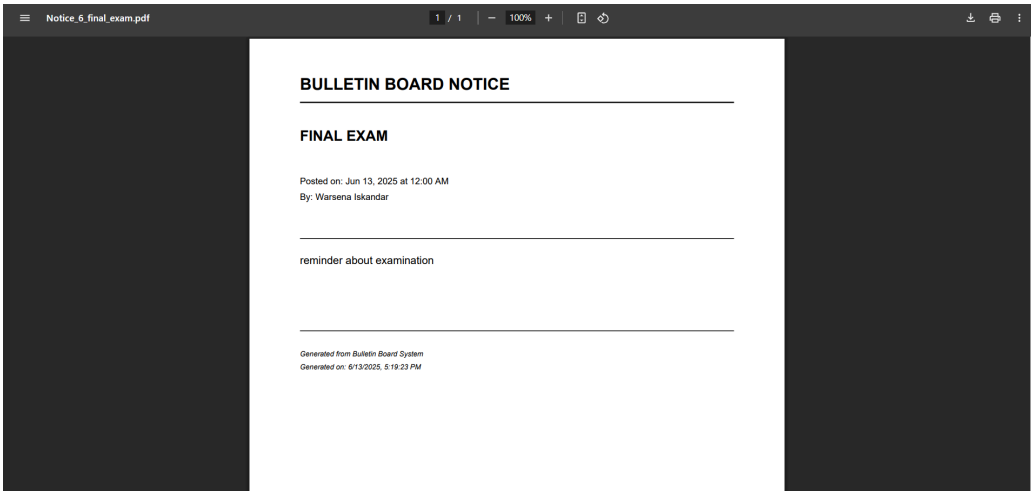
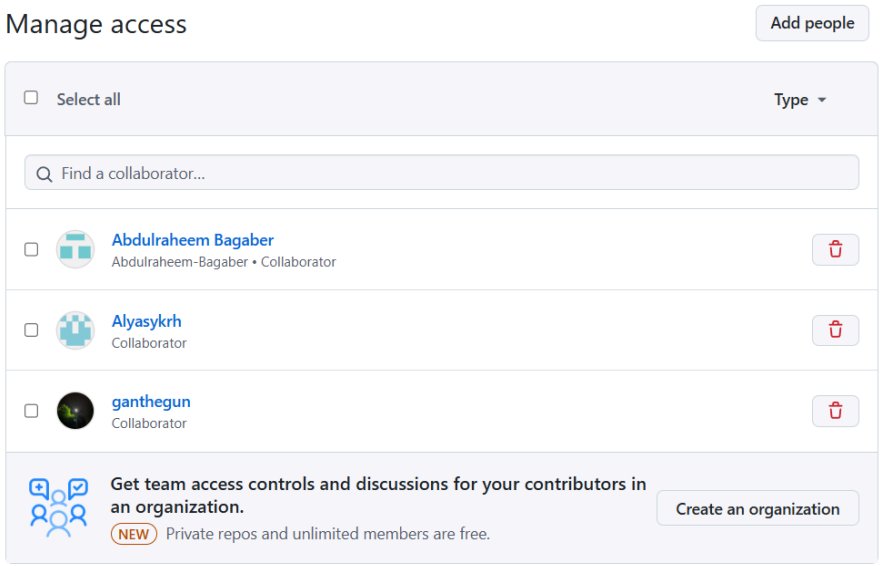


Figure 2.62 Downloaded PDF of Bulletin Notice

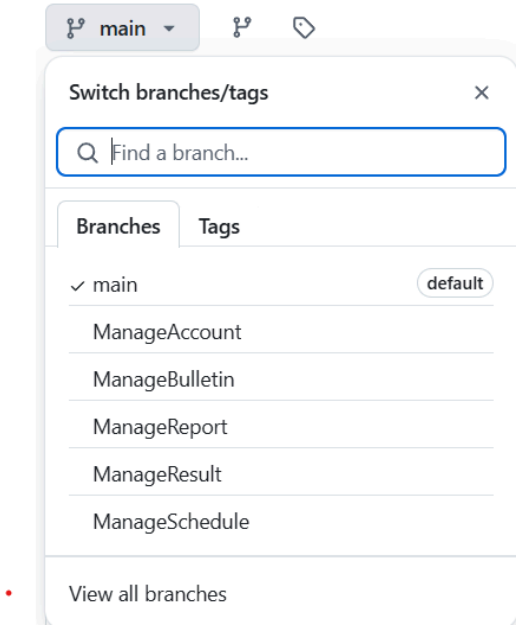
3.3.2. Members and role



Student Name	Student ID	Github ID	Role	Module
Umairah Shuhada Binti Ahmad	CB22090	umaishu03	Owner	Manage Result
Gan Jun Wei	CB22106	ganthegun	Collaborator	Manage Schedule
Bagaber AbdulRaheem Mohammed Sheikh	CB22016	Abdulraheem Bagaber	Collaborator	Manage Profile
Nur Alya Syakirah Binti Nasarudin	CB22141	Alyasykrh	Collaborator	Manage Bulletin

