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

AITCS

Vol. 6 No. 1 (2025) 2036-2055
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

TroubleShot – Troubleshooting Hub

Haikal Iskandar[1], Rozlini Mohamed[1]*

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

*Corresponding Author: rozlini@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.113

Article Info

Received: 30 July 2024
Accepted: 18 June 2025
Available online: 30 June 2025

Keywords

Digital Troubleshooting, IT Operations Efficiency, Prototyping Development Model, Centralized Solution, Real-time Tracking

Abstract

The TroubleShot – Troubleshooting Hub project innovates the IT troubleshooting process of Regal Motor Holdings. Addressing the inefficiencies of the manual method utilized by the IT team, this project introduces a centralized digital solution. The aim is to streamline processes, enhance collaboration, and improve cost-effectiveness in IT operations. The manual method previously usually leads to delays, undocumented informations and costs the company more time and money. The proposed system resolves all this issue by streamlining the process of troubleshooting. Utilizing an object-oriented, web-based approach, the project employs the Prototyping Development Model for iterative development. Key features include a comprehensive Troubleshooting Library, advanced search, real-time tracking, performance evaluation, and detailed reporting. This digital hub significantly quickens issue resolution, fosters team collaboration, and optimizes resource use, marking a substantial improvement in IT troubleshooting efficiency.

1. Introduction

The landscape of Information Technology (IT) troubleshooting within corporate environments is rapidly evolving, necessitating efficient and streamlined approaches to address emerging challenges [1]. Troubleshooting is the process of identifying, diagnosing, and resolving problems or issues that occur within a system, device, software, or any complex machinery [2]. Regal Motor Holdings, a prominent player in its industry, has encountered significant inefficiencies in its IT troubleshooting processes, primarily due to reliance on manual documentation and disjointed information management systems. These traditional methods have led to considerable delays, confusion among IT staff, and escalated operational costs, thereby undermining the overall efficiency and productivity of the organization.

Recognizing the critical need for a transformative solution, this research introduces the "TroubleShot – Troubleshooting Hub" project. This initiative is aimed at revolutionizing the existing IT troubleshooting framework within Regal Motor Holdings. The project proposes the development of a centralized digital solution, specifically designed to enhance the efficiency, collaboration, and cost-effectiveness of IT troubleshooting endeavors [3].

Central to this project is the objective to design and implement a comprehensive digital hub, utilizing an object-oriented, web-based approach. This innovative strategy is poised to significantly streamline technology troubleshooting processes within the corporate IT environment. The envisioned system is not only a tool for problem-solving but also a platform for enhancing knowledge sharing and collaboration among IT professionals.

The initiative is structured around the deployment of the Prototyping Development Model. This involves a series of phases, beginning with an in-depth requirement analysis, which includes brainstorming sessions with IT professionals, comprehensive reviews of existing documentation, and detailed interviews with key department members. Observations of daily IT operations are also integral to this phase, ensuring a holistic understanding of the current challenges and requirements.

The expected outcomes of the TroubleShot – Troubleshooting Hub project are multifaceted. Primarily, it aims to significantly improve the resolution times of IT issues, thereby reducing the downtimes and enhancing the overall operational efficiency. Moreover, the project seeks to foster a culture of collaboration among IT staff, enabling a more cohesive and integrated approach to troubleshooting. Finally, by optimizing resource utilization, the project is anticipated to yield considerable cost savings, contributing to the financial sustainability of the organization's IT operations.

In summary, the TroubleShot – Troubleshooting Hub project represents a strategic initiative to address the existing challenges in IT troubleshooting at Regal Motor Holdings. By leveraging a centralized digital solution, the project aims to transform the IT troubleshooting process, leading to enhanced efficiency, collaboration, and cost-effectiveness.

2. Related Work

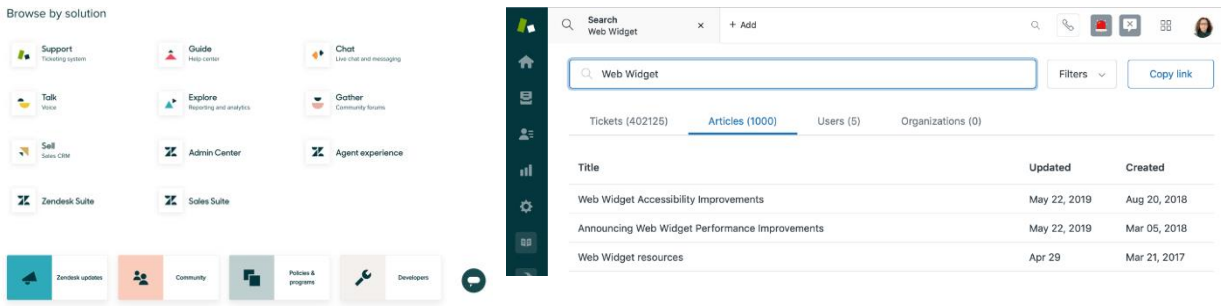
The realm of IT troubleshooting and digital solution development is extensive and multifaceted, encompassing various methodologies, technologies, and frameworks. In this context, the literature review and analysis of related work provide valuable insights into the current state of the art and the positioning of the TroubleShot – Troubleshooting Hub project within this landscape [4].

2.1 Systematic Approaches to Troubleshooting

The literature reveals a myriad of systematic approaches employed in troubleshooting complex systems, ranging from electronics to software. These methodologies underscore the importance of structured problem-solving, highlighting the necessity for coherent strategies in IT troubleshooting. Studies indicate that a systematic approach not only accelerates the resolution process but also enhances the accuracy and reliability of the solutions [5].

2.2 Existing Digital Solutions in IT Troubleshooting

A critical examination of existing digital solutions, such as Zendesk, Freshdesk, and Jira, reveals their strengths and limitations in managing IT issues. These platforms offer functionalities like troubleshooting libraries, search tools, and tracking systems. However, the analysis indicates a gap in their ability to fully cater to the specific needs of corporate IT environments, particularly in terms of customized integration and user-centric design.



(a) **(b)**

Fig. 1 *Knowledge Base Page in Zendesk Website. (a) The main interface of Zendesk's knowledge base solutions, including a variety of support and service modules; (b) Zendesk's search interface showcasing the filtering and search capabilities across different content types.*

2.3 Comparative Analysis

The comparative analysis of these existing systems with the proposed TroubleShot – Troubleshooting Hub is essential. It demonstrates how the hub aims to address the identified gaps by offering a more tailored solution for Regal Motor Holdings. This includes a focus on seamless integration, enhanced user experience, and features specifically designed for the corporate IT troubleshooting context.

2.4 Theoretical Frameworks and Models

Theoretical frameworks and models relevant to IT troubleshooting and digital solution development are also reviewed. These frameworks provide the foundational principles guiding the design and implementation of the TroubleShot – Troubleshooting Hub, ensuring that the solution is grounded in established theories and practices.

2.5 Integration of Advanced Technologies

The role of advanced technologies, such as artificial intelligence (AI) and machine learning (ML), in enhancing IT troubleshooting processes is also explored. The literature suggests that integrating these technologies can significantly improve the efficiency and effectiveness of troubleshooting systems.

2.6 Utilization of Case Studies and Real-World Examples

Case studies and real-world examples are instrumental in illustrating the practical applications and benefits of well-designed IT troubleshooting systems. They provide concrete evidence of how similar systems have successfully addressed complex troubleshooting challenges in various settings. The review of related work thus establishes a comprehensive backdrop against which the TroubleShot – Troubleshooting Hub project is situated. It not only validates the need for an innovative solution like the proposed hub but also lays the groundwork for its development based on best practices and proven methodologies. Table 1 below displays the comparison between related systems.

