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

AITCS

Vol. 6 No. 2 (2025) 1156-1175
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Dual-Platform Automated Logistics Management System with Real-Time Tracking

Teoh Wei Jun[1], Nur Liyana Sulaiman[1*]

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

*Corresponding Author: nrliyana@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.02.062

Article Info

Received: 15 June 2025
Accepted: 20 November 2025
Available online: 30 November 2025

Keywords

Logistics, Real-time Tracking, Dual-platform, Automated, Management System

Abstract

The logistics industry depends on efficient deliveries, but Eco Haulage Sdn Bhd, a haulage company, still relies on manual processes, resulting in inefficiencies and errors. This project aims to tackle the problems by introducing a Dual-Platform Automated Logistics Management System with Real-Time Tracking. This project's objectives are to design a system using object-oriented approach, to develop the system using web and Android technology, and to test the system using functionality testing and user acceptance testing (UAT). Software Development Life Cycle methodology is adopted, with Laravel for the web platform, Flutter for the Android mobile application, and Supabase as the database. Functionality testing recorded 100% pass rate, while UAT using System Usability Scale yielded an overall score of 85.36%, indicating usability. Future enhancements include expanding support to iOS and other platforms to improve accessibility. This project enhances logistics operational efficiency and provide a replicable blueprint for small and medium-sized logistics companies.

1. Introduction

Eco Haulage Sdn Bhd, a Penang-based haulage company, currently relies on manual processes like Google Sheets and Excel for logistics management, leading to inefficiencies, errors, and delays in document generation and salary calculations. The company manually handles requests for transport (ROT) and consignment notes (CN), creating additional workload and operational confusion. To address these challenges, the solution is a Dual-Platform Automated Logistics Management System, developed using PHP Laravel for the web and Flutter for mobile, with Supabase as the central database for real-time synchronization and data storage [1]. This system automates key tasks such as document generation, order management, and salary calculations, ultimately improving accuracy and streamlining operations.

The system includes a web platform for supervisors and an Android mobile app for drivers. The web application will manage customer orders, job assignments, and real-time tracking, while the mobile app will enable drivers to receive job assignments, track their performance, and update their real-time location. By automating critical tasks, the system will reduce manual errors, streamline data entry, enhance delivery speed, and improve communication between supervisors and drivers. With the use of Supabase, data will be stored centrally, ensuring seamless synchronization across both platforms [2].

This integrated system not only improve Eco Haulage's operational efficiency but also support the company's growth and scalability in terms of streamlined operations, reduced manual workload, real-time data accessibility, and the ability to handle increased order volume without proportional increases in resources or costs. By automating repetitive tasks, the system will reduce administrative workload, enhance productivity, and

contribute to faster delivery times, which in turn will improve client satisfaction [3]. Furthermore, this project serves as a model for other small and medium-sized logistics companies, showcasing the advantages of automation and real-time tracking in optimizing business operations and fostering scalability within the logistics industry. There are three main objectives that need to be achieved, which are:

i. To analyse and design a Dual-Platform Automated Logistics Management System using an object-oriented approach.
ii. To develop a Dual-Platform Automated Logistics Management System using both Android technology and web-based approach.
iii. To test the developed system using functional testing and user acceptance testing.

The rest of the paper is organized as follows: Section 2 presents the literature review and explores existing technologies relevant to the field. Section 3 outlines the methodology, detailing each phase of the Software Development Life Cycle (SDLC), including the Analysis and Design Phase in Section 3.1 and the Implementation Phase in Section 3.2. Section 4 discusses the results and findings, followed by Section 5, which provides a conclusion summarizing the key outcomes and proposing potential improvements.

2. Literature Review

This section presents the literature review for the Dual-Platform Automated Logistics Management System, covering domain background, related terms, technology, and comparison with the existing systems.

2.1 Logistics Industry

The logistics industry manages the flow of goods, services, and information to ensure efficient, cost-effective, and timely delivery [4]. Facing challenges like rising delivery demands, supply chain visibility needs, and COVID-19, the industry increasingly adopts advanced technologies like Artificial Intelligence (AI), machine learning, and automation to enhance operations and efficiency [5].

2.2 Logistics Management System

A Logistics Management System (LMS) is software designed to optimize supply chain processes, including inventory, transportation, warehousing, and route planning [6]. With real-time tracking, LMS enhances supply chain transparency, reduces delays, and improves resource allocation to meet customer expectations in competitive markets [7]. Additionally, integrating data analytics enables better decision-making, demand forecasting, and route optimization, reducing operational costs and improving overall performance [8].

2.3 Real-Time Tracking

Real-time tracking in logistics uses Global Positioning System (GPS) and communication technologies to monitor goods and vehicles, providing continuous updates on status and arrival times, enhancing efficiency and transparency [9]. It reduces delays, prevents theft, and allows dynamic responses to disruptions, optimizing routes and improving reliability. This capability enhances inventory management, reduces costs, ensures timely deliveries, and boosts customer satisfaction and competitive advantage [10]. Therefore, logistics management system should apply the real-time tracking feature to ensure more accurate and efficient delivery processes. The developed Dual-Platform Automated Logistics Management System integrates this functionality to enable supervisors to monitor driver locations in real time via the web platform, while drivers update their status and location through the mobile application.

2.4 Comparison with the Existing System

This section compares the three existing systems with the developed system as shown in Table 1. The three existing systems being compared are Swift Logistics Haulage Management System (SLHMS) used by Swift Logistics Berhad [11], LTS e-Logistics Management System (LTSMS) used by LTS Logistics (PG) Sdn Bhd [12], and Interway's Transport Management System (ITMS) used by Interway Transport Sdn Bhd [13].

Table 1 *Comparison between existing systems and the developed system*

Features	SLHMS	LTSMS	ITMS	Developed System
Order Management	✓	✓	✓	✓
Order Status Tracking	✓	✓	✓	✓
Document Generation	X	X	✓	✓

Table 1 *(cont)*

Features	SLHMS	LTSMS	ITMS	Developed System
Job Assignment	X	X	X	✓
Platform	Web	Web	Web	Web & Android
Data Analytics	X	X	X	✓
Real-Time Location Tracking	X	X	X	✓
Driver Performance Monitoring	X	X	X	✓
Attendance Record	X	X	X	✓
Driver Salary Calculation	X	X	X	✓
Salary Record	X	X	X	✓
Digital document Submission	✓	X	✓	✓

Table 1 highlights that the developed system surpasses the three existing systems in features and functionality. While all systems support order management and status tracking, only ITMS and the developed system can generate documents like ROT. Unlike the others, the developed system is available on both web and Android platforms. It also uniquely includes advanced features such as job assignment, data analytics, real-time tracking, driver performance monitoring, attendance tracking, and salary calculations. These capabilities make the developed system a more comprehensive and efficient logistics management solution.

3. Methodology

This section outlines the methodology for developing the Dual-Platform Automated Logistics Management System. In this project, the SDLC model is adopted to guide the development of the Dual-Platform Automated Logistics Management System for Eco Haulage. This model is selected for its clarity and ease of implementation, especially when the project requirements are relatively well-defined from the start. It also allows for structured project tracking and milestone-based progress evaluation [14].

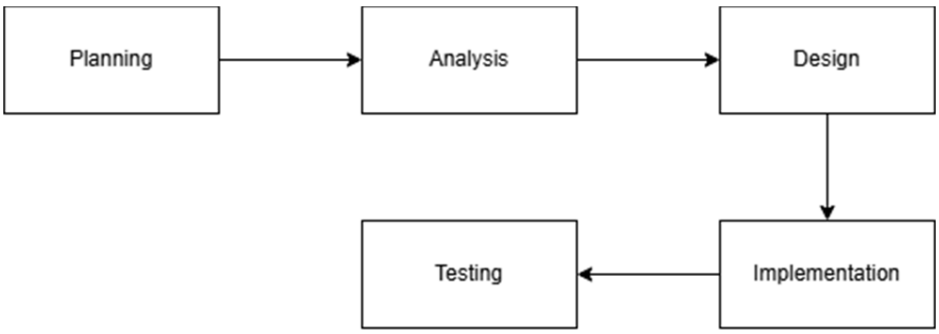


Fig. 1 *SDLC*

The SDLC is a systematic process used to develop software through a structured sequence of phases. The traditional SDLC model includes five main phases: Planning, Analysis, Design, Implementation and Testing [15]. This linear and disciplined approach ensures that each stage is completed before moving on to the next, providing clarity, control, and documentation at each step of the development process [16].

The use of a SDLC model ensures that the development process is methodical, reducing risks and improving the overall reliability of the software. It also supports documentation and traceability, which are important for long-term maintenance and upgrades [17]. Therefore, the SDLC model was utilized in this project to guide the development of the system, ensuring a structured and efficient workflow throughout each phase.

3.1 Analysis and Design Phase

This phase covers the system's analysis and design, starting with Requirement Analysis, which defines the system's functional and non-functional requirements. Unified Modeling Language (UML) diagrams such as use case diagram, activity diagram, and sequence diagram are used to represent system structure, workflows, and

interactions. This phase also outlines the general system architecture, detailing component interactions. Class Diagram is used for illustrating the database design for organizing data. This phase also involved Interface Design, highlighting user-friendly visual and interactive elements.

Requirements analysis is essential for identifying, defining, and documenting stakeholder needs to ensure the system meets its purpose. It involves gathering requirements through methods like interviews or surveys and organizing them into functional and non-functional categories using structured formats like use case specifications and requirement lists [18]. This process minimizes ambiguity, aligns the system with user expectations, and resolves potential conflicts early. Tools like Use Case, Activity, and Sequence Diagrams help visualize workflows, while Requirements Specification Documents formalize needs [19]. Effective analysis reduces risks, scope creep, delays, and costs, increasing project success [20].

