APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 1 (2025) 1036-1048
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Program Management System for Pusat Sukan Universiti (PSU)

Mohammad Ridhwan Suriansah[1], Mohd Zanes Sahid[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: zanes@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.057

Article Info

Received: 7 January 2025
Accepted: 19 June 2025
Available online: 30 June 2025

Keywords

Program Management System, Automation, Proposal Approval

Abstract

Within a year, as a university sports organisation, Pusat Sukan Universiti (PSU) can organize tens or hundreds of activities or events, such as seminars, workshops, exhibitions, sports, and others. This project aims to develop an online platform to streamline and automate the PSU's program approval process. The system will be designed to replace the current manual paperwork with an efficient, web-based solution accessible via the PSU website. Applicants can submit proposals and track their status online, while reviewers will have dedicated dashboards for evaluation. Key findings suggest that the system will significantly reduce approval times and improve transparency, accuracy, and user accessibility. The study concludes with recommendations for further enhancements to optimise the system's functionality.

1. Introduction

In managing university sports and recreational activities, it has become clear that there is an increasing need for processes to be made more efficient and streamlined. Pusat Sukan Universiti (PSU), established in 2011, serves as the central coordinator for sports and recreational programs for UTHM students and staff [1]. However, the current manual method of seeking approval for program ideas at PSU has been identified as time-consuming and prone to inefficiencies. The reliance on paperwork for submission and review processes has led to delays, extended approval periods, and increased administrative workloads for PSU administrators and applicants.

Recognising the challenges posed by the manual system, the project aims to introduce a web-based software solution to revolutionise the program approval process at PSU. By transitioning from traditional paper-based submissions to an online platform, the project seeks to simplify and automate program proposal submission, review, and approval. This shift towards digitalisation aligns with PSU's long-term goal of modernising its operations and enhancing service delivery to its users.

The primary objective of this project is to analyse, design, and implement a program management system for PSU that leverages modern technologies and best practices in web development and database management. By adopting a model-view-controller approach, the system aims to provide a user-friendly environment for UTHM event coordinators, club representatives, and students to submit and manage program applications seamlessly. Through the utilisation of open-source software, the system will not only streamline the approval process but also promote transparency, accuracy, and accessibility in program management at PSU.

The expected outcome of this project is to significantly reduce the time required for program proposal approvals, enhance program management efficiency, and improve the overall user experience for both applicants and reviewers. By automating document submission, tracking progress, generating reports, and managing

program information, the web-based system will contribute to the effectiveness and productivity of PSU's sports and recreational activities.

This article consists of six sections covering various aspects of the project. It begins with the background, followed by an analysis of related work. The methodology section explains the project's approach and design process. Implementation details and testing results are discussed in the fourth section. The project conclusion highlights contributions, limitations, and suggestions for future improvements. The final section expresses gratitude and acknowledgement to those who supported the project.

2. Related Work

This section will discuss the related concept of the Web-Based Program Management System for Pusat Sukan Universiti (PSU) and compare it with a similar system.

2.1 Management System

A management system is how a company organizes its operations to achieve specific goals, such as quality, efficiency, environmental performance, and workplace safety [2]. It involves coordinating activities to reach these objectives and is considered a key factor of production alongside equipment, materials, and finances.

A system consists of interconnected components working together to achieve a common goal by processing inputs into outputs [3]. In essence, a management system facilitates the collection, storage, and transfer of information to support business operations.

Examples include learning management systems, quality management systems, and integrated management systems [4]. To improve management efficiency, companies are transitioning from outdated manual methods to advanced digital systems.

2.2 Web-Based System

A web-based system uses the Internet and web technologies to provide information or services to users. These systems fall into four main categories: intranets for internal operations, web presences for marketing, e-commerce systems for consumer engagement, and extranets for connecting internal and external systems. They integrate models and data to address unstructured problems with significant user involvement through a user-friendly interface.

Development methods include using software frameworks or building applications from scratch. Frameworks are collections of programs that simplify coding and maintenance, often using languages like PHP, JavaScript, or Python. While they can streamline the development process, they may hinder users from fully mastering the programming language itself.

Building a system from scratch allows for greater control over hosting and enhances security against hacking, while also deepening one’s understanding of the code.

2.3 Pusat Sukan Universiti Manual System

The current program management system is managed manually by submitting a formal application letter for the program. Firstly, the paperwork must be submitted in person to the PSU. Then, the paperwork will be checked by the PSU to ensure that every component has fulfilled the requirements. The process will continue until students meet the requirements of the program application.

As the coordinator, the PSU is responsible for approving or declining any event proposals. In the meantime, Pejabat Bendahari UTHM (PBU) is responsible for verifying the program's budget. If the budget exceeds the amount set by the PBU, the organizer needs to adjust their budget accordingly. If any mistakes or parts do not meet the specified requirements and constraints, the organizer must revise their paperwork. But, if the program has been approved by PSU, PBU, and superiors, the events can proceed.

2.4 Study of Existing System

A study has been conducted on three existing systems in the market. This study is conducted to analyse and identify the advantages and disadvantages of the existing system to use as references when developing the system. The three existing related systems that have been chosen are UM Event System, Cvent, and Whova.