Table 1 *System's Comparison*

Features/System	Zendesk	Freshdesk	Jira	TroubleShot
Troubleshooting Library Module	√	√	√	√
Searching Module	√	√	√	√
Tracking Module	√	√	√	√
Evaluation Module	√	√	√	√
Report Module	√	√	√	√

3. Methodology/Framework

The process of developing TroubleShot – Troubleshooting Hub utilizes the Prototyping Model as the system development methodology. The Prototyping Model in software development involves constructing a prototype, followed by testing and iterative refinement until an acceptable prototype is achieved. This model is particularly effective in ensuring that the final system meets user requirements and expectations by incorporating continuous feedback and improvements. In this chapter, we will also discuss the selection of requirements, which are essential for developing the TroubleShot – Troubleshooting Hub as planned.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 2036-2055

3.1 System Development

The model chosen for this project is the Prototyping Model. A Prototyping Model in software development involves the construction of a prototype, followed by testing and iterative refinement until an acceptable prototype is achieved [5]. The prototyping model consists of several phases which are initial requirements, design, prototyping, customer evaluation, reviews and updates, implementation, and testing.

The initial requirements phase consists of the planning and analysis to get the gist of the current method of troubleshooting done in the Regal Motor Holdings company in order to obtain a detailed project requirement which includes the UML Diagrams and Entity Relationship Diagram. The database schema, system architecture and user interface are part of the design phase. The first prototype is created based on the project requirements. After the customer evaluation phase, a new and improved prototype is created. The system is then finalized after all the requirements are met. The testing phase will be the testing of the final system to identify and search for possible errors or bugs in the system. Table 2 below shows the phases and the tasks and outputs associated with said phases.

Table 2 *Software Development Activities and Their Tasks*

Phase	Task	Output
Initial Requirements	• Review of existing documents • Interviews with IT department members • Visualization of the system's design	• Proposal • UML Diagrams • Project Requirements • ERD
Design	• Create database schema. • Create system architecture. • Design user interface.	• Design of user interface. • System Architecture • Database Design
Prototyping	• Constructing the prototype using PHP, JavaScript, and Laravel framework in Visual Studio Code	• Prototype of the IT troubleshooting system
Customer Evaluation	• Receive feedback based on the current prototype from the company.	• Regal Motor Holdings feedback.
Review and Update	• Review the feedback. • Refine prototype. • Add new features based on requirements.	• Improved Prototype
Implementation	• Finalize the system. • Combine all modules into one complete system.	• Fully developed IT troubleshooting system ready for deployment
Testing	• Test the overall usability of the system.	• Validated and reliable IT troubleshooting system.

This table outlines the systematic approach taken during the development of the TroubleShot – Troubleshooting Hub, detailing the specific activities and tasks carried out in each phase, along with their corresponding outputs. It serves as a roadmap of the project's lifecycle, showcasing the meticulous process from initial requirements gathering to the final implementation.

3.2 Requirement Analysis

System Requirement Analysis is a process of identifying and understanding the needs of stakeholders for a system [7]. Analyzing system requirements is a vital step in understanding how the TroubleShot – Troubleshooting Hub will work and its limitations. This process includes breaking down modules and specifying functional, non-functional, and user requirements to guide its development.

Table 3 Functional Requirements of the proposed system

Requirement	Description
Troubleshooting Library	The system allows Staffs to report and view technical issues while IT Professionals should be able to view and manage and technical issues and add solutions.
Searching	The system should enable users to search for issues using keywords or tags and display the respective issues.
Tracking	The system should allow Staffs to view status of issues and IT Professionals should be able to update the status of issues. (Pending, Ongoing, Solved). The system automatically records response and resolution times.
Evaluation	The system should allow managers to evaluate the performance of IT team, compare capabilities based on issue types and provide performance analytics.
Reporting	The system should allow managers to view and download monthly reports based on troubleshooting records, resources, and costs.

Table 3 summarizes the functional requirements of the proposed system. Different users have different levels of access to each specific modules in the system depending on their respective roles. With Staffs being able to view and report technical issues and IT Professionals are able to manage and add solutions to issues. All users should be able to search for specific issues in the library. Staff should be able to view the status of issues and IT professionals should be able to update the statuses. Managers will be to evaluate IT team performance and generate monthly reports consisting of troubleshooting records, resources used and costs.

Table 4 Non-Functional Requirements of The Proposed System

Requirement	Description
Performance	The system should demonstrate high responsiveness and speed, with minimal latency in loading data and executing functions, ensuring efficient issue tracking and resolution.
Usability	The system must be user-friendly, with an intuitive interface that is easy to navigate for both IT Professionals and Staff, reducing the learning curve and enhancing user satisfaction.
Security	Sensitive data, including user information and technical issue details, must be securely stored, and protected from unauthorized access or breaches.
Compatibility	The system should be compatible with various browsers and devices to ensure accessibility across different platforms used within the organization.

Maintainability	The system should be easy to maintain and update, allowing for quick fixes, upgrades, and adaptations to evolving IT needs.

Table 4 presents the non-functional requirements for the proposed system, emphasizing its need for efficiency, ease of use, secure data management, cross-platform compatibility, and simple maintenance.

The development is recommended to be carried out on a device with Windows 10 or above, with a 2.0Ghz quad core processor, 8GB of RAM, 256GB of HDD or SSD storage. Wi-fi connections are recommended. The system should be tested on Google Chrome, Microsoft Edge, or Opera GX and MySQL are used for database management and querying. The recommended development tools are Laragon, Laravel Framework and Visual Studio Code along with the programming language of PHP, HTML and CSS.

3.3 System Analysis

Figure 1 depicts the use case diagram for TroubleShot – Troubleshooting Hub, illustrating the process of viewing and managing issues for both Staffs and IT Professionals, the process of tracking and searching issues for all users. Lastly the process for evaluating the performance of the IT Team and generating reports for the Manager.

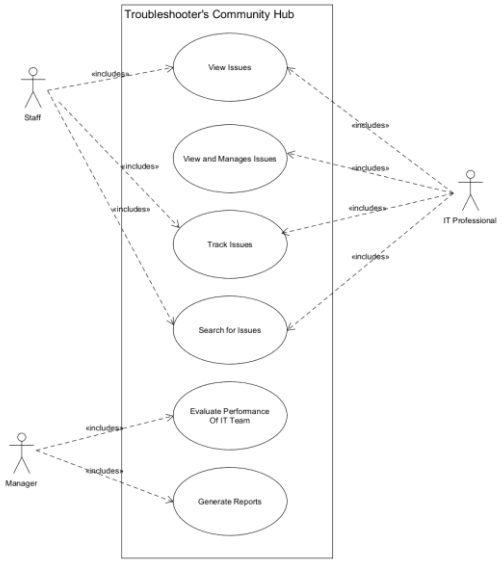


Figure 1: Use Case Diagram

The entity relationship diagram (ERD) in Figure 2 shows the connection between the entities in the system. The identified entities are User, Issue, Tag, Solution, Tracker, PerformanceEvaluator, and Report. Each entities have their own operations.