Functional requirements analysis identifies and defines the specific functions a system must perform to meet user and business needs. It outlines the system's features and capabilities [19]. Table 2 lists the web application's functional requirements, while Table 3 details those for the mobile application.

Table 2 *Functional Requirements*

Modules	Requirements	Actor
Login	The user must be able to log into the system, including entering credentials (email and password), verifying credentials against the database, and handling authentication failures. Additional security measures include password recovery options. Supervisor can only login on website while driver can only login on mobile app.	Supervisor Driver
View Data Analytics	Supervisor has access to a data analytics dashboard with key performance metrics. They can view detailed reports on driver performance, track job completion, and access real-time updates on driver activities. The system also provides visualizations such as graphs and charts to present the analytics data clearly.	Supervisor
	Drivers can track their own performance metrics, including attendance, job completion, and driving time. This module provides a visual representation of the driver's work performance over time.	Driver
Manage Users and Drivers	Supervisor can manage user and driver accounts, including viewing, adding, editing, or deleting user and driver records. This module allows the supervisor to update critical user information.	Supervisor
Manage Customer	Supervisor can manage customer records, including viewing, adding, editing, and deleting customer details. They can validate customer information before saving it in the system.	Supervisor
Manage Job and Zone	Supervisor assigns jobs to available drivers and manage zones associated with each job. They ensure the right driver is assigned to the correct job based on availability. Once the delivery started, supervisors can track the real-time location of drivers via a map.	Supervisor
	Drivers access their job dashboard, which lists all assigned jobs, along with details such as the job to be delivered, time of delivery, and job-specific information. The system notifies drivers of new job assignments and allows them to start jobs. Real-time location tracking begins when the job starts.	Driver
Manage Order	Supervisor oversees order creation, assignment, and tracking. They can add new orders, view and edit order details, and generate required documents such as ROT and CN. They also receive real-time updates on the status of each order.	Supervisor

Table 2 *(cont)*

Modules	Requirements	Actor
Manage Driver Attendance	Supervisor is responsible for managing the attendance records of drivers. This includes ensuring that attendance data is accurate and valid, with QR code functionality integrated for drivers to mark their attendance.	Supervisor
	Drivers mark their attendance using QR codes generated by the supervisor. The system ensures that the attendance data is valid and accurate, preventing the use of expired or invalid QR codes for attendance marking.	Driver
Manage Driver Salary	Supervisor can view and edit driver salary records, as well as calculate salaries based on driver attendance and performance. They ensure all salary data is validated for consistency and accuracy before finalization.	Supervisor
Edit Profile	Supervisor can update their personal information, including contact details. The system validates all changes to ensure they are correct and saves updates in real time.	Supervisor
	Driver can view their own profile details and change the password of the account, assuring the account is secure.	Driver

Table 2 outlines the key functional requirements for both Supervisor and Driver roles within the system. Supervisors are responsible for overseeing logistics operations, including managing driver and user accounts, maintaining customer records, and assigning jobs to drivers. The system equips supervisors with tools to monitor real-time driver locations, manage attendance and salary records, and generate essential documents such as the Request of Transport (ROT) and Consignment Note (CN). Additionally, supervisors can access a data analytics dashboard for insights into driver performance and job completion, enabling informed decision-making. Profile management features ensure supervisors can maintain accurate personal information.

Drivers, on the other hand, manage their individual tasks through the system, such as logging in, marking attendance using Quick-Response (QR) code validation, viewing assigned jobs, and updating real-time locations during deliveries. The driver dashboard provides detailed job information and notifications for new assignments. Drivers can also track their performance and update their personal profile, with all changes validated and saved in real time. The continuous update of location data allows for accurate tracking by supervisors throughout the delivery process.

Non-functional requirements analysis focuses on identifying and evaluating the quality attributes and constraints a system must meet, such as reliability, security, scalability, and performance [18]. Unlike functional requirements, which define what the system does, non-functional requirements focus on how it performs, ensuring it meets user and business expectations [21].

Table 3 *Non-Functional Requirements*

Modules	Requirements
Performance	Ensures the system remains responsive during peak usage.
Scalability	Guarantees the system can grow alongside increasing order and driver data.
Availability	Provides high availability to ensure real-time tracking and order management are always accessible.
Security	Ensures data confidentiality and protection against unauthorized access.
Data Integrity	Prevents data corruption and maintains integrity during operations like job assignment and salary calculation.
Usability	Enhances user experience for drivers using the mobile app.
Compatibility	Ensures cross-platform functionality for both web and mobile applications.
Maintainability	Facilitates seamless updates without significantly impacting operations.

Table 3 outlines the non-functional requirements crucial for the success of the system. These requirements address essential aspects such as performance, which ensures the system remains responsive under heavy use,

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1156-1175

and scalability, which guarantees that the system can grow alongside increasing data. Availability emphasizes the need for high uptime, ensuring that real-time tracking and order management are consistently accessible. Security is a priority to protect sensitive data from unauthorized access, while data integrity ensures accuracy and consistency during operations. Usability focuses on creating an intuitive experience for drivers using the mobile app, and compatibility ensures seamless cross-platform functionality between the web and mobile applications. Lastly, maintainability facilitates smooth updates and system enhancements without significantly disrupting operations. These non-functional requirements together ensure that the system is efficient, secure, and user-friendly, while also being scalable and easy to maintain.

A use case diagram visually represents a system's functional requirements by illustrating interactions between actors such as users or external systems and use cases such as system functionalities [22]. Its primary purpose is to capture high-level system requirements, ensuring stakeholders and developers share a common understanding of the system's scope. The diagram helps in creating detailed specifications, workflows, and test cases, aligning the system design with project objectives. Its simplicity and user-centric focus make it an essential tool for guiding system design and implementation, as shown in Fig. 2.

Fig. 2 *Use Case Diagram*

This project involves two primary actors: the Supervisor, who manages logistics via the web application, and the Driver, who uses the mobile app to update job statuses, locations, and attendance. The use case diagram

highlights key use cases such as login, view data analytics, edit profile, manage job and zone, manage driver attendance record, manage users and drivers, manage order info, manage driver salary record and manage customer info. Relationships between actors and use cases, including associations and extend links, ensure a modular and scalable design, accommodating the system's complexities.

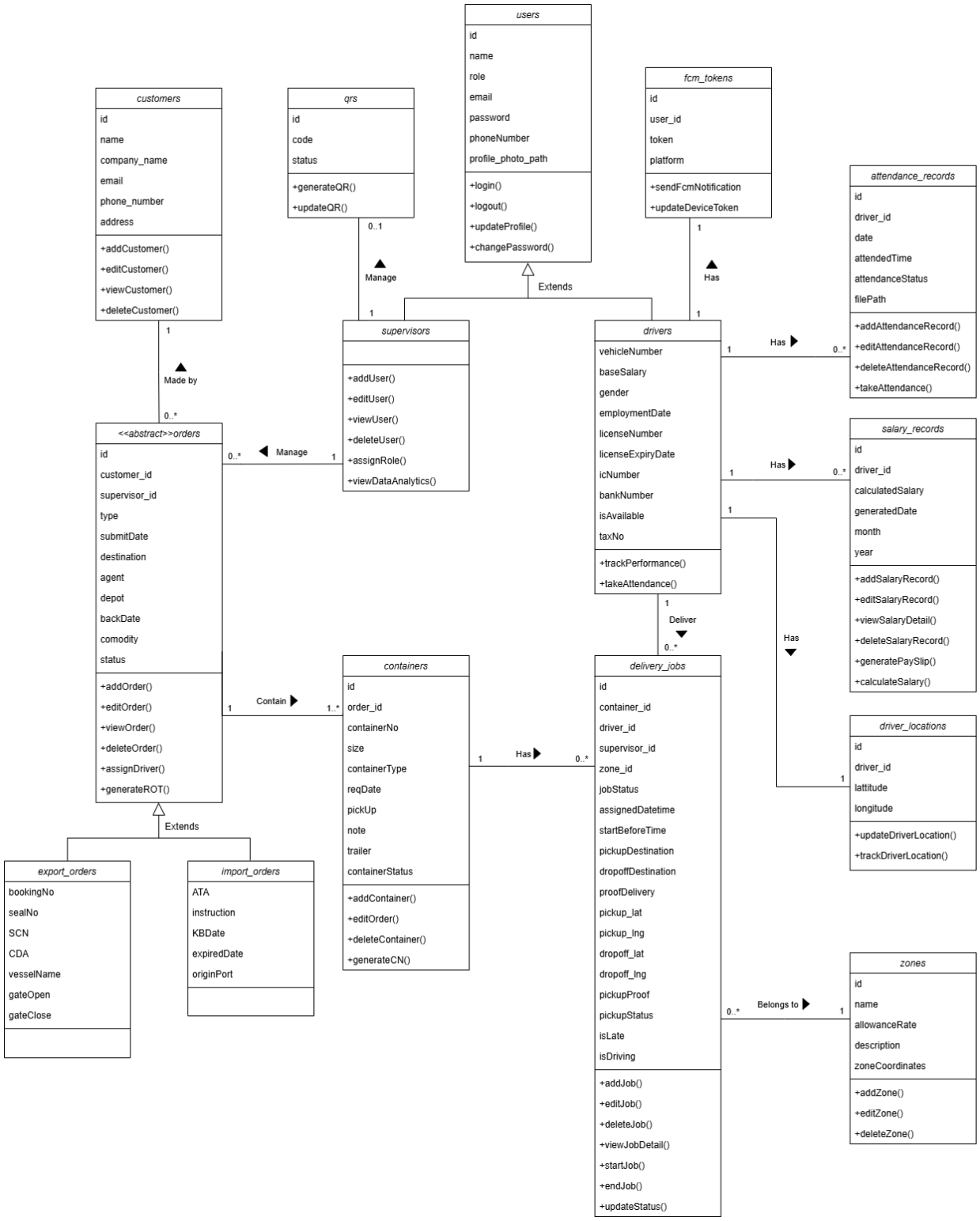


Fig. 3 *Class Diagram*

Class diagram shows in Fig. 3 visually represents the relationships, structures, and interactions between classes, detailing their attributes (properties) and methods (behaviors). In this project, it comprises 15 interconnected classes with core classes such as supervisors, drivers, orders, containers, delivery_jobs, zones,

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1156-1175

customers, attendance_records, salary_records, and driver_locations. It illustrates key relationships and multiplicities (e.g., one-to-many, many-to-one), enabling a structured representation of real-world logistics entities and operations. Serving as a blueprint for software design and implementation, the class diagram helps organize system components logically, ensuring scalability, maintainability, and effective interaction between objects and classes [23].