Table 1 *Comparison of existing systems with the proposed system*

Feature / System	UM Event System **[5]**	Cvent **[6]**	Whova **[7]**	Proposed System
Login	√	√	√	√
Register	√	√	√	√
Update User Profile	√	√	X	√

Submit Program Application	√	X	X	√

Table 1: *(cont)*

Manage Application	X	√	√	√
Track Application Status	√	√	√	√
Manage Report	X	X	X	√

The proposed system provides a customized solution designed specifically for the needs of the PSU. It addresses various aspects, including detailed program application submissions, role management, and report management. In contrast, the UM Event System, Cvent, and Whova primarily focus on general event logistics and promotion, and they lack some essential features necessary for managing student clubs within an educational institution.

3. Methodology

This section provides an overview of the methodology and analysis employed in this project. The development of the web-based program management system follows the evolutionary prototyping model. It contains requirement analysis, Unified Modelling Language diagrams, database design, and general system architecture to ensure a thorough understanding of the system's structure and functionality.

3.1 System Development

The software process model used in developing the system is the Evolutionary Prototyping Model. This model involves creating prototypes initially and then incorporating feedback to enhance the system until the final product is developed. The Evolutionary Prototyping Model allows for customisation and alignment with project submission deadlines while maintaining the advantages of prototyping [8].

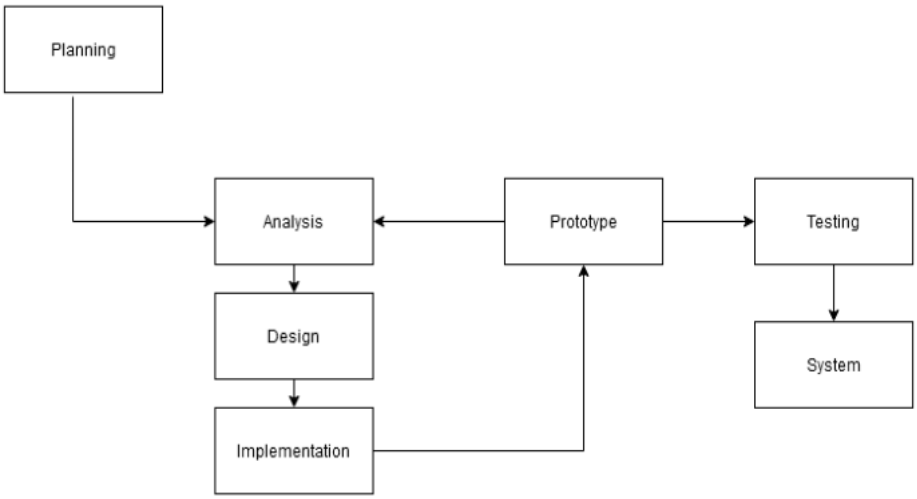

Fig. 1 *Evolutionary Prototype Model Flow*

The model involves six main phases, as shown in Fig. 1: planning, analysis, design, implementation, prototyping, and testing. In the planning phase, the sequence of activities is determined, issues are identified, objectives are set, and a proposal is presented outlining the significance and expected results of the project. The analysis phase involves examining issues identified in planning, conducting stakeholder interviews, and documenting system requirements using various diagrams and definitions. The design phase focuses on developing user interfaces and designing the system database, incorporating stakeholder feedback. Implementation involves converting the design into source code, implementing graphical interfaces, and creating prototypes in each iteration. The prototyping phase includes developing prototypes based on stakeholder feedback to enhance system functionalities. Finally, the testing phase ensures the system meets all requirements through rigorous testing and verification. The activities and output of each phase from the methodology are presented in Table 2.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1036-1048

Table 2 *System Development Activities and their tasks*

Phase	Task/Activities	Techniques	Tool
Planning	• Identify and map the existing problems • Determine the topic and propose the title of the project • Determine the scope, problem statement, and project significance • Develop the timeline in Gantt chart form	• Project proposal • List of scope, problem statement, objective, and project significance • Gantt Chart • Interview	• Microsoft Word • WhatsApp
Analysis	• Determine the user requirements/stakeholders • Collect data from Pusat Sukan UTHM • Conduct a study on the existing system • Determine the system requirement • Determine the list of hardware and software requirement • Draw the use case diagram, activity diagram, sequence diagram, and class diagram	• Figure of swim lane diagram (as-is model) • List of potential hardware and software specifications • Use case diagram, activity diagram, sequence diagram, and class diagram • Requirement Definition	• Microsoft Word • Visual-Paradigm
Design	• Design all the modules of the user interface of the proposed system • Design the database of the proposed system	• Initial user interface of the web-based system • Second user interface • Final user interface • Database scheme	• Microsoft Word • Figma
Implementation	• Develop 3 prototypes • Code the program using PHP • Connect MySQL to the system	• Prototype 1 • Prototype 2 • Prototype 3	• Visual Studio Code • XAMPP
Prototype	• Develop Prototype 1 of Iteration 1 • Develop Prototype 2 of Iteration 2 • Develop Prototype 3 of Iteration 3	• Prototype 1 (interfaces of all use cases) • Prototype 2 (connection between interfaces and database) • Prototype 3 (navigation between interfaces)	• Visual Studio Code • XAMPP
Testing	• Develop system • Create the test cases list • Testing the system function • Fixing bugs and errors	• System • Test cases • Requirement Traceability Matric (Test Cases vs. Requirements)	• Visual Studio Code • Microsoft Word

4. Analysis and Design

This section focuses on the analysis and design process of the system, including use cases, UML diagrams, and system interfaces. It covers the methodologies to develop high-quality information systems that integrate IT, people, and data to meet business requirements. Additionally, it highlights using use case diagrams to illustrate primary system functions and the interaction between modules and system users [9].