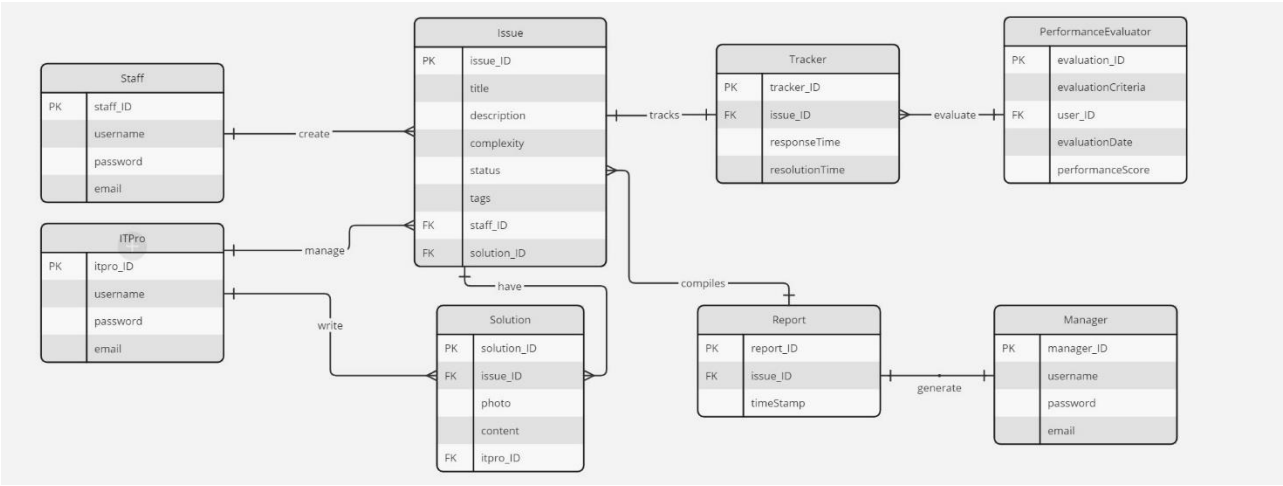
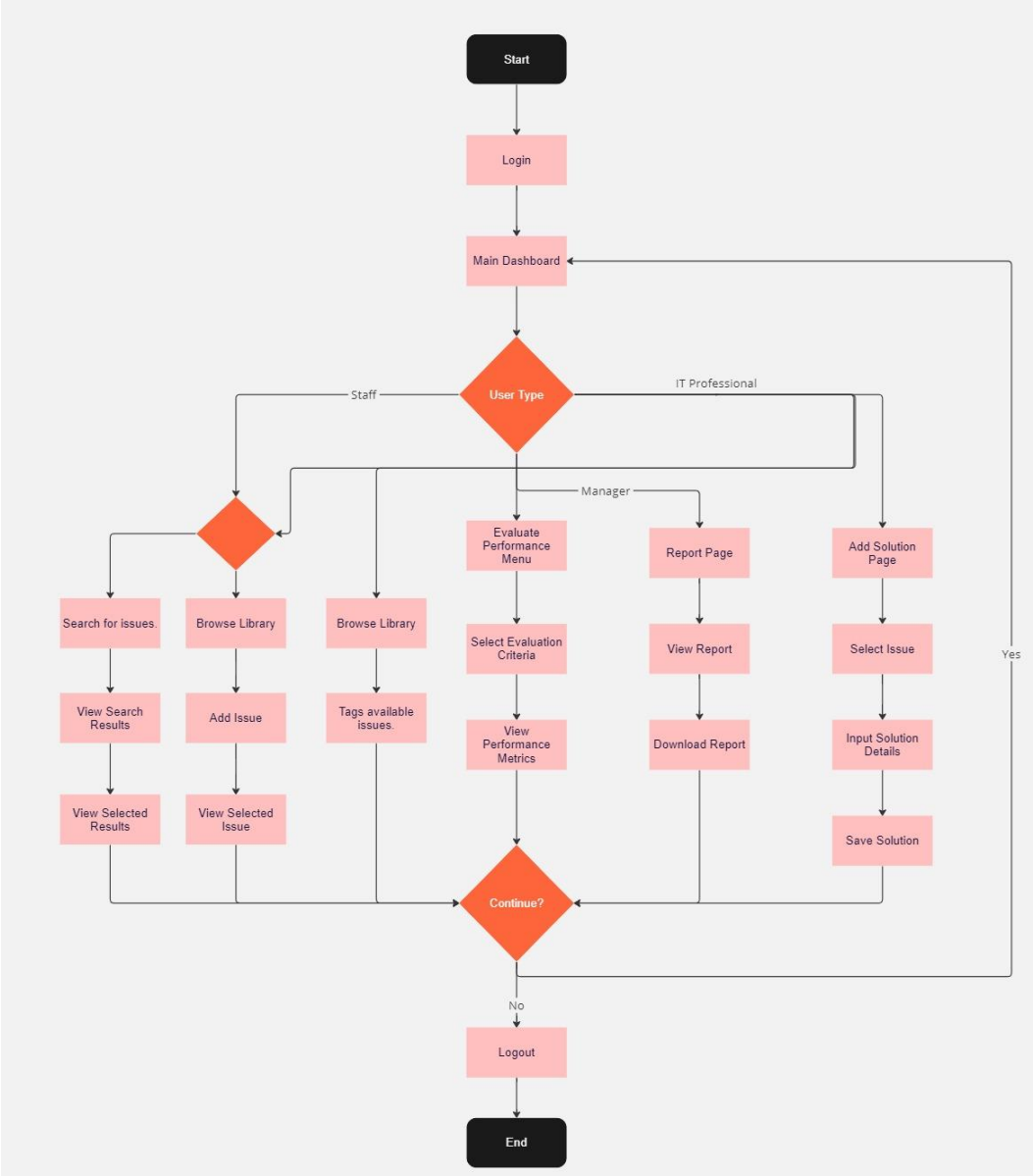


Figure 2: Entity Relationship Diagram (ERD)

The Flowchart of the proposed system is shown in Figure 3. It showcases the flow of processes in the system which aids the analysis and design of the system.



Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 2036-2055

Figure 3: Flowchart of Proposed System

3.4 System Design

The Figure 4 below shows the system architecture diagram of the TroubleShot - Troubleshooting Huboutlining the user roles and the processes they go through.

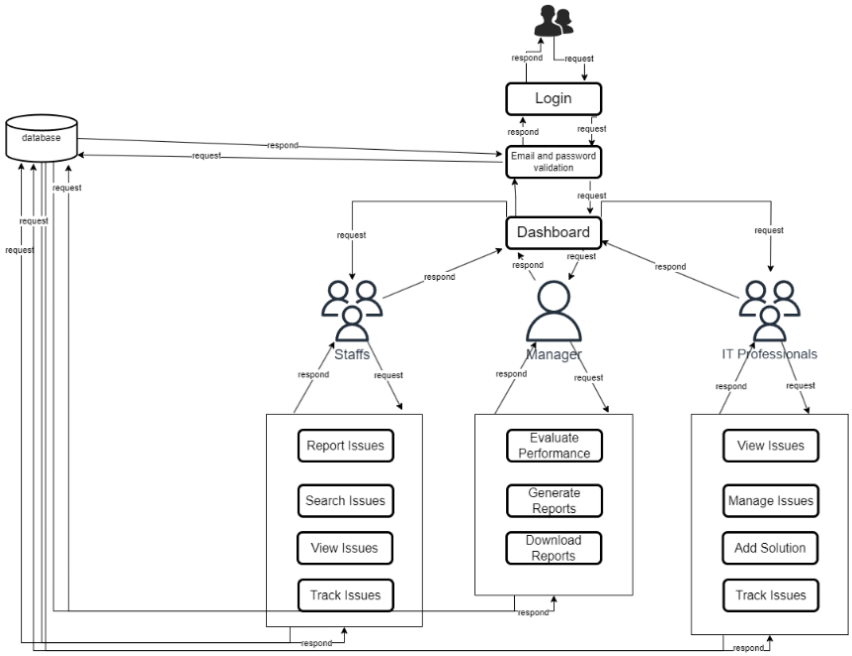


Figure 4: System Architecture Design

The following are the user interface design of each page in the system. UI design, or user interface design, pertains to the visual and interactive aspects of a digital product's interface, encompassing its appearance and functionality [8]. The interfaces are designed with usability and aesthetics kept in mind.

3.4.1 Dashboard

The dashboard interface in Figure 5 is a personal page where Staffs can choose the activity they want to proceed with such as browsing the Troubleshooting Library, searching for specific issues or track their reported issues that are still ongoing.

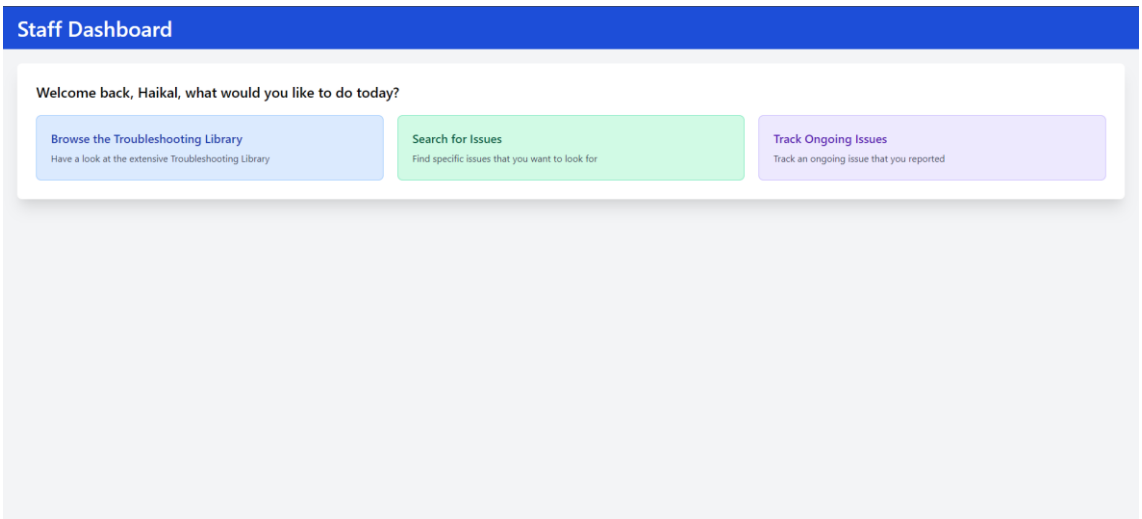


Figure 5: Dashboard Page

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 2036-2055

3.4.2 Troubleshooting Library

The Troubleshooting Library shown in Figure 6 is the library where all the technical issues are situated, Staffs or IT Professionals can see whether an issue is pending, ongoing or solved. They can click on any individual issues to open the issue page for more details.