For the design phase, general system architecture is the high-level implementation of a system, outlining key components, their relationships, and interactions to meet system objectives. It provides a blueprint for development, ensuring cohesion and clarity among stakeholders, developers, and designers [24]. Fig. 4 shows the 3-tier architecture of the system.

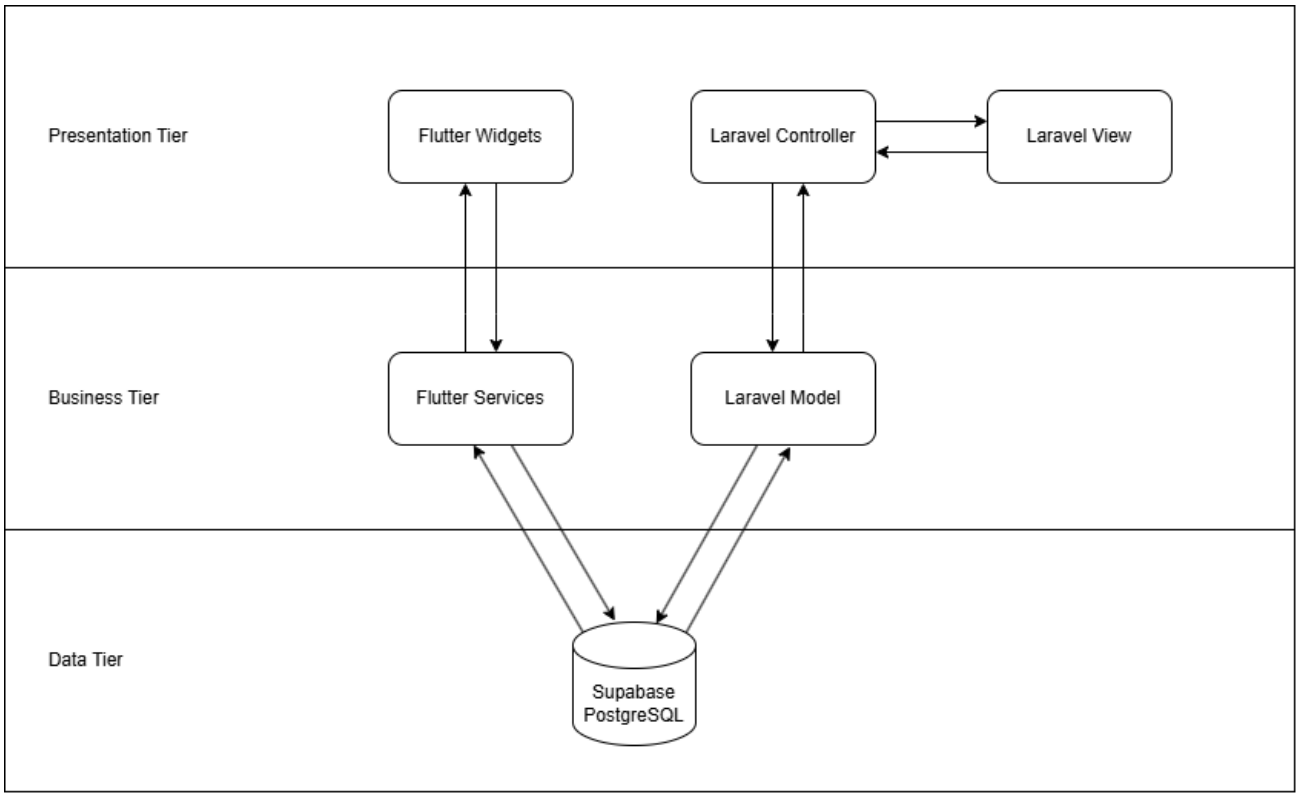


Fig. 4 3-*Tier System Architecture*

The 3-tier architecture divides applications into three tiers: the presentation tier (UI), the business tier (business logic), and the data tier (database). Each tier handles distinct responsibilities, ensuring separation of concerns, scalability, and maintainability. The presentation tier interacts with users, the business tier processes requests and communicates with the data tier, which stores and manages data. In this project, the 3-Tier architecture enhances system efficiency by clearly separating the user interface, business logic, and data management, allowing for easier maintenance.

3.2 Implementation Phase

The implementation phase focuses on translating the system design into a functional application. It covers the development of core modules such as login, view data analytics, edit profile, manage driver attendance, manage job and zone, manage driver salary, manage user and driver, manage order, and manage customer. Each module is developed using appropriate tools and technologies such as Laravel for web-based module, Flutter for module involving website, Supabase act as the central database, and Google Map Application Programming Interface (API) for real-time tracking feature.