4.1 Use Case Diagram

Figure 2 shows the class diagram, which provides an overview of the behavioural processes used in system design and represents the interaction between modules and system users. Seven functional modules involve three users: student, organiser, and coordinator. The use cases include Login, Register, Checklist, Submit Application, Review and Approve Application, Track Progress, and Generate Report. Each use case represents a specific functionality or feature of the system, showcasing how users interact with the system to achieve their goals.

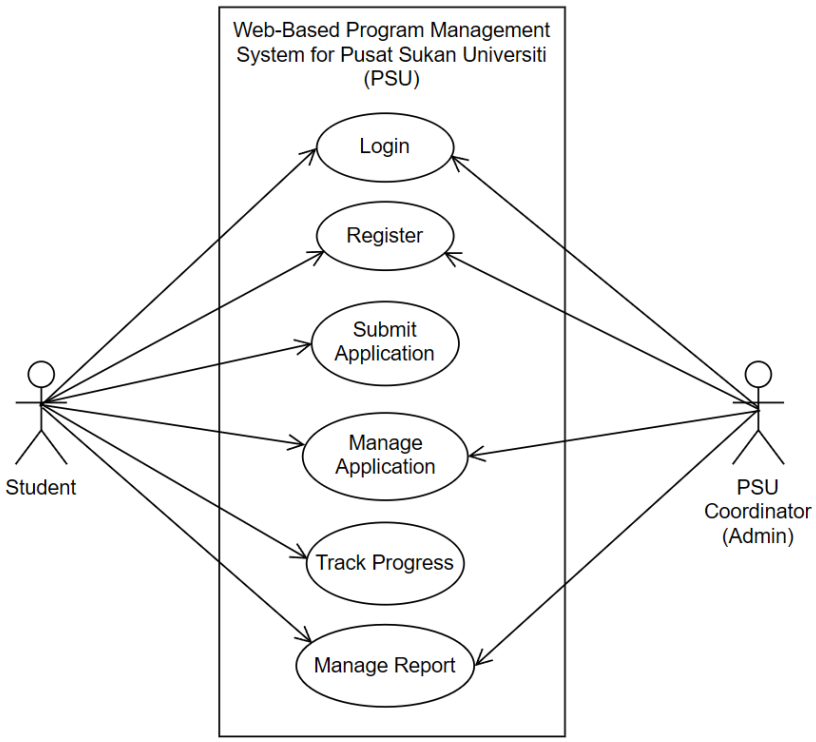

Fig. 2 *Use case diagram*

4.2 Requirement Traceability Matrix

The Requirement Traceability Matrix links project requirements back to their origins and monitors their fulfilment throughout the project lifecycle. Table 3 shows the requirement traceability matrix of the Web-Based Program Management System.

Table 3 *Requirement Traceability Matrix*

Allocation	Description
REQ_100	**Login**
REQ_101	The system should display the login page.
REQ_102	The system should authenticate valid usernames and passwords.
REQ_103	The system should redirect users to the homepage after successful login.
REQ_200	**Register**
REQ_201	The system should display the register page.
REQ_202	The system should accept input for registration details.
REQ_203	The system should verify the password format.
REQ_204	The system should display a success message for the valid information.

Table 3: *(cont)*

REQ_205	The system should display an error message for the invalid information.
REQ_206	The system should redirect users to the homepage after successful registration.
REQ_300	**Submit Application**
REQ_301	The system should allow users to input relevant data in form fields.
REQ_302	The system should allow users to attach paperwork.
REQ_303	The system should validate the application data for completeness.
REQ_304	The system should provide a success message upon successful application submission.
REQ_400	**Manage Application**
REQ_401	The system should display the submitted application list.
REQ_402	The system should present detailed content of the selected program application for reviewing.
REQ_403	The system should allow admins to approve an application, updating its status to "Approved".
REQ_404	The system should allow admins to reject an application, updating its status to "Rejected", and add feedback.
REQ_500	**Track Progress**
REQ_501	The system should display the user's submitted applications.
REQ_502	The system should present the status of the selected program application and show comments.
REQ_503	The system should enable users to update their submitted applications based on the feedback provided by the admin.
REQ_600	**Manage Report**
REQ_601	The system should display the report page.
REQ_602	The system should allow students to upload report files and input links for the program's images.

4.3 General System Architecture

The architectural design of a system involves organizing its components and determining how they interact with one another. The Web-Based Program Management System for PSU uses the MVC (Model-View-Controller) architecture, which divides the application into three key components: the model, the view, and the controller.

The model is responsible for data management, the view displays the data to the user, and the controller manages user input and interactions. Specifically, the model retrieves and stores data, the view presents this data to the user, and the controller processes user input and updates the display accordingly. The relationship among these three components can be visualized in Fig. 3.