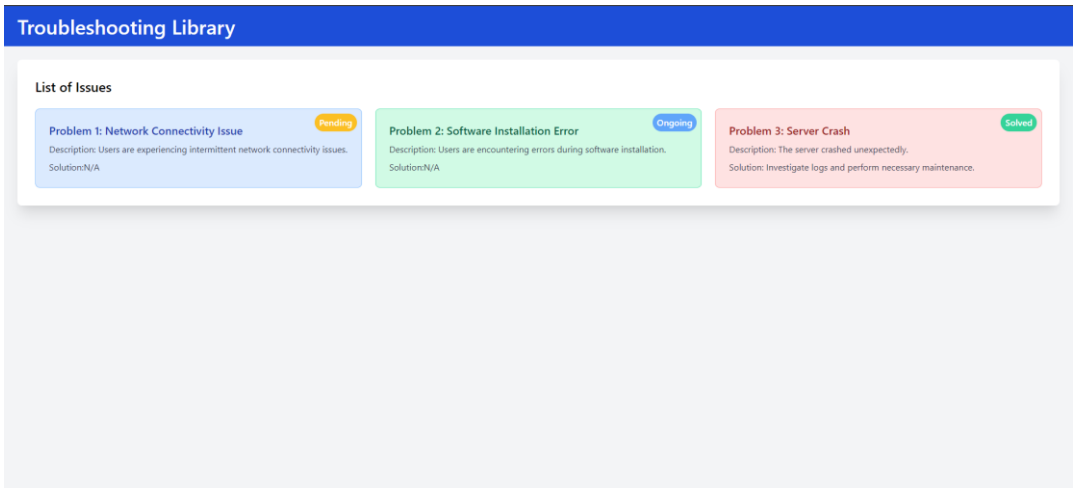


Figure 6: Troubleshooting Library Page

3.4.3 Issues Details Page

The issue details page in Figure 7 below is where more details regarding a technical issue as shown such as reported and resolution time, issue type (hardware or software) and a progress bar for tracking the issue's current state.

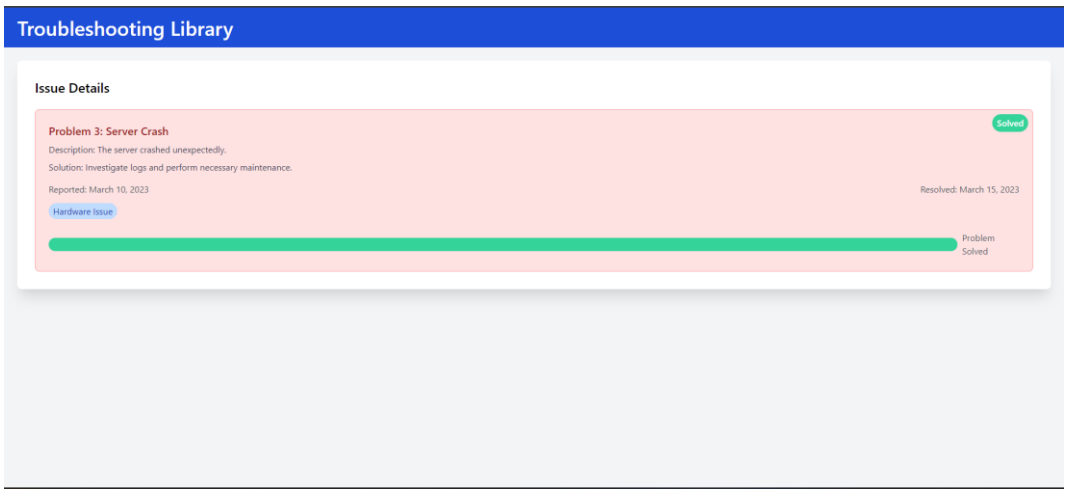


Figure 7: Issues Details Page

3.4.4 Evaluation Page

In this evaluation page, Managers can evaluate the performance of the IT Team with the information displayed in the charts which are the IT Team performance in resolving issues.

Figure 8: Evaluation Page

4. Result and Discussion

This section delves into the implementation of the web-based Troubleshooting Hub System, including the development of the troubleshooting library, search, tracking, evaluation, and reporting modules. Additionally, it discusses the comprehensive functional testing performed to assess the system's performance.

4.1 Implementation

This section outlines the development of functional modules within a system, accompanied by program code for clarity.

4.1.1 Troubleshooting Library Module

```
1 reference | 0 overrides
public function login(Request $request)
{
    $email = $request->input('email');
    $password = $request->input('password');

    if (Auth::guard('staff')->attempt(['email' => $email, 'password' => $password])) {
        session(['user_role' => 'Staff']);
        session(['username' => Auth::guard('staff')->user()->username]);
        return redirect('/dashboard');
    }

    if (Auth::guard('itpro')->attempt(['email' => $email, 'password' => $password])) {
        session(['user_role' => 'IT Professional']);
        session(['username' => Auth::guard('itpro')->user()->username]);
        return redirect('/dashboard');
    }

    if (Auth::guard('manager')->attempt(['email' => $email, 'password' => $password])) {
        session(['user_role' => 'Manager']);
        session(['username' => Auth::guard('manager')->user()->username]);
        return redirect('/dashboard');
    }

    // Authentication failed
    return redirect()->back()->withErrors(['login_error' => 'Invalid credentials']);
}
```

Figure 9: Login Source Code

```php
2 references | 0 implementations
class ForgotPasswordController extends Controller
{
    1 reference | 0 overrides
    public function resetPassword(Request $request)
    {
        $email = $request->input('email');
        $userRole = $request->input('user_role');

        // Determine which model to use based on the user role
        $user = null;
        switch ($userRole) {
            case 'itpro':
                $user = ITPro::where('email', $email)->first();
                break;
            case 'manager':
                $user = Manager::where('email', $email)->first();
                break;
            case 'staff':
                $user = Staff::where('email', $email)->first();
                break;
        }

        if (!$user) {
            return redirect()->back()->withErrors(['email' => 'Invalid email or user role']);
        }


        return redirect('/login')->with('success', 'Password reset successful');
    }
}
```

Figure 10: Forgot Password Source Code

```php
18 references | 0 implementations
class IssuesController extends Controller
{
    1 reference | 0 overrides
    public function index()
    {
        $issues = Issues::with('solutions')->where('status', '!=', 'pending')->get();
        return view('library', ['issues' => $issues]);
    }

    5 references
    protected $emailService;

    0 references | 0 overrides
    public function __construct(EmailService $emailService)
    {
        $this->emailService = $emailService;
    }

    1 reference | 0 overrides
    public function store(Request $request)
    {
        $rules = [
            'title' => 'required|string|max:255',
            'description' => 'required|string',
            'complexity' => 'required|string|max:255',
            'status' => 'required|string|max:255',
            'tags' => 'required|string|in:Hardware,Software,Both',
            'staff_id' => 'required|integer',
            'images.*' => 'nullable|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        ];

        $validatedData = $request->validate($rules);

        $issueFolderName = Str::slug($validatedData['title'] . '-' . $validatedData['staff_id'] . '-' . now()->timestamp);

        $imagePaths = [];
        if ($request->hasFile('images')) {
            foreach ($request->file('images') as $index => $image) {
                $imageName = $issueFolderName . '_' . ($index + 1) . '.' . $image->getClientOriginalExtension();
                $imagePath = $image->storeAs('public/images/issues/' . $issueFolderName, $imageName);
                $imagePaths[] = 'storage/images/issues/' . $issueFolderName . '/' . $imageName;
            }
```

Figure 11: Troubleshooting Library Source Code

The provided source code snippets show the controllers responsible for making the TroubleShot system functional. Figure 9 shows the login page code, which manages user authentication for three roles (staff, IT professional, and manager), storing the user role and username in the session upon successful login and redirecting to the dashboard, while handling errors for invalid credentials. Figure 10 illustrates the forgot password functionality in the Forgot Password Controller, processing password reset requests by verifying the user's email and role, redirecting to the login page with a success message if successful, or returning an error if the user is not found. Figure 11 depicts the Issues Controller, which includes methods to retrieve and display issues that are not pending, along with their solutions, and to validate, process, and store new issue submissions, including handling image uploads.