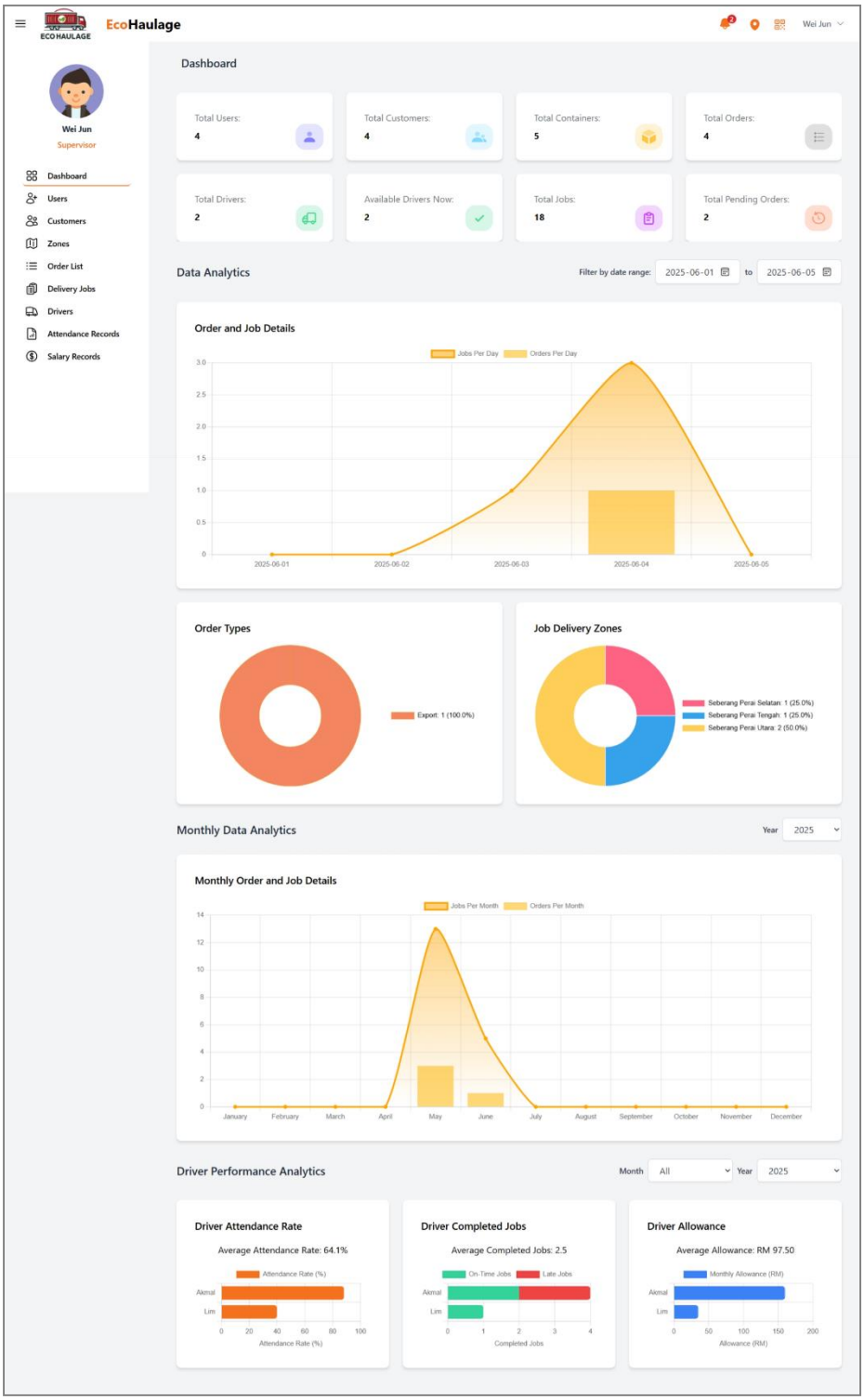


Fig. 5 *Website Dashboard Interface*

Fig. 5 shows the dashboard interface for the website, which serves as a centralized view of key operational data within the system. The dashboard displays summarized statistics including the total number of customers, users, jobs, and orders to provide an immediate overview of system activity. It features a bar and line chart illustrating the daily number of jobs and orders, along with two doughnut charts that visualize the distribution of order types (Import and Export) and the count of jobs by zone. Additionally, a monthly bar and line chart presents broader trends in job and order volumes over time. The dashboard also includes a driver performance section highlighting attendance rate, job completion rate, and total allowance, all of which are filterable by specific criteria such as date range, zone, and driver. This interactive dashboard enables supervisors to monitor logistics operations efficiently and make data-driven decisions.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1156-1175

```php
// FOR DATE RANGE ORDER AND JOB DETAILS
$startDate = now()->startOfMonth()->format(format: 'Y-m-d'); // default: start of current month
$endDate = now(tz: 'Asia/Kuala_Lumpur')->endOfDay();
// Get daily orders
$ordersRaw = Order::whereBetween(column: 'created_at', values: [$startDate, $endDate])
    ->selectRaw(expression: 'DATE(created_at) as day, COUNT(*) as total')
    ->groupBy(groups: 'day')
    ->pluck(column: 'total', key: 'day')
    ->toArray();

// Get daily completed jobs
$jobsRaw = DeliveryJob::whereBetween(column: 'created_at', values: [$startDate, $endDate])
    ->selectRaw(expression: 'DATE(created_at) as day, COUNT(*) as count')
    ->groupBy(groups: 'day')
    ->pluck(column: 'count', key: 'day')
    ->toArray();

// Generate date range
$dates = [];
$start = Carbon::parse(time: $startDate);
$end = Carbon::parse(time: $endDate);
while ($start <= $end) {
    $key = $start->format(format: 'Y-m-d');
    $dates[] = $key;
    $start->addDay();
}

// Fill missing dates
$ordersPerDay = [];
$jobsPerDay = [];
foreach ($dates as $day) {
    $ordersPerDay[$day] = $ordersRaw[$day] ?? 0;
    $jobsPerDay[$day] = $jobsRaw[$day] ?? 0;
}

$orderTypes = Order::whereBetween(column: 'created_at', values: [$startDate, $endDate])
    ->selectRaw(expression: "type, COUNT(*) as count")
    ->groupBy(groups: 'type')
    ->pluck(column: 'count', key: 'type'); // returns ['Export' => 15, 'Import' => 10]

$zoneCounts = DeliveryJob::join(table: 'zones', first: 'delivery_jobs.zone_id', operator: '=', second: 'zones.id')
    ->whereBetween(column: 'delivery_jobs.created_at', values: [$startDate, $endDate])
    ->selectRaw(expression: 'zones.name as zone_name, count(*) as count')
    ->groupBy(groups: 'zones.name')
    ->pluck(column: 'count', key: 'zone_name'); // key = zone name, value = count
```

Fig. 6 *Website Orders and Jobs Analytics Code Segment*

Fig. 6 shows the source code of the orders and jobs analytics. The system fetches order and delivery job records within a specified date range using Laravel's query builder and Carbon for time manipulation. It groups and counts the records by day, filling in gaps for dates with no activity to maintain chart continuity. Order types are also categorized (e.g., Export or Import), and delivery jobs are joined with zone data to show distribution across operational regions. These data points are prepared and structured for visualization, helping supervisors identify workload patterns, monitor zone efficiency, and assess overall operational health.

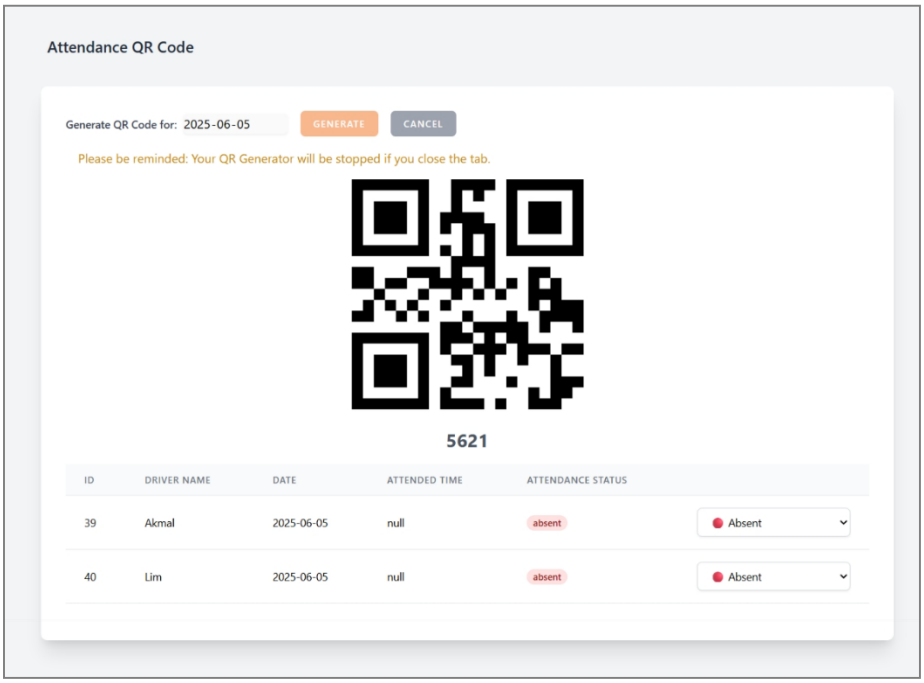


Fig. 7 *Website Attendance QR Interface*

The Manage Driver Attendance Module enables supervisors to monitor and control driver attendance records efficiently through the website, while drivers can conveniently mark their attendance using a QR code system on the mobile application as shown in Fig. 7. This dual-platform approach ensures that attendance data is accurately recorded and synchronized, minimizing manual effort and errors. The system includes features for creating, updating, and deleting attendance records, generating daily QR codes, and marking attendance in real time by scanning the code on the mobile app. This integration ensures real-time validation and a seamless attendance-taking process across the platform.

```php
public function show(): View
{
    return view(view: 'qr-show');
}


1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comment | ×
public function generate(Request $request): View
{
    $date = request(key: 'date');
    $exists = \App\Models\AttendanceRecord::whereDate(column: 'date', operator: $date)->exists();

    if (!$exists) {
        $drivers = \App\Models\Driver::all();
        foreach ($drivers as $driver) {
            \App\Models\AttendanceRecord::create(attributes: [
                'driver_id' => $driver->id,
                'date' => $date,
                'attendanceStatus' => 'absent',
            ]);
        }
    }

    // Generate QR code with text "Hello, Laravel 11!"
    $num = rand(min: 1000, max: 9999);
    $qrCode = QrCode::size(pixels: 300)->generate(text: $num);

    if (Qr::count() == 0) {
        $qr = Qr::create(attributes: [
            'code' => $num,
            'status' => 'active',
        ]);
    } else {
        $qr = Qr::first();
        $qr->update(attributes: [
            'code' => $num,
            'status' => 'active',
        ]);
    }

    return view(view: 'qrcode', data: compact(var_name: 'qrCode', var_names: 'num'));
}


1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comment | ×
public function cancel(): JsonResponse|mixed
{
    $qr = Qr::first();
    $qr->update(attributes: [
        'status' => 'inactive',
    ]);

    return response()->json(data: ['status' => 'inactive']);
}

0 references | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comment | ×
public function check(): JsonResponse|mixed
{
    if (Qr::count() == 0) {
        return;
    } else {
        $qr = Qr::first();
        if ($qr->status == 'active') {
            return response()->json(data: ['status' => 'active']);
        } else {
            return response()->json(data: ['status' => 'inactive']);
        }
    }
}
```

Fig. 8 *Website Attendance QR Code Segment*

Fig. 8 shows the implementation of the attendance QR feature. The system supports the generation of a unique QR code each day through the website. When a supervisor generates the code, the system first checks if attendance records exist for the selected date. If not, it auto-generates records for all drivers with a default status of 'absent'. A random 4-digit code is then generated and embedded into the QR code, which is displayed on the web view. This code is stored in the database with an 'active' status. The QR code's status can be monitored and updated in real time, allowing the supervisor to deactivate it when necessary. This functionality ensures that QR codes are valid only during specific time windows, enhancing control and preventing misuse.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1156-1175