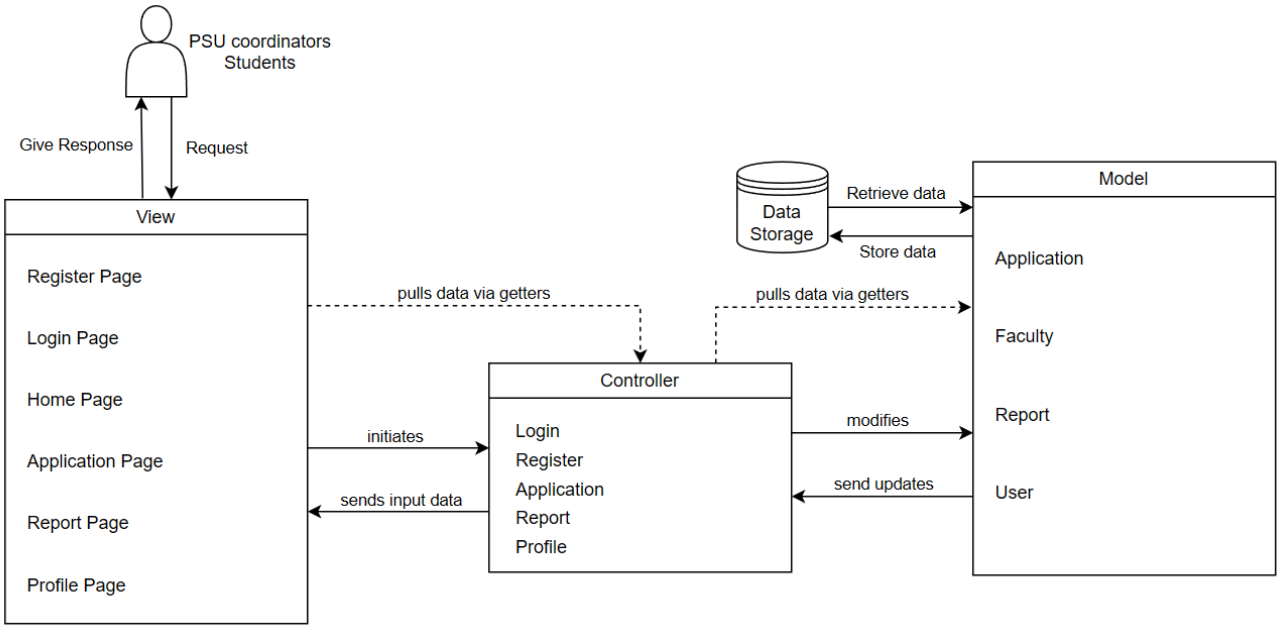

Fig. 3 *System Design Diagram for the system*

4.4 Class Diagram

Fig. 4 shows the class diagram for the system. A class diagram is a fundamental component in object-oriented modelling, illustrating the different objects in a system along with their attributes, operations, and relationships. It helps developers and stakeholders understand the overall structure of the application by depicting the classes and their interactions [10].

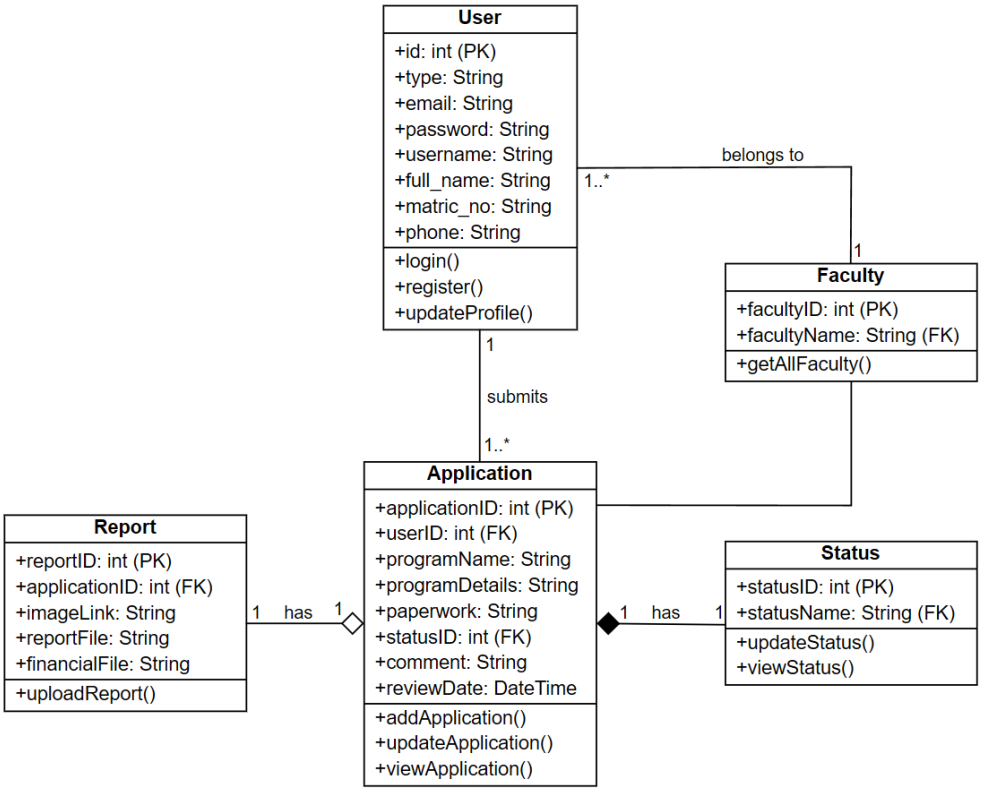

Fig. 4 *Class diagram of the proposed system*

4.5 Interface Design

Interface design or prototype design consists of interfaces that the viewer will use. Prototype design is important as the interface helps visualise the actual system to the customer and developer. In Fig. 5, the login interface is displayed. This interface is where the user is prompted to input their email address and password to gain access to the system. In Fig. 6, the register interface is depicted. This interface enables new users to set up an account within the system by supplying the required information, such as their full name, password, and email address. In Fig. 7, the Submit Application interface is depicted. This interface serves as a platform for students to input all the necessary information and upload the required documents for the submission of their applications.