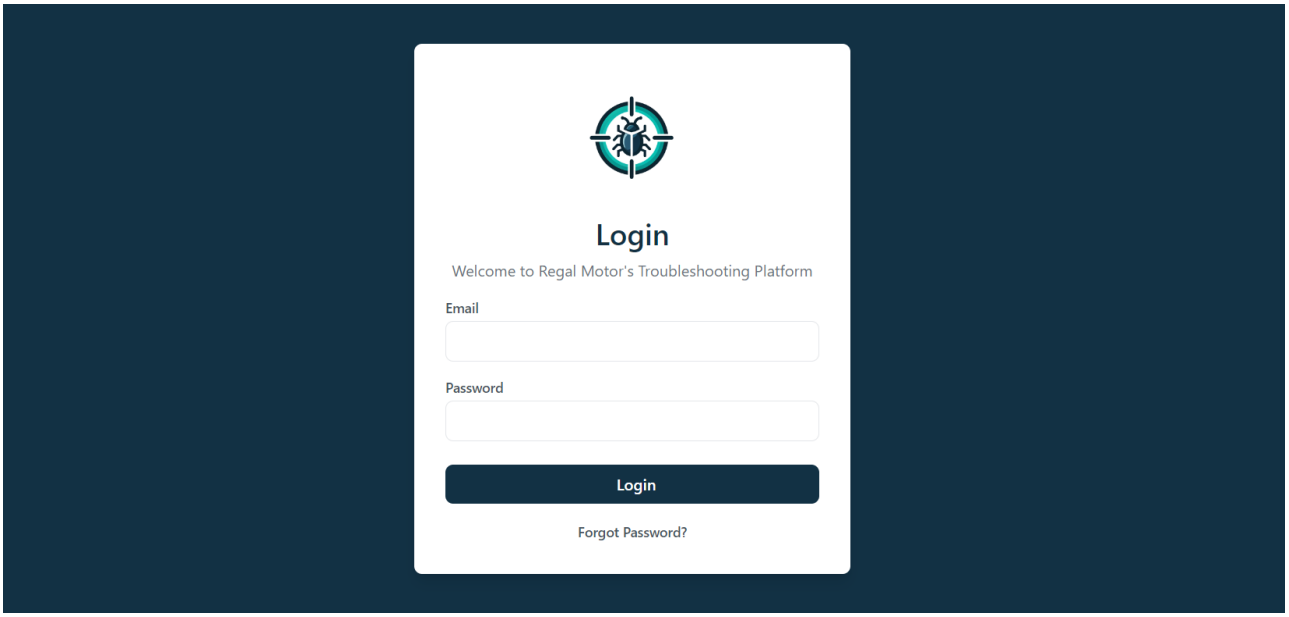


Figure 12: Login User Interface

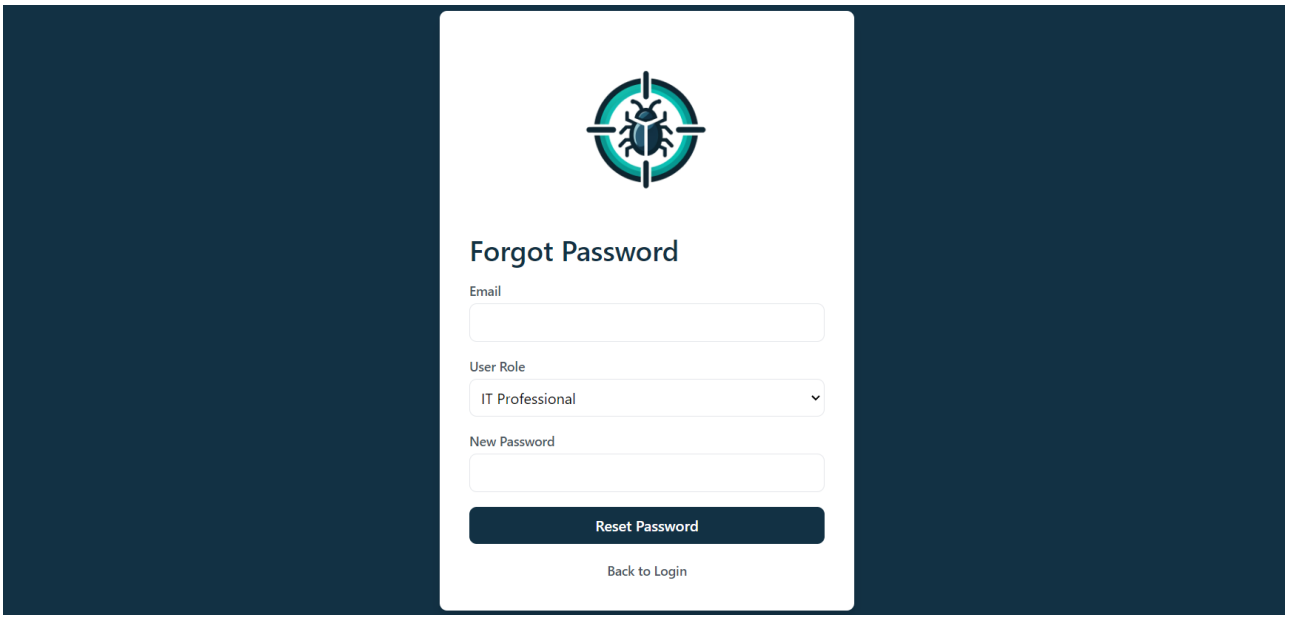


Figure 13: Forgot Password User Interface

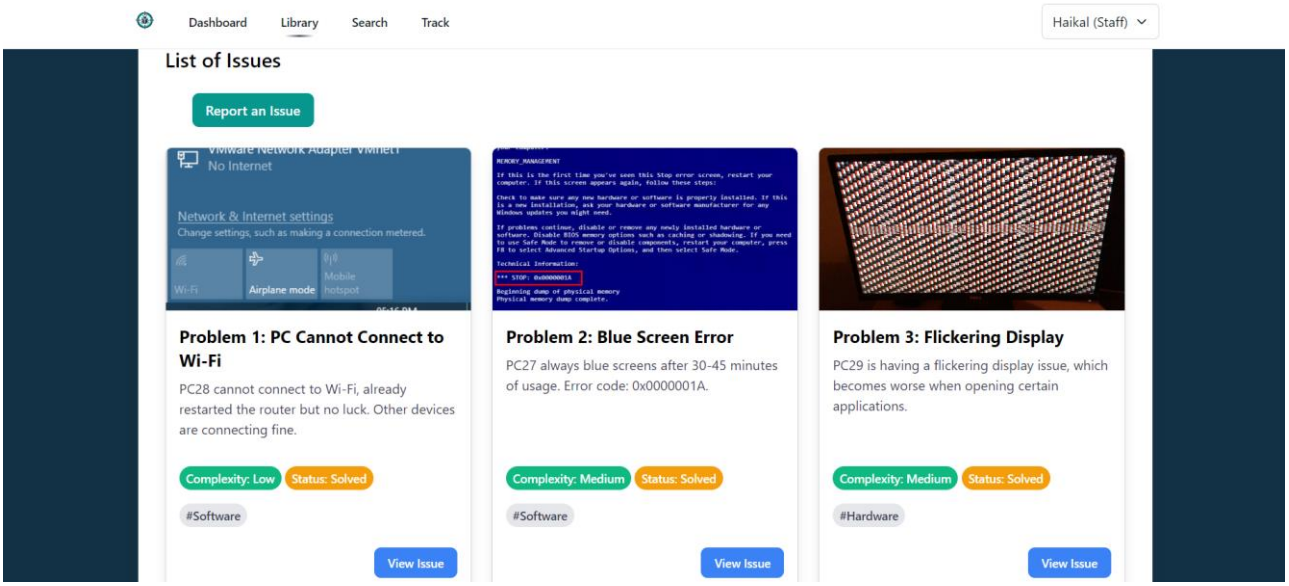


Figure 14: Troubleshooting Library User Interface

Next, Figure 12 displays the login user interface which requires users to insert their email address and password to log in to the system. Figure 13 shows the forgot password user interface which allows users to reset their password. Figure 14 illustrates the user interface of the Troubleshooting Library, where issues are stored and can be viewed by users. If the user is a Staff, they can also report an issue through the library by clicking the "Report an Issue" button which will redirect them to the Report Issue user interface.