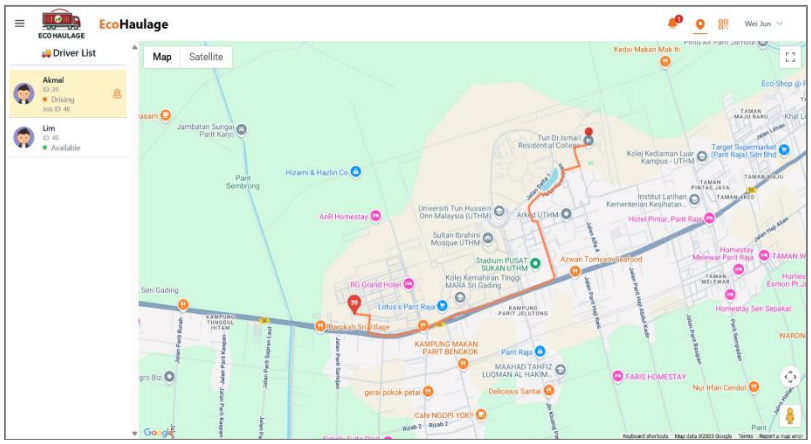


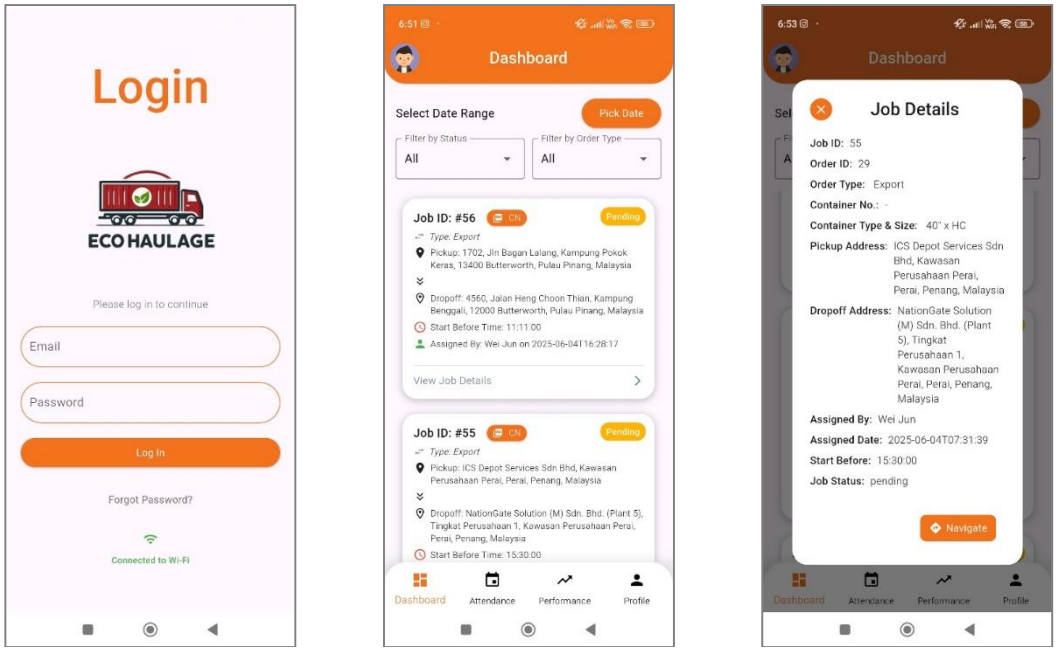
Fig. 9 *Website Driver Real-Time Tracking Interface*

```
async function fetchDriverLocations() {
  const {
    data,
    error
  } = await supabase
    .from('driver_locations')
    .select(`*,driver:driver_id (isAvailable)`)
    .eq('driver.isAvailable', false); // Only where isAvailable is false
  if (error) {
    console.error('Error fetching driver locations:', error);
    return;
  }
  // Filter and place markers
  data.forEach(driver => {
    if (driver.driver?.isAvailable === false) {
      updateDriverMarker(driver);
    }
  });
}

// Update or Create a Marker for a Driver
function updateDriverMarker(driver) {
  const {
    driver_id,
    latitude,
    longitude
  } = driver;
  let driverName = '';
  if (driver_id !== undefined && driver_id !== null) {
    let drivers = @json(value: $drivers);
    const driverData = drivers.find(driver => driver.user_id == driver_id);

    if (driverData && driverData.user) {
      driverName = driverData.user.name;
    } else {
      console.warn("User data not found for driver ID:", driver_id);
    }
  }
  if (driverMarkers[driver_id]) {
    // Update existing marker position
    driverMarkers[driver_id].setPosition(new google.maps.LatLng(latitude, longitude));
  } else {
    // Create a new marker
    driverMarkers[driver_id] = new google.maps.Marker({
      position: new google.maps.LatLng(latitude, longitude),
      map: map,
      title: `Driver ${driver_id} - ${driverName}`,
      icon: {
        scaledSize: new google.maps.Size(40, 40), // resize icon
        labelOrigin: new google.maps.Point(20, 12),
      },
      label: {
        text: `${driver_id}`, // or full name if available
        color: "#ffffff",
        fontWeight: "bold",
        fontSize: "12px",
      },
      animation: google.maps.Animation.DROP, // drop effect on marker load
    });

  }
}

// Subscribe to Supabase Real-time Updates
function subscribeToLiveUpdates() {
  console.log("tracking subscribed");
  supabase
    .channel('driver_locations')
    .on('postgres_changes', {
      event: '*',
      schema: 'public',
      table: 'driver_locations'
    }, payload => {
      console.log('🔄 Location updated:', payload.new);
      updateDriverMarker(payload.new);
    })
    .subscribe();
}
```

Fig. 10 *Website Fetch Driver Location Code Segment*

Supervisors can view drivers' real-time locations on the website, the interface can be seen in Fig. 9 while Fig. 10 shows the implementation of fetching driver location. The system uses Supabase to fetch driver_locations

where the driver is marked as unavailable (actively on a job). Each driver's position is plotted as a marker on Google Maps with details such as their ID and name. The system intelligently updates existing markers or creates new ones as needed. This functionality enhances monitoring and operational efficiency by giving supervisors real-time visibility into their fleet's whereabouts.



(a) **(b)** **(c)**

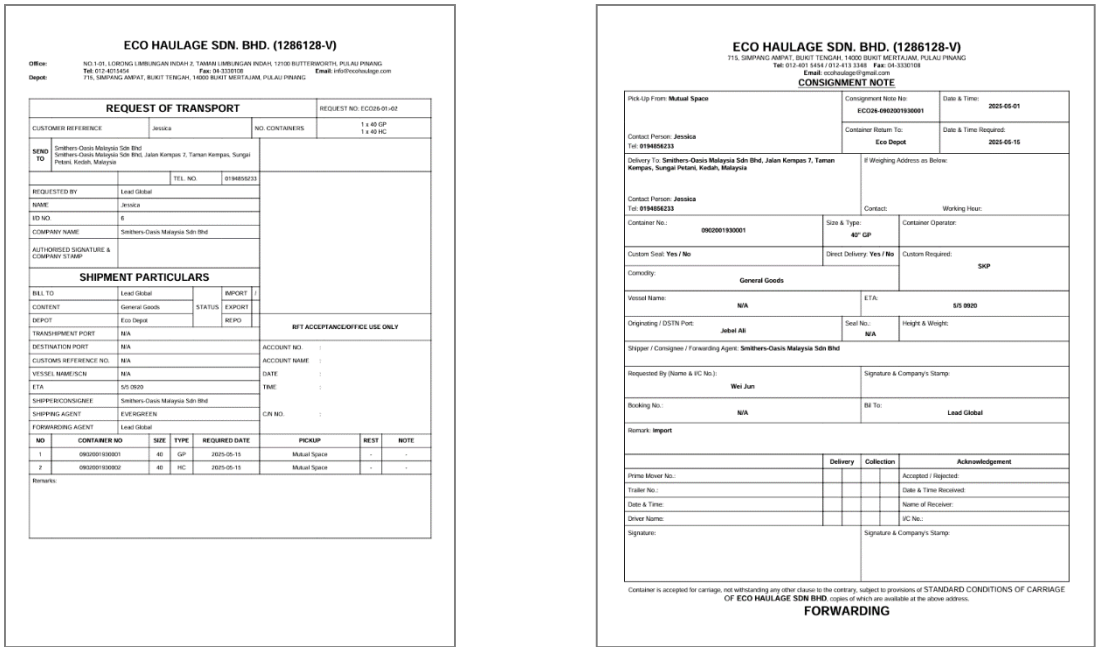
Fig. 11 *(a) Mobile Login Interface; (b) Mobile Job Dashboard Interface; (c) Mobile View Job Details Interface*

```
Future<List<dynamic>> fetchDeliveryJobs() async {
  try {
    final token = await getToken();
    if (token != null) {
      // Fetch all jobs with related data
      final response = await Supabase.instance.client
          .from('delivery_jobs')
          .select('*, supervisor_id(name), container_id(id, containerType, size, containerNo, order_id, order_id(id,   type))')
          .eq('driver_id', token)
          .order('assignedDatetime', ascending: false);

      List<Map<String, dynamic>> jobs = List<Map<String, dynamic>>.from(response);
      // Filter by date range
      if (_startDate != null && _endDate != null) {
        jobs = jobs.where((job) {
          final date = DateTime.tryParse(job['assignedDatetime']);
          return date != null &&
              date.isAfter(_startDate!.subtract(const Duration(days: 1))) &&
              date.isBefore(_endDate!.add(const Duration(days: 1)));
        }).toList();
      }

      // Filter by jobStatus
      if (_selectedStatus != null && _selectedStatus != 'All') {
        final dbStatus = mapStatusToDbValue(_selectedStatus!);
        jobs = jobs.where((job) => job['jobStatus'] == dbStatus).toList();
      }

      // Filter by order type (nested in container_id -> order_id -> type)
      if (_selectedType != null && _selectedType != 'All') {
        jobs = jobs.where((job) {
          final container = job['container_id'];
          final order = container?['order_id'];
          final orderType = order?['type'];
          return orderType == _selectedType;
        }).toList();
      }
      return jobs;
    } else {
      throw Exception("User token is missing.");
    }
  } catch (e) {
    print("Error fetching delivery jobs: $e");
    return [];
  } finally {

  }
}
```

Fig. 12 *Mobile Job Dashboard Code Segment*

On the mobile application, drivers interact with the Job Dashboard to view their assigned jobs. The mobile client fetches job data from Supabase, retrieving nested relationships that include supervisor information and container details, as well as the type of order associated with the container. Drivers are able to filter their job