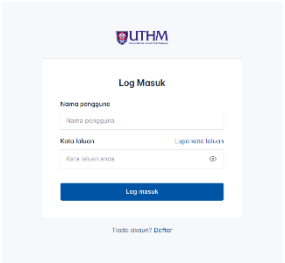

Fig. 5 *Login Interface*

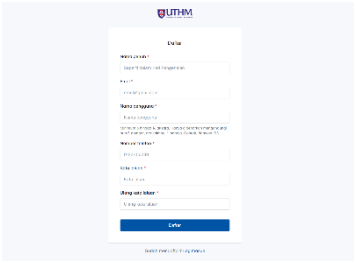

Fig. 6 *Register Interface*

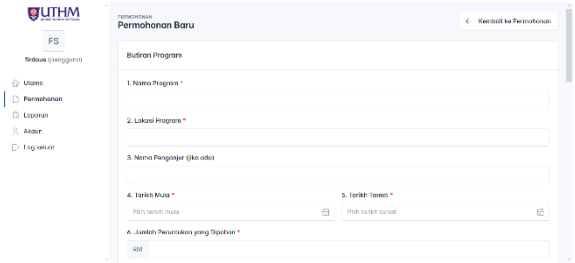

Fig. 7 *Submit Application interface*

Fig. 8 shows the user interface for the Manage Application process, allowing administrators to evaluate and decide on student applications based on set criteria. Fig. 9 presents the Track Progress interface, which provides students and coordinators with a visual overview of submitted applications, highlighting completed and pending statuses for easier management. Fig. 10 displays the Manage Report interface, enabling users to attach program and financial reports and input links for program images.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1036-1048

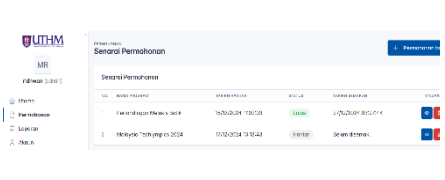

Fig. 8 *Manage Application Interface*

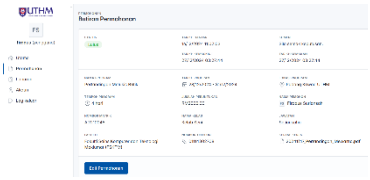

Fig. 9 *Track Progress Interface*

Fig. 10 *Manage Report Interface*

4.6 Design

The schema table, as illustrated in the class diagram (see **Fig. 4**), contains essential attributes and relationships that enhance the overall functionality and coherence of the system's data architecture. Below is the database schema derived from the class diagram:

i. Users (id, type, email, password, username, full_name, matric_no, phone)
ii. Application (applicationID, userID, programName, programDetails, paperwork, statusID, comment, reviewDate)
iii. Status (statusID, statusName)
iv. Faculty (facultyID, facultyName)
v. Report (reportID, applicationID, imageLink, reportFile, financialFile)

5. Result and Discussion

This section will cover two main components: the implementation phase and the testing phase. During the implementation phase, we will present the developed interface and code segments. The testing phase will be split into two parts: system testing and user acceptance testing.

5.1 Implementation

The system is primarily developed as a website using CodeIgniter, which helps keep the project organized and easy to maintain. It utilizes HTML, CSS, JavaScript, and PHP. For backend tasks, CodeIgniter's features securely manage user logins and handle the database through phpMyAdmin. The code is mainly written and edited in Visual Studio Code. This setup facilitates future system growth since CodeIgniter is designed to integrate well with other tools and can accommodate more complex tasks if needed.

The Login module allows users to log in and access the system. Fig. 11 shows the login interface for the user. The user needs to enter a username and password into the input fields. The Register module allows users to sign up for an account to access the system. The Register interface is shown in Fig. 12. The user must sign into the system by inserting full name, email address, username, phone number, and password.