4.1.2 Searching Module

```php
class IssuesController extends Controller
{
    public function search(Request $request)
    {
        $query = $request->input('search');
        $selectedTag = $request->input('tags');
        $selectedComplexity = $request->input('complexity');
        $selectedStatus = $request->input('status');

        $issuesQuery = Issues::query();

        if ($query) {
            $issuesQuery->where(function ($q) use ($query) {
                $q->where('title', 'like', '%' . $query . '%')
                  ->orWhere('description', 'like', '%' . $query . '%');
            });
        }

        if ($selectedTag && $selectedTag != 'all') {
            $issuesQuery->where('tags', $selectedTag);
        }

        if ($selectedComplexity && $selectedComplexity != 'all') {
            $issuesQuery->where('complexity', $selectedComplexity);
        }

        if ($selectedStatus && $selectedStatus != 'all') {
            $issuesQuery->where('status', $selectedStatus);
        }

        $issues = $issuesQuery->with('solutions')->get();
        $tags = Issues::distinct()->pluck('tags');

        return view('search', ['issues' => $issues, 'tags' =>
        $tags, 'query' => $query, 'selectedTag' => $selectedTag, 'selectedComplexity' => $selectedComplexity, 'selectedStatus' => $selectedStatus]);
    }
```

Figure 15: Search Source Code

Figure 15 shows the source code implementing the search functionality within the search method of the Issues Controller. This method allows users to search for specific issues in the troubleshooting library. It begins by retrieving the user's search query and selected filters for tags, complexity, and status from the request inputs. An initial query is constructed using the Issues model. If a search query is provided, the code adds conditions to the query to search for the input text within the 'title' and 'description' fields of the issues, using a SQL 'LIKE' clause for partial matches. Additional filters are applied based on the selected tag, complexity, and status, provided they are not set to 'all'. The method then fetches the filtered issues from the database, including their associated solutions, and retrieves distinct tags for use in the search interface. Finally, the results are passed to the 'search'

view, along with the user's original search parameters, ensuring the search form retains the selected filters and query. This functionality provides users with a robust tool for narrowing down issues in the library based on specific criteria, enhancing the overall user experience.

4.1.3 Tracking Module

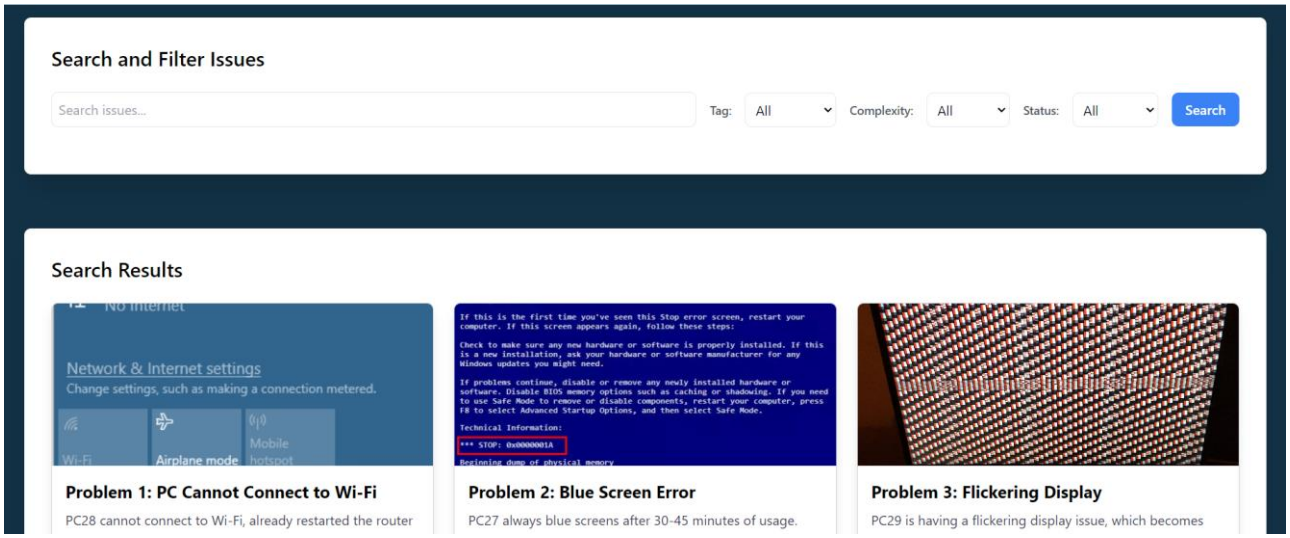


Figure 16: Searching User Interface

Moving on, Figure 16 shows the searching user interface which is composed of the search bar, tag filter, complexity filter and status filter. These options allow users to have the freedom to freely search and filter for the specific kind of issues they are looking for. Figure 18 shows the search results user interface which is the results of the searching and filtering that the user has done.

```
1 reference | 0 overrides
public function trackIssues()
{
    $staffId = auth()->user()->id;
    $issues = Issues::with(['solutions', 'itpro'])
        ->where('staff_id', $staffId)
        ->get();

    return view('track', compact('issues'));
}
```

Figure 17: Tracking Source Code

Figure 17 shows the source code for the trackIssues method in the Issues Controller. This method is designed to help staff members track the issues they are responsible for. It retrieves the ID of the currently authenticated user and then queries the Issues model to fetch all issues associated with this user's staff ID. The query includes related solutions and IT professional data, ensuring comprehensive information is retrieved. The fetched issues are then passed to the 'track' view for display, allowing staff members to monitor the progress and status of the issues they have reported or are managing. This method provides a personalized view for each staff member, enhancing their ability to stay informed and engaged with their assigned tasks.

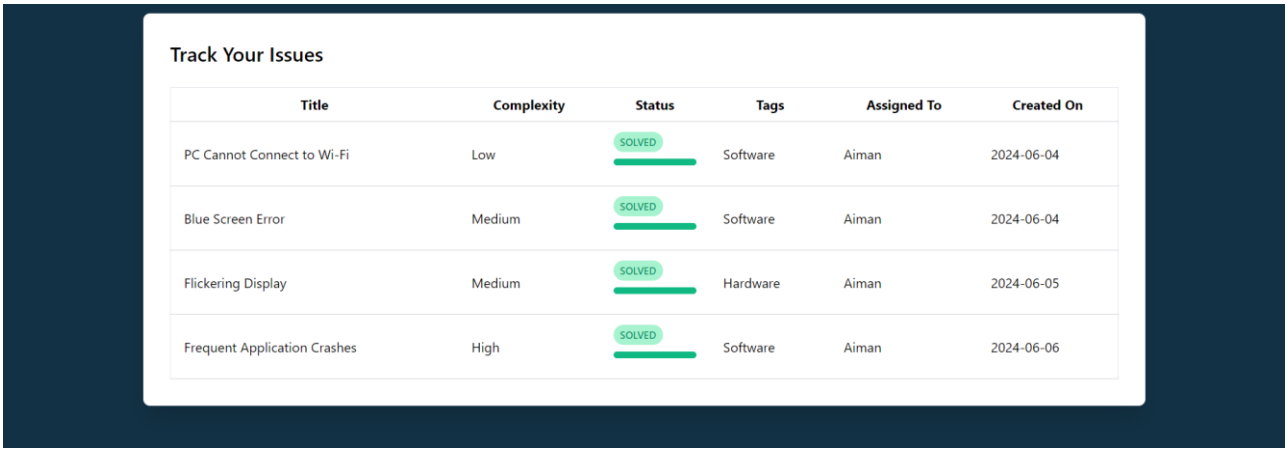


Figure 18: Tracking User Interface

Figure 18 shows the user interface for the "Track Your Issues" section, which is designed to help staff members monitor the status of issues they are handling. The table displays a list of issues assigned to the currently authenticated staff member, providing key details for each issue: the title, complexity level, current status, associated tags, assigned IT professional, and the date the issue was created. This interface allows staff members to quickly assess the progress of their reported issues, noting which ones have been solved and their respective complexity levels and tags. The clear and organized presentation aids in efficient issue management and follow-up, ensuring staff members stay informed about their responsibilities and the resolution status of each issue.