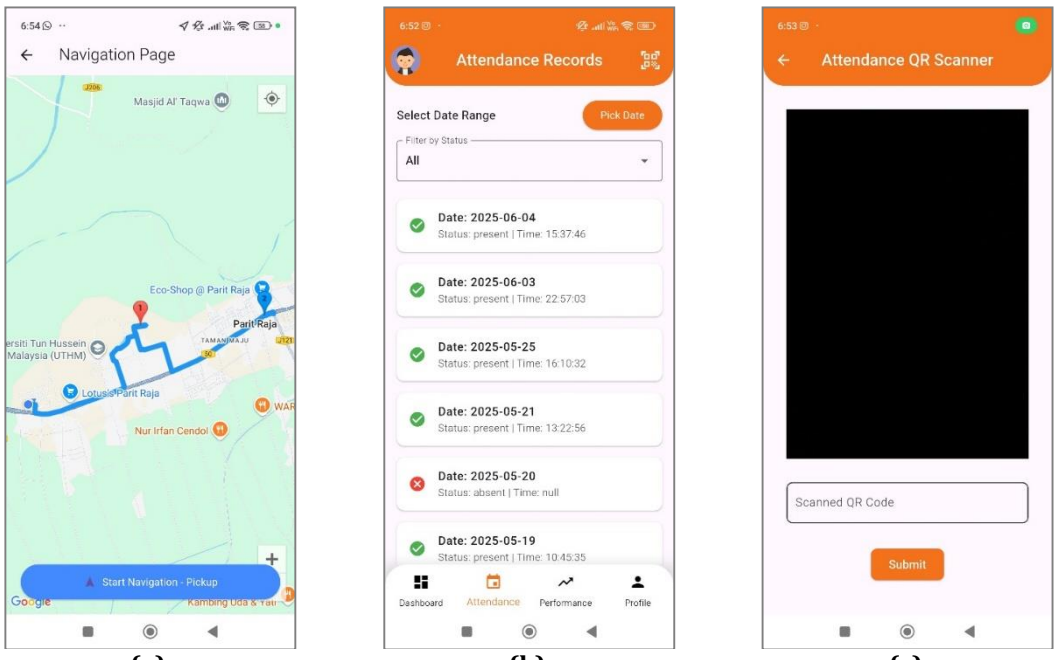
history by date, job status, and order type. The filtering logic is performed on the client side after all jobs are retrieved, providing flexible querying capabilities without requiring complex backend operations. Fig. 11 (b) shows the mobile job dashboard interface whiel Fig. 12 shows the implementation of the mobile job dashboard.



(a) **(b)**

Fig. 13 *(a) Request of Transport (ROT) Document; (b) Consignment Note (CN) Document*

The Request of Transport (ROT) and Consignment Note (CN) are essential logistics documents generated by the system to support haulage operations. The ROT and CN are shown in Fig. 13 (a) and (b) respectively. The ROT document is created by supervisors to formally assign and detail transportation tasks, including pickup and delivery information. Once a job is assigned, the generated CN is required to be printed out for the driver and can be viewed by the driver via the mobile application, serving as proof of delivery with key details such as cargo type, destination, and delivery confirmation. These documents help ensure proper documentation, tracking, and accountability throughout the transport process.



(a) **(b)** **(c)**

Fig. 14 *(a) Mobile Navigation Interface; (b) Mobile Attendance List Interface; (c) Mobile Scan Attendance Interface*

```dart
void _startLiveLocationUpdates() {
  _locationUpdateTimer =
      Timer.periodic(const Duration(seconds: 5), (timer) async {
    final position = await Geolocator.getCurrentPosition();
    final lat = position.latitude;
    final lng = position.longitude;
    setState(() {
      _currentPosition = LatLng(lat, lng);
    });
    final token = await storage.read(key: 'user_token');
    await supabase.from('driver_locations').upsert({
      'driver_id': token,
      'latitude': lat,
      'longitude': lng,
      'updated_at': DateTime.now().toIso8601String(),
    }, onConflict: 'driver_id');
    // ✅ Check distance to pickup
    if (_pickupPoint != null || _dropoffPoint != null) {
      bool isNearPickup = false;
      bool isNearDropoff = false;
      if (_pickupPoint != null) {
        double distanceToPickup = Geolocator.distanceBetween(
          lat,
          lng,
          _pickupPoint!.latitude,
          _pickupPoint!.longitude,
        );
        if (distanceToPickup <= 50) {
          isNearPickup = true;
        }
      }
      if (_dropoffPoint != null) {
        double distanceToDropoff = Geolocator.distanceBetween(
          lat,
          lng,
          _dropoffPoint!.latitude,
          _dropoffPoint!.longitude,
        );
        if (distanceToDropoff <= 50) {
          isNearDropoff = true;
        }
      }
      setState(() {
        _isNearPickup = isNearPickup;
        _isNearDropoff = isNearDropoff;
      });
    }
  });  // Timer.periodic
}
```

(a)

```dart
void _onQRViewCreated(QRViewController controller) {
  this.controller = controller;
  controller.scannedDataStream.listen((scanData) {
    setState(() {
      _textController.text = scanData.code ?? '';
    });
    controller.pauseCamera(); // Pause after scanning
    _submit();
  });
}

Future<void> _submit() async {
  scannedText = _textController.text.trim();
  if (await attendanceService.checkQrStatus(scannedText)) {
    int attendanceResult = await attendanceService.markAttendance();
    print(attendanceResult);
    if (attendanceResult == -1) {
      _showErrorDialog('Record not found.');
    } else if (attendanceResult == 1) {
      _showSuccessDialog("Attendance taken successfully.");
    }
  } else {
    _showErrorDialog('Invalid QR code. Please try again.');
  }
}

void _showErrorDialog(String text) {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text('Error'),
      content: Text(text),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context);
            // Resume the camera after closing the dialog
            controller?.resumeCamera();
          },
          child: const Text('OK'),
        ),
      ],
    ),
  );
}
```

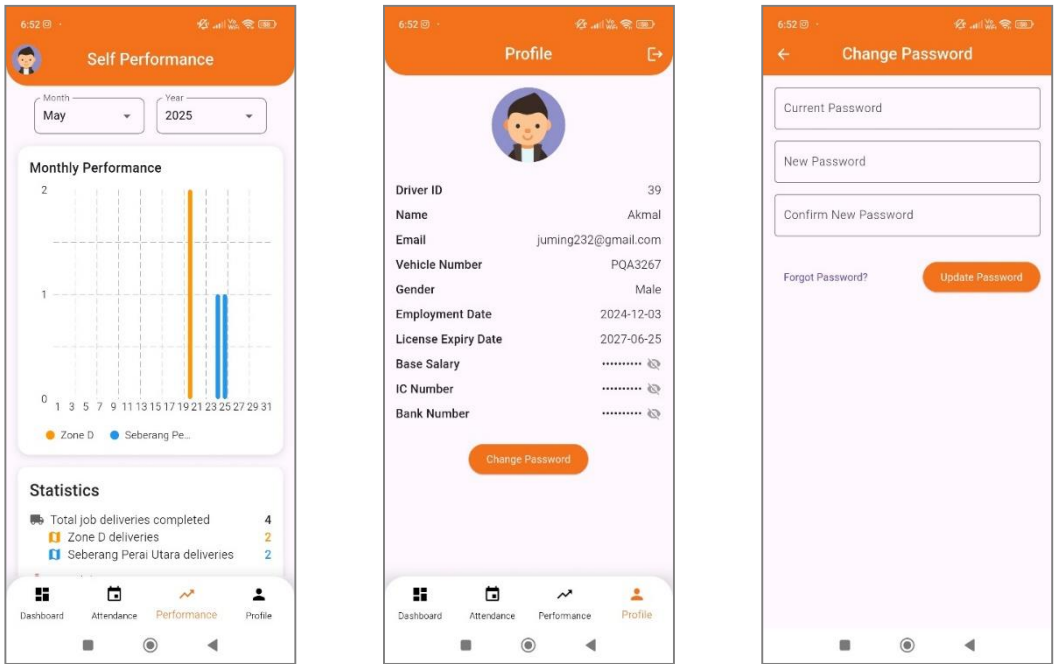
(b)

Fig. 15 *(a) Mobile Update Driver Location Code Segment; (b) Mobile QR Take Attendance Code Segment*

Fig. 15 (a) shows the source code for updating the driver location. The mobile app automatically updates the driver's location every 5 seconds using the device's GPS data. It uploads the current latitude and longitude to Supabase via an upsert operation to ensure a single active location record per driver. The system also checks the proximity of the driver to the pickup and drop-off points. When within 50 meters, the system flags the driver as "near," aiding in validating delivery activity and enhancing tracking accuracy for both supervisors and clients. The interface can be see in Fig.14 (a).

Fig. 15 (b) shows implementation of the QR take attendance feature. The driver app uses a built-in QR scanner to detect and decode the QR code displayed by the supervisor, the scanner interface can be seen in Fig. 14 (c). Upon scanning, the scanned text is temporarily stored and processed. The app verifies the QR code's validity by checking its active status with the backend. If valid, it attempts to mark attendance for the logged-in driver. Depending on the backend's response, the app either displays a success message indicating that attendance was recorded or shows an error message if no matching record is found. If the QR code is inactive or incorrect, the driver is prompted to try again. This seamless interaction ensures secure, efficient, and real-time attendance submission directly from the driver's mobile device. Attendance interface can be seen in Fig. 14 (b).

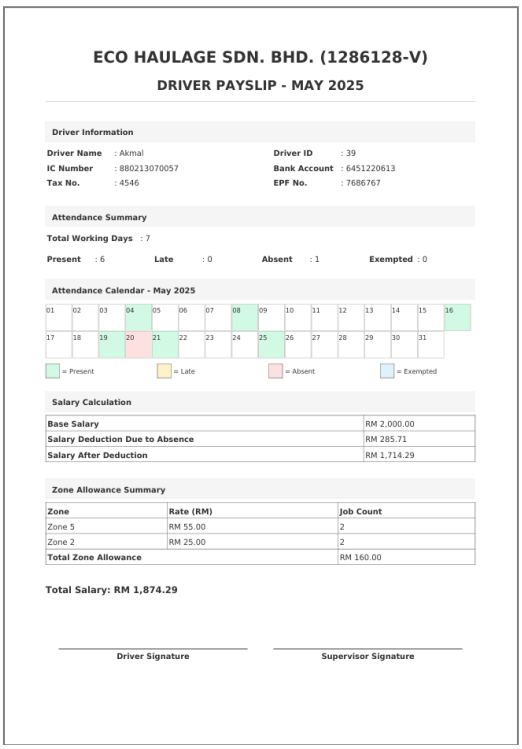
Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1156-1175



(a) **(b)** **(c)**

Fig. 16 *(a) Mobile Self-Performance Interface; (b) Mobile Profile Interface; (c) Mobile Change Password Interface*

Fig. 16 illustrates three key mobile interfaces designed for driver interaction within the system. In (a) the Mobile Self-Performance Interface, users can filter their performance data by month or year, with a bar chart displaying the number of completed jobs categorized by different zones, enabling drivers to visually track and assess their performance over time. (b) The Mobile Profile Interface presents detailed driver information, such as name, contact details, and assigned vehicle, along with a convenient "Change Password" button for quick access to account security settings. (c) The Mobile Change Password Interface allows users to securely update their credentials by entering their current password, a new password, and confirming the new password, ensuring proper validation and account protection.