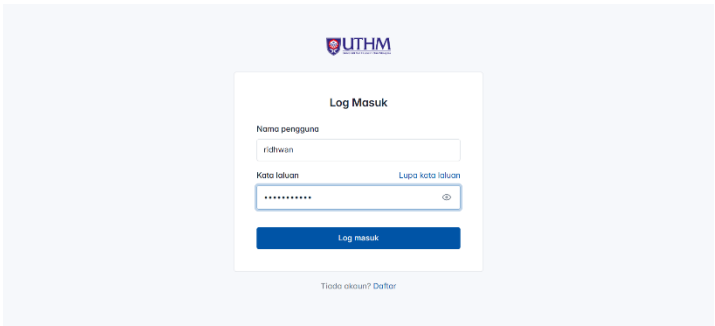

Fig. 11 *Login Interface*

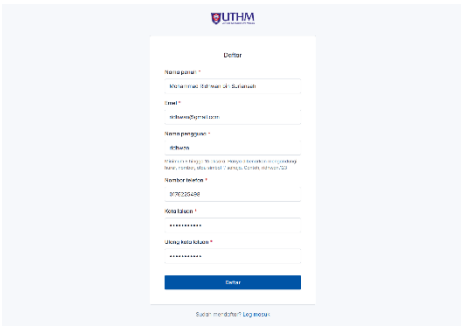

Fig. 12 *Register Interface*

Fig. 13 shows the code segment for login. The user can log into the system if the username and password are valid. If they are not valid, the system will display an error message indicating that the username or password is incorrect. Fig. 14 shows the code segment for the register. After successfully inserting all data, the system will validate the user inputs. The user will be redirected to the profile page if there are no errors.

```php
public function loginAction(): RedirectResponse
{
        $rules = $this->getValidationRules();
        if (! $this->validateData($this->request->getPost(), $rules, [], config('Auth')->DBGroup)) {
            return redirect()->back()->withInput()->with('errors', $this->validator->getErrors());
        }
        $credentials             = $this->request->getPost(setting('Auth.validFields')) ?? [];
        $credentials             = array_filter($credentials);
        $credentials['password'] = $this->request->getPost('password');
        $remember                = (bool) $this->request->getPost('remember');
        $authenticator = auth('session')->getAuthenticator();
        $result = $authenticator->remember($remember)->attempt($credentials);
```

Fig. 13 *Login Code Segment*

```php
public function registerAction(): RedirectResponse
    {
        if (auth()->loggedIn()) {
            return redirect()->to(config('Auth')->registerRedirect());
        }
        if (! setting('Auth.allowRegistration')) {
            return redirect()->back()->withInput()
                ->with('error', lang('Auth.registerDisabled'));
        }
        $users = $this->getUserProvider();
        $rules = $this->getValidationRules();
        if (! $this->validateData($this->request->getPost(), $rules, [], config('Auth')->DBGroup)) {
            return redirect()->back()->withInput()->with('errors', $this->validator->getErrors());
        }
        $allowedPostFields = array_keys($rules);
        $user              = $this->getUserEntity();
        $user->fill($this->request->getPost($allowedPostFields));
```

Fig. 14 *Register Code Segment*

The Submit Application module allows students to submit their program application. Fig. 15 shows the Submit Application interface. The Manage Application module enables administrators and PSU coordinators to review and approve student applications. Fig. 16 shows the Manage Application interface.

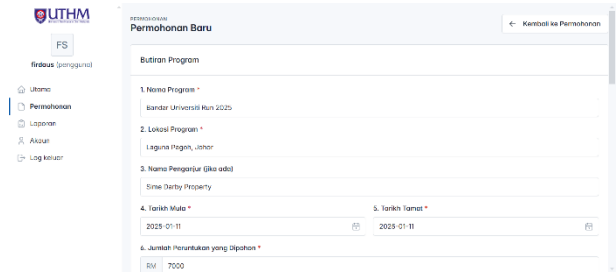

Fig. 15 *Submit Application Interface*

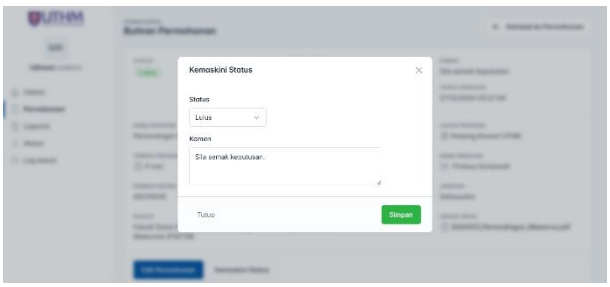

Fig. 16 *Manage Application Interface*

Fig. 17 shows the code segment for the Submit Application. To apply, students must be logged in to their accounts. They can then access the application page and select the “Add Application” option. The system will display a form for the students to fill in details and attach the paperwork for the program. A success message will appear if the details are valid, and the system will redirect to the application list. If the details are invalid, an error message is shown. Fig. 18 shows the code segment for the Manage Application. To begin the review process, there must be at least one application submitted for evaluation. The system will display a list of submitted applications, allowing administrators to select any application to view its details. After reviewing an application, the administrators can update its status to "Approved" or "Rejected." If an application is rejected, the administrators can provide feedback for students to view.

```php
public function addPermohonan()
    {
        if (!$this->validate($this->permohonanValidationRules)) {
            $fakulti = $this->fakultiModel->getAllFakulti();

            $data = [
                'ctx' => 'permohonan',
                'fakulti' => $fakulti,
                'pretitle' => 'Permohonan',
                'title' => 'Permohonan Baru',
                'validation' => $this->validator,
                'oldInput' => $this->request->getVar()
            ];
            return view('permohonan/tambah-permohonan', $data);
        }
```

Fig. 17 *Submit Application Code Segment*

```php
public function updateStatus()
    {
        $status_id = $this->request->getPost('status_id');
        $id_permohonan = $this->request->getPost('id_permohonan');
        $status_id = $this->request->getPost('status_id');
        $komen = $this->request->getPost('komen');
        $reviewedDate = $this->request->getPost('reviewed_at');

        $data = [
            'status_id'   => $status_id,
            'komen'       => $komen,
            'reviewed_at' => $reviewedDate,
        ];

        $updated = $this->permohonanModel->update(['id_permohonan' => $id_permohonan], $data);
```

Fig. 18 *Manage Application Code Segment*

The Track Progress module enables students to monitor the status of their submitted applications. Fig. 19 shows the Track Progress interface. The Manage Report module allows students to upload their program report, financial report, and an image link. Fig. 20 shows the Manage Report interface.