4.1.4 Evaluation Module

```
2 references | 0 implementations
class EvaluationController extends Controller
{
    1 reference | 0 overrides
    public function evaluateITPros()
    {
        $itpros = ITPro::all();

        $evaluationData = $itpros->map(function($itpro) {
            $approvedIssues = Issues::where('itpro_id', $itpro->id)->get();

            $approvedCount = $approvedIssues->count();
            $totalApprovalTime = $approvedIssues->sum(function($issue) {
                return Carbon::parse($issue->approved_at)->diffInDays(Carbon::parse($issue->created_at));
            });
            $totalSolveTime = $approvedIssues->sum(function($issue) {
                return Carbon::parse($issue->solved_at)->diffInDays(Carbon::parse($issue->approved_at));
            });

            $averageApprovalTime = $approvedCount > 0 ? round($totalApprovalTime / $approvedCount, 2) : 0;
            $averageSolveTime = $approvedCount > 0 ? round($totalSolveTime / $approvedCount, 2) : 0;
            $totalSolved = $approvedCount;

            $issueTypes = $approvedIssues->groupBy('tags')->map->count();

            return [
                'itpro' => $itpro,
                'approved_count' => $approvedCount,
                'average_approval_time' => $averageApprovalTime,
                'average_solve_time' => $averageSolveTime,
                'total_solved' => $totalSolved,
                'issue_types' => $issueTypes
            ];
        });
```

Figure 19: Evaluation Source Code

Figure 19 presents the source code for the evaluateITPros method in the EvaluationController. This method assesses the performance of IT professionals by collecting and analyzing various metrics related to the issues they have resolved. The method begins by retrieving all IT professionals from the database. For each IT professional, it gathers all approved issues they have handled. It then calculates the total number of approved issues, the total approval time, and the total solve time by parsing and computing the time differences using the Carbon library. The average approval time and average solve time are derived from these totals. The method also counts the total number of issues solved and groups the issues by tags to categorize the types of issues handled. This evaluation data, including the IT professional's details, counts, average times, and issue types, is compiled into an array and

passed to the 'evaluate-itpro' view. This view likely displays a comprehensive performance evaluation, providing insights into the efficiency and effectiveness of each IT professional in resolving issues.

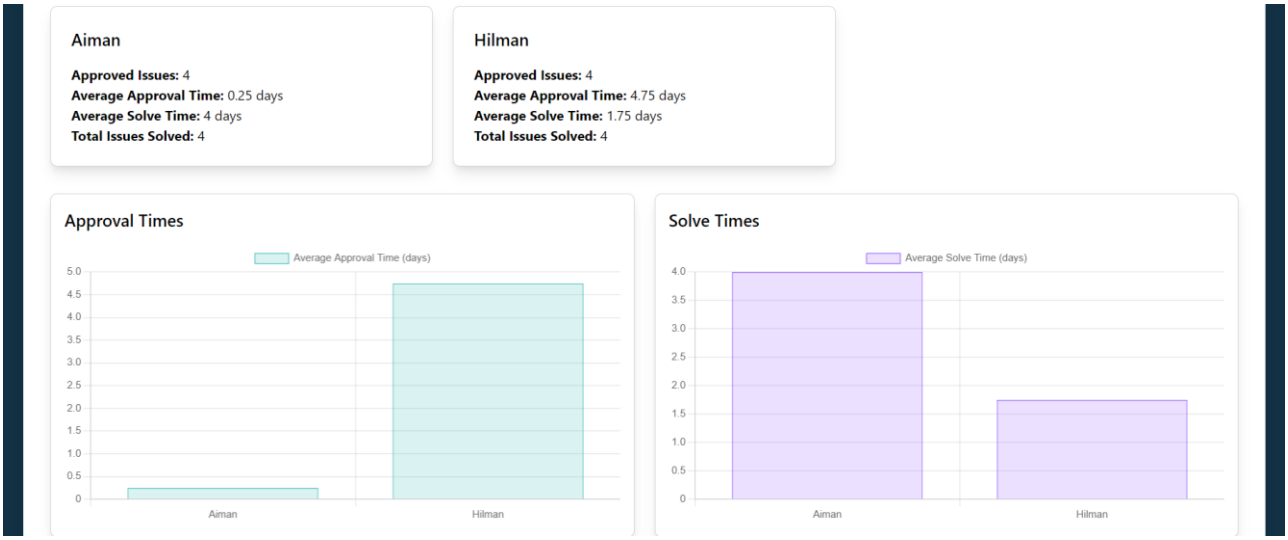


Figure 20: Evaluation User Interface

Figure 20 illustrates the user interface for evaluating IT professionals, showcasing key performance metrics and visual representations. The top section provides a summary for each IT professional, displaying the number of approved issues, average approval time, average solve time, and total issues solved. For instance, Aiman has resolved 4 issues with an average approval time of 0.25 days and an average solve time of 4 days, while Hilman has also resolved 4 issues but with a significantly higher average approval time of 4.75 days and a lower average solve time of 1.75 days. Below the summary, two bar charts visually represent these metrics: one chart compares the average approval times, highlighting Aiman's efficiency in getting issues approved quickly, and the other chart compares the average solve times, showing the time taken by each professional to resolve issues. This interface helps in evaluating the performance and efficiency of IT professionals in handling and resolving issues, providing clear insights through both data and visual aids.

4.1.5 Reporting Module

```
16 references | 0 implementations
class ReportController extends Controller
{
    0 references | 0 overrides
    public function __construct()
    {
        $this->middleware('auth:manager'); // Ensure only managers can access
    }

    2 references | 0 overrides
    public function index(Request $request)
    {
        $issues = Issues::all();
        $totalIssues = $issues->count();
        $statusCounts = $issues->groupBy('status')->map->count();
        $complexityCounts = $issues->groupBy('complexity')->map->count();
        $tagCounts = $issues->groupBy('tags')->map->count();
        $staffIssues = DB::table('issues')
            ->join('staff', 'issues.staff_id', '=', 'staff.id')
            ->select('staff.username as staff_name', DB::raw('count(*) as total'))
            ->groupBy('staff.username')
            ->get();
        $itProIssues = DB::table('issues')
            ->join('itpros', 'issues.itpro_id', '=', 'itpros.id')
            ->select('itpros.username as itpro_name', DB::raw('count(*) as total'))
            ->groupBy('itpros.username')
            ->get();
        $reevaluationCount = $issues->where('needs_reevaluation', true)->count();
```

Figure 21: Reporting Source Code

Figure 21 shows the source code for the reporting module implemented in the Report Controller. The index method aggregates various statistics related to the issues tracked in the system. It starts by counting the total

number of issues and then groups and counts issues based on their status, complexity, and tags. The method also retrieves and counts issues reported by staff and IT professionals by joining the issues table with the staff and it pros tables respectively, grouping the results by the username and calculating the total number of issues each user has reported. Additionally, the method counts the number of issues that need reevaluation. These aggregated data points are then passed to the 'report' view, which will likely display the total number of issues, distributions of issues by status, complexity, and tags, as well as detailed counts of issues reported by individual staff and IT professionals. The view might also include a line chart showing the trend of issues reported over time and a pie chart illustrating the distribution of issue statuses, providing a comprehensive overview of the issue reports in the system.