ECO HAULAGE SDN. BHD. (1286128-V)

DRIVER PAYSLIP - MAY 2025

Driver Information

Driver Name	: Akmal	**Driver ID**	: 39
IC Number	: 880213070057	**Bank Account**	: 6451220613
Tax No.	: 4546	**EPF No.**	: 7686767

Attendance Summary

Total Working Days : 7

Present : 6 **Late** : 0 **Absent** : 1 **Exempted** : 0

Attendance Calendar - May 2025

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

= Present = Late = Absent = Exempted

Salary Calculation

Base Salary	RM 2,000.00
Salary Deduction Due to Absence	RM 285.71
Salary After Deduction	RM 1,714.29

Zone Allowance Summary

Zone	**Rate (RM)**	**Job Count**
Zone 5	RM 55.00	2
Zone 2	RM 25.00	2
Total Zone Allowance		RM 160.00

Total Salary: RM 1,874.29

Driver Signature **Supervisor Signature**

Fig. 17 *Driver Pay Slip Document*

The system also features an automated payslip generation function, which creates a detailed summary of the driver's monthly performance as shown in Fig. 17. This includes attendance breakdowns (e.g., present, absent, late), job statistics per zone, total allowances, and deductions due to absences. The final salary is computed by deducting a proportion of the base salary based on absent days and adding allowances for completed jobs in various zones. The data is then compiled into a structured PDF payslip with visual calendar representation and downloadable format for transparency and record-keeping.

4. Result and Discussion

This section presents the outcomes of the system testing and evaluates its performance in terms of functionality and user satisfaction. It includes functionality testing to verify that all nine modules operate as intended and user acceptance testing to assess the system's usability and effectiveness from the end-user perspective. The results are analyzed to confirm that the system meets its objectives and provides a reliable and user-friendly experience.

4.1 Functionality Testing

Functionality testing was conducted to ensure that each module of the system performs according to the specified requirements. Test cases were designed and executed to validate both expected behaviors and exception handling. The results confirm that all core functionalities, including login, order management, driver tracking, and data analytics, operate correctly and reliably as shown in Table 4.

Table 4 *Functionality Testing*

No.	Module Name	Number of Test Cases	Key Features Tested	Test Result
1	Login Module	7	Successful/failed login, password reset, session management, error handling, role-based access	All Passed
2	View Data Analytics Module	10	Access control, date filtering, data accuracy, real-time updates, visualization (charts/graphs)	All Passed
3	Edit Profile Module	7	Profile editing, input validation, password change, access control	All Passed
4	Manage Driver Attendance Module	6	Supervisor record management, QR code attendance, QR generation, invalid/expired QR validation	All Passed
5	Manage Job and Zone Module	20	Job/zone creation & assignment, GPS tracking, notifications, photo uploads, duplicate prevention, map-based zone editing, allowance calculation	All Passed
6	Manage Driver Salary Module	8	Salary CRUD, attendance & performance-based calculation, payslip generation, duplicate/incomplete salary prevention	All Passed
7	Manage Users and Drivers Module	6	User/driver CRUD, input validation, duplicate email prevention	All Passed
8	Manage Order Module	9	Order CRUD, ROT & CN document generation, CN download (mobile), search & filter	All Passed
9	Manage Customer Module	6	Customer CRUD, field validation, data integrity	All Passed

Table 4 summarizes the test cases executed for each core module of the system, highlighting the number of test cases, key functionalities tested, and their outcomes. A total of 79 test cases were conducted across nine modules, covering critical aspects such as authentication, analytics, profile management, attendance tracking, job and zone operations, salary calculations, user and driver management, order processing, and customer data handling. All test cases were successfully executed and passed, demonstrating that each module performs reliably, securely, and as intended.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1156-1175

4.2 User Acceptance Testing

User Acceptance Testing (UAT) was conducted using a Google Form and involved 2 supervisors and 5 drivers, to evaluate the system's usability, functionality, and overall user experience. Feedback was collected through direct interaction with the system and measured using the System Usability Scale (SUS). The SUS score offers a reliable benchmark to assess user-friendliness [25]. The results indicated a high level of user satisfaction, with users finding the system intuitive, efficient, and aligned with their needs. Suggestions for minor improvements were noted for future refinement.

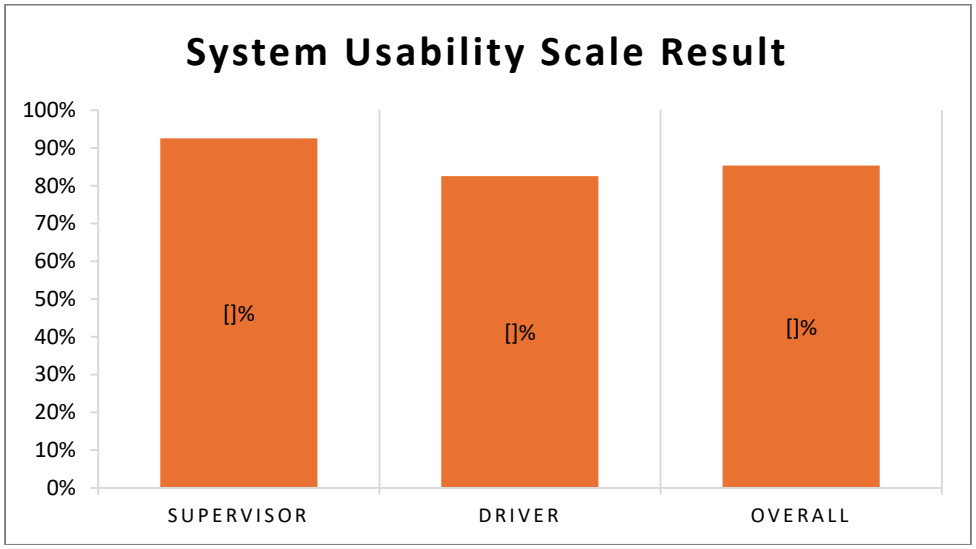


Fig. 18 *UAT Form SUS Results*

Fig. 18 displays the SUS scores for 7 respondents which consists of 2 supervisors and 5 drivers based on their responses to ten items. The average SUS score for supervisors is 92.50 while the average SUS score for drivers is 82.50. Sum it up, the overall average SUS score across all respondents is 85.36, suggesting a generally high level of system usability perceived by the users.

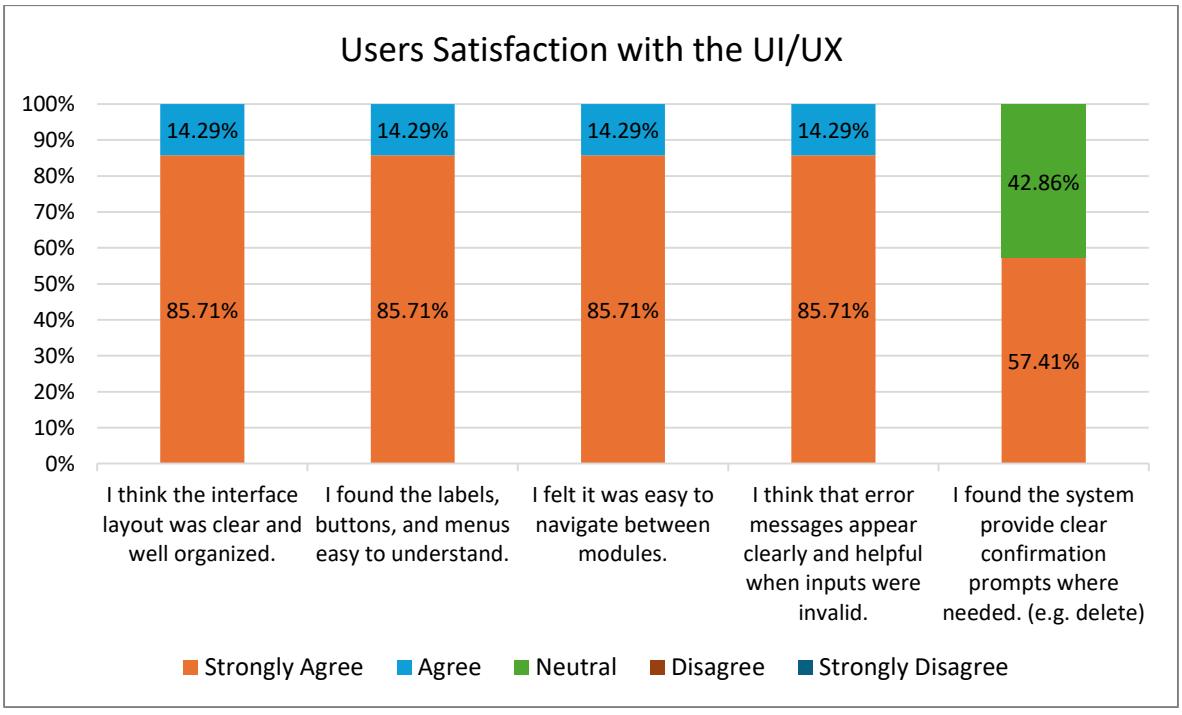


Fig. 19 *UAT Form UI/UX Results*

Feedback on the system's user interface (UI) and user experience (UX) was overwhelmingly positive, as illustrated in Fig. 19. Users found the overall design clear and the navigation intuitive, contributing to a smooth interaction with the system. Specific suggestions for improvement included adding a refresh function on the job dashboard and increasing font sizes for better readability on mobile devices. Most users did not provide additional comments, indicating general satisfaction with the system's visual layout and usability. However, a notable 42.86% of respondents gave neutral responses regarding confirmation prompts. This was primarily due to the absence of confirmation prompts on the mobile platform, while such prompts were present on the web version. These insights confirm the system's strong usability while highlighting minor areas for refinement in future updates