Fig. 21 shows the code segment for the Track Progress. The system will display a list of the user's submitted applications. The user can then select an application to view its details. The system will show the application status, and any feedback received. Fig. 22 shows the code segment for the Manage Report. Administrators can review these submissions. The user interface features a report submission form where students enter their Program Report File, Financial Report File, and Image Link (URL). After clicking "Submit Report," the system validates the inputs and files. If validation fails, an error message is displayed; if successful, the files and links are saved in the database, confirming the report submission.

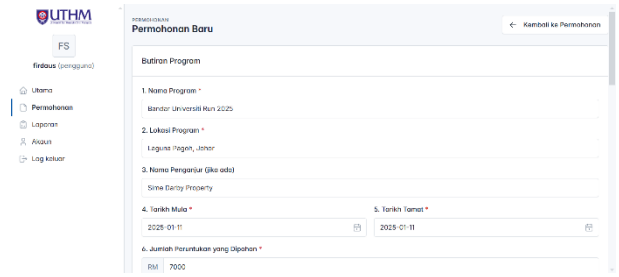

Fig. 19 *Track Progress Interface*

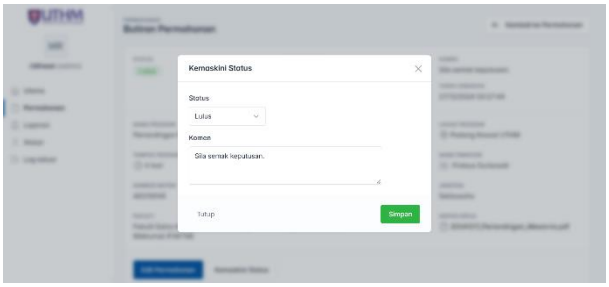

Fig. 20 *Manage Report Interface*

```
public function detail($id)
    {
        $status = $this->permohonanModel->getAllStatus();

        $result = $this->permohonanModel->getPermohonanById($id);

        if (!$result) {
            throw new \CodeIgniter\Exceptions\PageNotFoundException('Permohonan tidak
wujud.');
        }

        if (!auth()->user()->inGroup('superadmin', 'admin') && $result['id_pemohon'] !==
(string)user_id()) {
            throw new \CodeIgniter\Exceptions\PageNotFoundException('Anda tiada akses ke
halaman ini.');
        }
```

Fig. 21 *Submit Application Code Segment*

```
public function index()
    {
        if (auth()->user()->inGroup('superadmin', 'admin')) {
            $permohonan = $this->permohonanModel->getAllPermohonan();
        } else {
            $permohonan = $this->permohonanModel->getPermohonanByUserId(user_id());
        }
        $report = $this->laporanModel->getAllLaporan();
        $data = [
            'pretitle' => 'Laporan',
            'title' => 'Senarai Laporan',
            'ctx' => 'laporan',
            'report' => $report,
            'application' => $permohonan,
            'empty' => empty($permohonan)
        ];
        return view('laporan/index', $data);
    }
```

Fig. 22 *Manage Application Code Segment*

5.2 Testing

The testing phases focus on assessing the functionality of the system. In this section, both functional testing and user acceptance testing (UAT) are conducted. Functional testing verifies that each module within the Web-Based Program Management System operates as intended. User acceptance testing applies alpha and beta testing to verify the acceptance of stakeholders and users of the system.

5.2.1 Test Cases

Table 4 outlines the test cases for all six modules within the system. The primary objective of these tests is to verify that each module functions as intended.

Table 4 *List of Test Cases*

Login Module			
Function	**Test Case**	**Expected Output**	**Actual Output**
Login	TC_100_01	Verify that the user can enter their username and password	PASS
	TC_100_02	Verify that the system displays the home page for the valid login	PASS
	TC_100_03	Verify that the system displays the error message for the invalid login	PASS
Register Module			
Function	**Test Case**	**Expected Output**	**Actual Output**
Register	TC_200_01	Verify that the system navigates to the registration page after clicking the 'Register' button	PASS
	TC_200_02	Verify that the user can input their email	PASS
	TC_200_03	Verify that the user can create a password	PASS
	TC_200_04	Verify that the system displays a success message for the valid information	FAIL
	TC_200_05	Verify that the system displays an error message for the invalid information	PASS
	TC_200_06	Verify that the system displays an error message for existing user information	PASS

Table 4: *(cont)*

Submit Application Module			
Function	**Test Case**	**Expected Output**	**Actual Output**
Submit Application	TC_400_01	Verify that the system displays the application submission page.	PASS
	TC_400_02	Verify that the system provides a form for users to complete.	PASS
	TC_400_03	Verify that the system allows users to attach supporting documents during application submission.	PASS
	TC_400_04	Verify that the system validates the completeness and validity of the submitted application.	PASS
	TC_400_05	Verify that the system provides a success message upon successful submission.	PASS
Manage Application Module			
Function	**Test Case**	**Expected Output**	**Actual Output**
Manage Application	TC_500_01	Verify that the system lists all the submitted applications awaiting review by PSU coordinators.	PASS
	TC_500_02	Verify that the system presents a detailed content of selected program applications for PSU coordinators to review.	PASS
	TC_500_03	Verify that the system allows PSU coordinators to approve an application, updating its status to "Approved".	PASS
	TC_500_04	Verify that the system allows PSU coordinators to reject an application, update its status to "Rejected", and add feedback.	PASS
Track Progress Module			
Function	**Test Case**	**Expected Output**	**Actual Output**
Track Progress	TC_600_01	Verify that the system displays the status of the user's application.	PASS
	TC_600_02	Verify that the system presents detailed content of the user's program applications.	PASS
	TC_600_03	Verify that the system allows user to edit and save their program application.	PASS
	TC_600_04	Verify that the system allows user to cancel and delete their program application.	PASS
Manage Report Module			
Function	**Test Case**	**Expected Output**	**Actual Output**
Manage Report	TC_700_01	Verify that the system displays the program report page.	PASS
	TC_700_02	Verify that the system lists all completed programs.	FAIL
	TC_700_03	Verify that the system allows users to attach program and financial reports for PSU coordinators to review.	PASS

5.2.2 Overall Test Result

In this section, we will discuss the summary of all test results. The system underwent testing on 6 modules with 25 test cases. Table 5 presents a comprehensive overview of the test case results.