3.3.3. Build and test code on the preferred branch

For our project, we use branches to ensure it works smoothly when creating a change from the core source without disturbing modules or enhancements. The branches that are created such as ManageAccount, ManageBulletin, ManageReport, ManageResult and ManageSchedule. It is important to ensure that team members work without disturbing each other's work. The best branch is indeed the branch on which a team member is working.

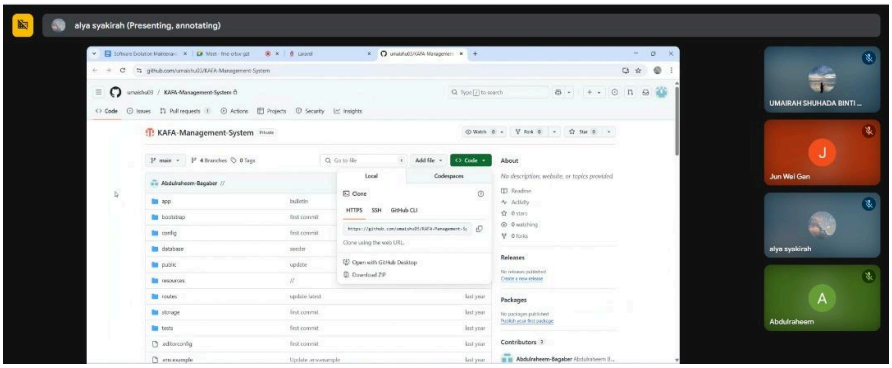


3.3.4. Minutes of Meeting

3.3.4.1. Meeting 1

Minutes of Meeting	
Date	9 May 2025
Meeting Platform	Face-To-Face
Attendees	1. Umairah Shuhada Binti Ahmad 2. Gan Jun Wei 3. Bagaber AbdulRaheem Mohammed Sheikh 4. Nur Alya Syakirah Binti Nasarudin
Progress	Selected the existing system to use in this project and assigned module by each team member.
Attendance	All
Absence	None
Problem	None