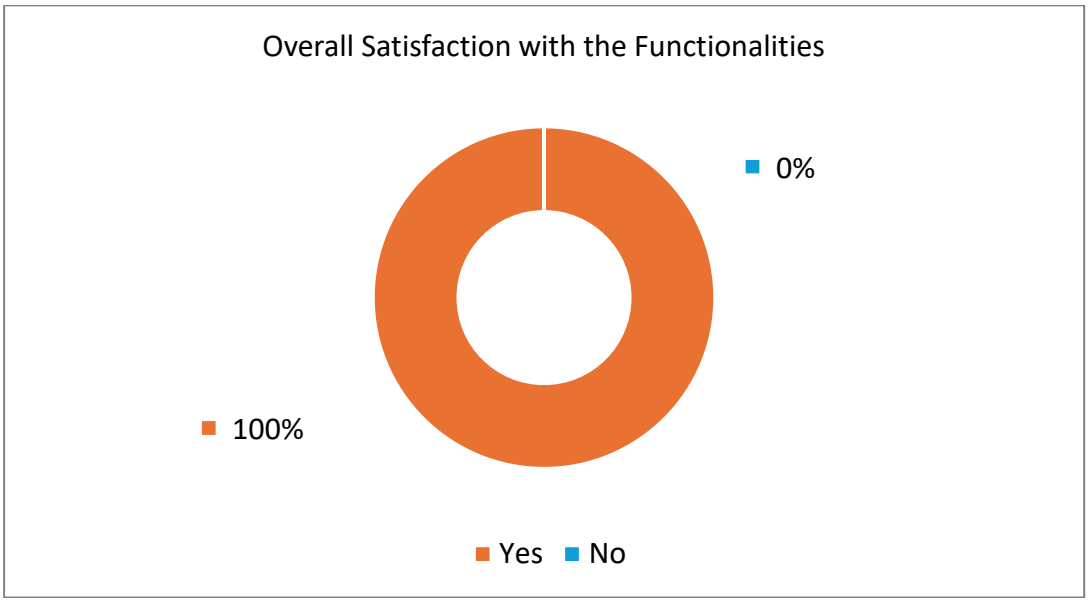


Fig. 20 *UAT Form Functionalities Results*

All users expressed satisfaction with the core functionalities, including job assignment, real-time tracking, and document generation as seen in Fig. 20. One supervisor suggested improving system performance and enhancing the accuracy of gate open/close timestamps to the minute, as aligned with terminal data. A driver also proposed adding navigation features like Google Maps or Waze with estimated time arrival and voice navigation, as well as a notification history page to improve usability. These suggestions have been acknowledged for consideration in future system updates.

5. Conclusion

In conclusion, the development of the Dual-Platform Automated Logistics Management System with Real-Time Tracking successfully met its three objectives. The system was analyzed and designed using an object-oriented approach, resulting in a well-structured and modular architecture based on the 3-tier architecture model, which separates the presentation layer, application logic, and data management to enhance scalability, maintainability, and clarity in the system's structure. It was developed using both Android technology and a web-based approach to support dual-platform accessibility. Furthermore, the system was thoroughly tested through functional testing and user acceptance testing, confirming its usability.

Through the integration of Android and web technologies, supported by real-time data handling via Supabase, the system successfully streamlined key processes such as order management, driver tracking, and document generation. Although some limitations were identified, such as platform restrictions, lack of offline functionality, and limited external integrations, the system proved to be effective and user-friendly. Proposed future enhancements aim to overcome these limitations and further improve the system's scalability, usability, and overall performance. This project underscores the significance of digital transformation in logistics by enhancing operational efficiency through automation, improving real-time visibility, and offering a scalable and replicable solution tailored for small and medium-sized logistics companies.

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:*** *Teoh Wei Jun, Nur Liyana Sulaiman;* ***data collection:*** *Teoh Wei Jun, Nur Liyana Sulaiman;* ***analysis and interpretation of results:*** *Teoh Wei Jun, Nur Liyana Sulaiman;* ***draft manuscript preparation:*** *Teoh Wei Jun, Nur Liyana Sulaiman. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] A. Ahmed, "Web Technologies with Frameworks," *IRSD2024*, vol. 40, 2023.
[2] A. Z. Ayezabu, "Supabase vs Firebase: Evaluation of performance and development of Progressive Web Apps," Bachelor's Thesis, Metropolia University of Applied Sciences, 2022.
[3] S. R. Uplenchwar, U. S. Denge, A. S. Bajoriya, and S. A. Bachwani, "Review on detail information about flutter cross platform," *Int. J. Res. Appl. Sci. Eng. Technol.*
[4] R. N. Southern, "Historical Perspective of the Logistics and Supply Chain Management Discipline," *Transportation Journal*, vol. 50, no. 1, pp. 53–64, 2011. [Online]. Available: https://doi.org/10.5325/transportationj.50.1.0053
[5] I. Al-Hsani and Z. Al-Balushi, "Logistics Sector in Post-COVID-19: Challenges and Opportunities," in A. Mishrif, Ed., *Business Resilience and Market Adaptability: Pandemic Effects and Strategies for Recovery*, Springer Nature Singapore, pp. 137–157, 2024. [Online]. Available: https://doi.org/10.1007/978-981-97-2962-3_8
[6] M. Christopher, *Logistics and Supply Chain Management*, 5th ed., London: Pearson Education, 2016.
[7] A. Shamsuzzoha and P. Helo, "Real-time Tracking and Tracing Systems: Potentials for the Logistics Network," *Proc. 2011 Int. Conf. Ind. Eng. Oper. Manag.*, 2011, pp. 22–24.
[8] N. Chaudhari, "Impact of Automation Technology on Logistics and Supply Chain Management," *Am. J. Theor. Appl. Bus.*, vol. 5, pp. 53, 2019. [Online]. Available: https://doi.org/10.11648/j.ajtab.20190503.12
[9] P. Xiao, "Real-time Tracking System for Freshness of Cold Chain Logistics based on IoT and GPS Platforms," *2020 2nd Int. Conf. Inventive Res. Comput. Appl. (ICIRCA)*, pp. 834–837, 2020. [Online]. Available: https://doi.org/10.1109/ICIRCA48905.2020.9182835
[10] V. S. Devarajulu, "Real-Time Visibility and Tracking for Supply Chain Systems: Improving Inventory Management and Reducing Operational Costs Through Technology," *Int. J. Future Res. J.*, vol. 7, no. 1, pp. 3787, 2020.
[11] Swift Logistics, *Swift Logistics* [Online]. Available: https://swiftlogistics.com.my/. (Accessed: Nov. 9, 2024).
[12] LTS Group, *Lee Ting San Group of Companies* [Online]. Available: http://www.ltsgroup.com.my/en/home/. (Accessed: Nov. 9, 2024).
[13] Interway Transport, *Sing Chuan Aik Interway Transport* [Online]. Available: https://interway.com.my/. (Accessed: Nov. 9, 2024).
[14] T. Ruas and W. Grosky, *Software Engineering: A Practitioner's Approach*, 9th ed. 2020.
[15] I. Sommerville, *Software Engineering*, 10th ed. Pearson Education, 2016.
[16] A. Dennis, B. H. Wixom, and R. M. Roth, Systems Analysis and Design, 6th ed. Wiley, 2014.
[17] G. B. Shelly and H. J. Rosenblatt, *Systems Analysis and Design*, 9th ed. Cengage Learning, 2012.
[18] S. Guruwada, "Understanding Requirement Analysis Phase," *Int. J. Sci. Res. Comput. Sci., Eng. Inf. Technol.*, pp. 474–476, 2021. [Online]. Available: https://doi.org/10.32628/CSEIT217293
[19] L. Gunawardhana, "Process of Requirement Analysis Link to Software Development," *J. Softw. Eng. Appl.*, vol. 12, pp. 406–422, 2019. [Online]. Available: https://doi.org/10.4236/jsea.2019.1210025
[20] J. T. Catanio, "Requirements Analysis: A Review," in T. Sobh and K. Elleithy, Eds., *Advances in Systems, Computing Sciences and Software Engineering*, Springer Netherlands, pp. 411–418, 2006.
[21] K. Pohl, *Requirements Engineering Fundamentals*, 2nd ed., United States: Rocky Nook, 2016.
[22] M. Seidl, M. Scholz, C. Huemer, and G. Kappel, "The Use Case Diagram," in *UML @ Classroom: An Introduction to Object-Oriented Modeling*, Springer International Publishing, pp. 23–47, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-12742-2_3
[23] H. Osman, A. van Zadelhoff, D. R. Stikkolorum, and M. R. V. Chaudron, "UML class diagram simplification: What is in the developer's mind?" *Proc. 2nd Int. Workshop Exp. Empir. Stud. Softw. Modelling*, 2012. [Online]. Available: https://doi.org/10.1145/2424563.2424570
[24] G. J. Klir, *Architecture of Systems Problem Solving*, Springer Science & Business Media, 2013.
[25] J. Brooke, "SUS: A quick and dirty usability scale," in Usability Evaluation in Industry, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, Eds. London: Taylor & Francis, 1996, pp. 189–194.