Table 5 *Test Result of Test Cases*

Test Cases	**Total Test Cases**	**Total Success**	**Total Failed**	**Pass Percentage**	**Fail Percentage**
TC_100	3	3	0	100%	0%
TC_200	6	5	1	83.33%	16.67%
TC_300	5	5	0	100%	0%
TC_400	4	4	0	100%	0%
TC_500	4	4	0	100%	0%
TC_600	3	2	1	66.67%	33.33%
Overall	**25**	**23**	**2**	**93.10%**	**6.90%**

5.2.3 User Acceptance Testing

An important step in the development of the Web-Based Program Management System is user acceptance testing (UAT). This phase includes both alpha and beta testing. The alpha testing is carried out by a PSU coordinator, while a chosen group of 8 students participates in the beta testing. The results of the alpha testing conducted by stakeholders are summarized in Table 6.

Table 6 *Summary of Alpha Testing*

No.	Acceptance Criteria	Test Result	Pass Percentage (%)
1	Admin can log in to the system using the provided username and password.	PASS	100

Table 6: *(cont)*

2	Admin can view the main page after successfully logging in.	PASS	100
3	Admin can view the list of applications submitted by students.	PASS	100
4	Admin can review the details of each submitted application.	PASS	100
5	Admin can approve or reject applications.	PASS	100
6	Admin can add comments on applications.	PASS	100
7	Admin can view the report for all applications.	PASS	100
	Total Pass Percentage		**100**

Beta testing follows alpha testing, involving external users in real-world evaluations to gather feedback on usability and performance. Table 7 shows the Positive Value Analysis (PVA) for the feedback.

Table 7 *Positive Value Analysis (PVA)*

Question	Positive Responses	Total Responses	Positive Value (%)
1	7	8	100
2	7	8	100
3	7	8	87.50
4	6	8	87.50
5	6	8	87.50
6	7	8	87.50
7	7	8	87.50
8	6	8	100
9	6	8	100
Total	**59**	**72**	**81.94**

6. Conclusion

In conclusion, the project aims to develop a web-based system for Pusat Sukan Universiti (PSU) to streamline the program application process. The objectives include enhancing user experience by providing a user-friendly interface, transitioning from manual processes to automation for improved efficiency, securely storing program information in a database, and offering an overview of organised programs for better management.

For future updates, several enhancements are being considered to improve the overall system. These enhancements aim to make the user interface more intuitive and introduce automated notifications and reminders. Additionally, improvements are planned for reporting features to provide deeper insights, incorporating feedback mechanisms for ongoing enhancements, and strengthening data security protocols. The focus of these improvements is to create a system that is more robust and user centred.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: study conception and design: Mohammad Ridhwan Suriansah, Mohd Zanes Sahid; data collection: Mohammad Ridhwan Suriansah; analysis and interpretation of results: Mohammad Ridhwan Suriansah, Mohd Zanes Sahid; draft manuscript preparation: Mohammad Ridhwan Suriansah, Mohd Zanes Sahid. All authors reviewed the results and approved the final version of the manuscript.

References

[1] Pusat Sukan Universiti. (n.d.). Mengenai kami. Pusat Sukan Universiti UTHM. https://sukan.uthm.edu.my/index.php/homepage/mengenai-kami
[2] Nygiyeva, A., Zhumaly, A., Soltanbayeva, A., Maksat, S., & Hamada, M. A. (2021). The role of Management information systems and Technology on business profitability. 2021 IEEE International Conference on Smart Information Systems and Technologies (SIST). https://doi.org/10.1109/sist50301.2021.9465905
[3] Rainer, R. K., Prince, B., Sánchez-Rodríguez, C., Hogeterp, I. S., & Ebrahimi, S. (2020). Introduction to information systems: Supporting and Transforming Business.
[4] ISO - Management system standards. (n.d.). ISO. https://www.iso.org/management-system-standards.html
[5] UM Event System. (n.d.). https://umevent.um.edu.my/
[6] Event Marketing & Management Solutions | CVENT. (n.d.). https://www.cvent.com/en/event-marketing-management
[7] Event management system for conferences and trade shows | Whova. (n.d.). Whova. https://whova.com/event-management-software/
[8] Zhang, X., Lv, S., Xu, M., & Mu, W. (2010). Applying evolutionary prototyping model for eliciting system requirement of meat traceability at agribusiness level. Food control, 21(11), 1556-1562.
[9] Mule, S. S., Waykar, Y., & Mahavidyalaya, S. V. (2015). Role of USE CASE diagram in s/w development. International Journal of Management and Economics.
[10] Al-Fedaghi, S. (2017). Diagramming the class diagram: toward a unified modeling methodology. arXiv preprint arXiv:1710.00202.