3.3.4.2. Meeting 2

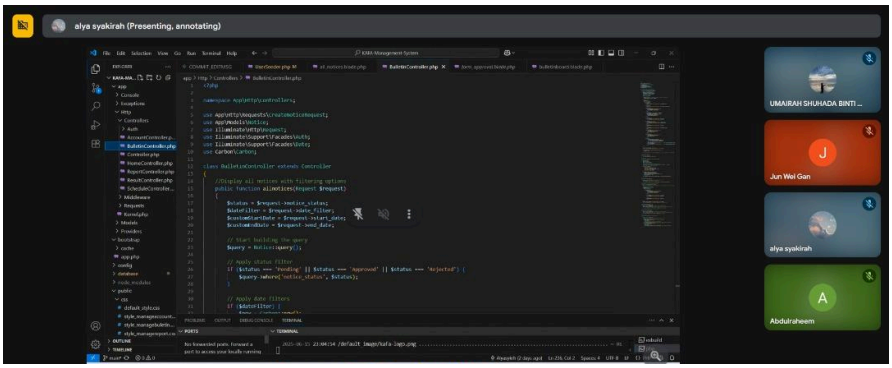


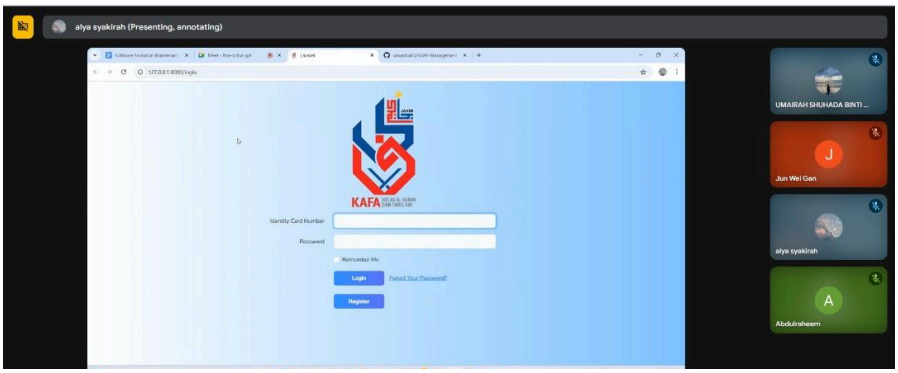
Minutes of Meeting	
Date	12 May 2025
Meeting Platform	Online - Google Meet
Attendees	1. Umairah Shuhada Binti Ahmad 2. Gan Jun Wei 3. Bagaber AbdulRaheem Mohammed Sheikh 4. Nur Alya Syakirah Binti Nasarudin
Progress	The GitHub repository has been successfully created under the team account.
Attendance	All
Absence	None
Problem	None

3.3.4.3. Meeting 3

Minutes of Meeting	
Date	19 May 2025
Meeting Platform	Face-To-Face
Attendees	1. Pn. Ku Saimah Binti Ku Ibrahim 2. Umairah Shuhada Binti Ahmad 3. Gan Jun Wei 4. Bagaber AbdulRaheem Mohammed Sheikh 5. Nur Alya Syakirah Binti Nasarudin
Progress	Presented the idea for enhancement or corrective for the selected existing system and took the feedback.
Attendance	All
Absence	None
Problem	None

3.3.4.4. Meeting 4





Minutes of Meeting	
Date	13 June 2025
Meeting Platform	Online - Google Meet
Attendees	1. Umairah Shuhada Binti Ahmad 2. Gan Jun Wei 3. Bagaber AbdulRaheem Mohammed Sheikh 4. Nur Alya Syakirah Binti Nasarudin
Progress	Discussed the progress for each part.
Attendance	All
Absence	None
Problem	None

4. References

1. *Fiix. (2025, May 12). What is Corrective Maintenance? | Fiix.* *https://fiixsoftware.com/glossary/corrective-maintenance/*
2. *GeeksforGeeks. (2024, November 25). How to Embed PDF file using HTML ? GeeksforGeeks.* *https://www.geeksforgeeks.org/html/how-to-embed-pdf-file-using-html/*
3. *Voss, E. (2024, October 18). The evolution of the maintenance industry. Fiix.* *https://fiixsoftware.com/blog/evolution-maintenance-practice/*
4. *Sensemore. (2024, May 2). The evolution of maintenance strategies. Sensemore.* *https://sensemore.io/the-evolution-of-maintenance-strategies/?srsltid=AfmBOopA0dCaXCaKIggfwvQfmxUEUBAbZHjTS0ICmvvQF-sLFzUb16I2*
5. *Ali, A. (2024, January 18). How to Create Registration Form in HTML with Database. Medium.* *https://arhaanali.medium.com/how-to-create-registration-form-in-html-with-database-883947f87b27*
6. *Byrnes, J. (2023, June 16). How to manage a Decision-Enhancement System Implementation - CFO Leadership. CFO Leadership.* *https://cfoleadership.com/how-to-manage-a-decision-enhancement-system-implementation/*

7. *Nisbet, N., Zhang, Z., Ma, L., Chen, W., & Çıdık, M. S. (2023). Semantic correction, enrichment and enhancement of social and transport infrastructure BIM models. Advanced Engineering Informatics, 59, 102290. https://doi.org/10.1016/j.aei.2023.102290*
8. *Clawson, E. (2022, January 5). Adding a delete button in 4 easy steps - E Clawson - Medium. Medium. https://medium.com/@emclawson1/adding-a-delete-button-in-4-easy-steps-85b06f53da6b*
9. *W3Schools.com. (n.d.-b). https://www.w3schools.com/howto/howto_js_filter_elements.asp*
10. *Anderson, C. (n.d.). What is productivity enhancement solutions? – Focuskeeper Glossary. https://focuskeeper.co/glossary/what-is-productivity-enhancement-solutions*
11. *Enhancing activity guides*. (n.d.). https://docs.oracle.com/cd/F30998_01/pt858pbr2/eng/pt/tprt/task_EnhancingActivityGuides.html
12. *Types of Maintenance: The complete guide*. (n.d.). TRACTIAN. https://tractian.com/en/blog/types-of-maintenance