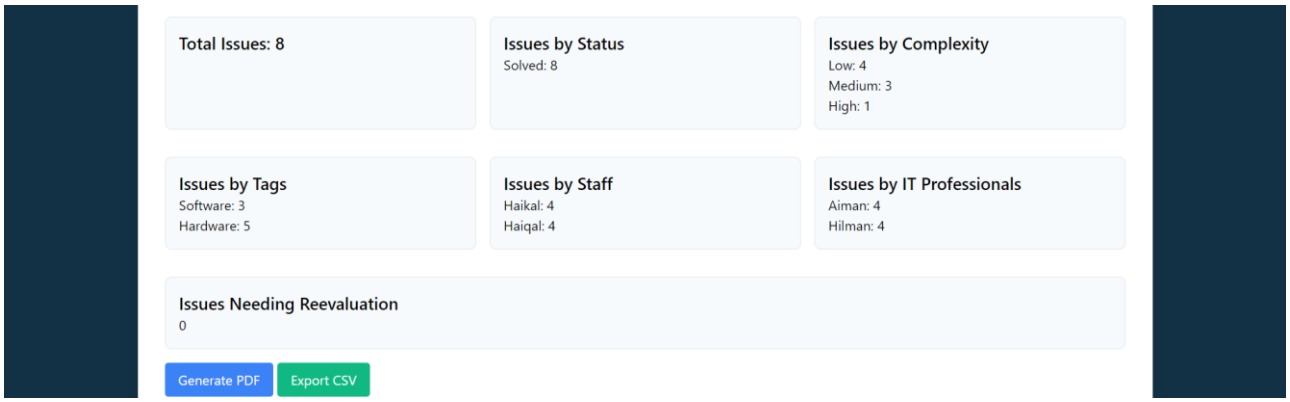


Figure 22: Reporting User Interface

Figure 22 shows the user interface of the reporting module, presenting various sections and metrics related to issue tracking. It includes a summary of the total number of issues, categorized by status, complexity, tags, staff, and IT professionals. Each section provides a quick overview, such as the number of issues solved, the complexity levels (low, medium, high), and the distribution of issues by tags like software and hardware. Additionally, it lists issues reported by individual staff members and IT professionals, and highlights any issues needing reevaluation, which currently stands at zero. The interface also features options to generate a PDF or export a CSV file, facilitating the management and reporting of issues within the system.

4.2 Testing Result

Module	Description	Expected Result	Actual Result	Pass/Fail
Troubleshooting Library Module	To check whether staff, it professional and manager can log in to the system.	Staff, it professional and manager is able to log in to the system.	Staff, it professional and manager successfully logged in to the system.	Pass
	To check whether staff, it professional and manager can view the Troubleshooting Library.	Staff, it professional and manager is able to view the Troubleshooting Library.	Staff, it professional and manager successfully viewed the Troubleshooting Library	Pass
	To check whether staff can report issues to the Troubleshooting Library.	Staff is able to report issues to the Troubleshooting Library	Staff reported issues to the Troubleshooting Library successfully	Pass
Searching Module	To check whether staff, it professional and manager can search for issues.	Staff, it professional and	Staff, it professional and manager	Pass

		manager is able to search for issues.	successfully searched for issues	
Tracking Module	To check if staff and it professional can track their assigned issues.	Staff and it professional should be able to track their assigned issues.	Staff and it professional successfully track their assigned issues.	Pass
Evaluation Module	To check if managers can evaluate it professionals.	Managers should be able to evaluate it professionals.	Managers successfully evaluated it professionals	Pass
Reporting Module	To check if staff, it professionals and managers can view reports.	Staff, it professionals and managers should be able to view reports.	Staff, it professionals and managers successfully viewed reports.	Pass

5. Conclusion

The project successfully accomplished its objectives by developing a web-based TroubleShot - Troubleshooting Hub for Regal Motor Holdings. The system underwent thorough testing and demonstrated its functional capabilities.

The key advantages of this system include providing a centralized platform for IT troubleshooting, fostering knowledge sharing, expediting issue resolution, and providing data-driven insights. The digital hub ensures IT professionals can operate at peak efficiency, minimize downtime, and contribute to the overall success and competitiveness of the business. Additionally, the system enables real-time tracking, performance evaluation, and detailed reporting, significantly improving IT troubleshooting efficiency.

However, there are limitations to the project. The real-time tracking feature is currently limited to basic status updates and lacks advanced notification capabilities. The performance evaluation module, while functional, could benefit from more sophisticated analytics and integration with other IT systems. Additionally, the reporting feature could be expanded to include more customizable reporting options and advanced data visualization tools.

Future work for the project includes addressing these limitations. Implementing advanced notification systems for real-time updates and developing more comprehensive analytics for performance evaluation will enhance the system's functionality. Additionally, expanding the reporting module to include customizable reports and advanced data visualization will provide deeper insights into IT operations.

In conclusion, the development of the TroubleShot - Troubleshooting Hub is an innovative and strategic initiative that significantly enhances IT troubleshooting processes at Regal Motor Holdings. Its success demonstrates the potential of digital solutions in transforming IT operations, fostering collaboration, and improving cost-effectiveness. As the project progresses, its impact on the efficiency and effectiveness of IT troubleshooting is eagerly anticipated, with the potential to set a new standard in IT operations management.

Acknowledgement

I wish to extend my heartfelt gratitude to everyone who played a part in the successful completion of the "TroubleShot - Troubleshooting Hub" project. First and foremost, my deepest thanks go to my parents for their unwavering support, encouragement, and love. Your belief in me has been a powerful motivator throughout this journey. To my little brother, who took care of our family while I was in the university. To my housemates, thank you for your patience, understanding, and for creating a supportive environment that fostered both productivity and camaraderie. Your support has been invaluable. My sincere thanks to the lecturers of the Faculty of Computer Science and Information Technology. Your guidance, knowledge, and dedication to teaching have provided me with the foundation needed to undertake this project. A special thank you to my supervisor, whose expertise, insightful feedback, and continuous support were instrumental in shaping the direction and success of this project.

Your mentorship has been incredibly valuable. Lastly, I wish to acknowledge everyone else who, directly or indirectly, contributed to this project. Your support, whether large or small, has been crucial to its completion.

Author Contribution

This journal requires that all authors take public responsibility for the content of the work submitted for review. The contributions of all authors must be described in the following manner:

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Haikal Iskandar, Rozlini Binti Mohamed;* ***data collection:*** *Haikal Iskandar, Rozlini Binti Mohamed;* ***analysis and interpretation of results:*** *Haikal Iskandar, Rozlini Binti Mohamed;* ***draft manuscript preparation:*** *Haikal Iskandar, Rozlini Binti Mohamed. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] J. N. Luftman, P. R. Lewis, and S. H. Oldach, "Transforming the enterprise: The alignment of business and information technology strategies," IBM Systems Journal, vol. 32, no. 1, pp. 198-221, 1993.

[2] P. Kirvan and A. Zola, “troubleshooting,” *WhatIs*, May 13, 2022. https://www.techtarget.com/whatis/definition/troubleshooting

[3] M. Alemu, A. Adane, B. K. Singh, and D. P. Sharma, "Cloud-based outsourcing framework for efficient IT project management practices," International Journal of Advanced Computer Science and Applications, vol. 11, no. 9, 2020.

[4] B. Mirel, Interaction Design for Complex Problem Solving: Developing Useful and Usable Software. Morgan Kaufmann, 2004.

[5] F. Bontempi, K. Gkoumas, and S. Arangio, "Systemic approach for the maintenance of complex structural systems," Structure and Infrastructure Engineering, vol. 4, no. 2, pp. 77-94, 2008.

[6] Kislay, “What is Requirement Analysis? Tools and Techniques,” *Intellipaat*, Sep. 2, 2023. https://intellipaat.com/blog/what-is-requirement-analysis/

[7] J. Hannah, “What Is A User Interface & What Are The Key Elements?,” *CareerFoundry*, Apr. 24, 2023. https://careerfoundry.com/en/blog/ui-design/what-is-a-user-interface/