**Jurnal Kecerdasan Buatan dan Teknologi Informasi**
Vol. 5, No. 3, September 2026, pp. 388–404
e-ISSN: 2964-2922, p-ISSN: 2963-6191

Open access

**Original Research**

# Integrated Web-Based Motorcycle Rental System with Real-Time Fleet Availability Tracking

Arrizqi Hilman Rahmatulloh [a*], Robby Maududy [b]

[a,b] Department of Software Engineering, Faculty of Science and Technology, Universitas Cipasung Tasikmalaya, Indonesia

*Corresponding Author: iki291571@gmail.com

## ABSTRACT

Supermoto Motorcycle Rental is a motorcycle rental business that still manages vehicle data, customer records, rental transactions, and operational reports through manual processes. This condition leads to several critical issues, including difficulties in data retrieval, duplicate records, vehicle data discrepancies, delays in report generation, and the inability to monitor vehicle availability in real time. This study aims to design and implement an integrated web-based motorcycle rental information system to improve operational effectiveness and efficiency. The system was developed using the Rapid Application Development (RAD) method, encompassing requirement planning, user design, construction, and implementation phases. The application was built with the PHP programming language, the CodeIgniter 3 framework implementing the Model-View-Controller (MVC) architecture, and a MySQL database for persistent data storage. Key features include motorcycle fleet management, customer data management, rental transaction processing, payment verification, real-time vehicle availability monitoring, and automated operational reporting. System functionality was rigorously evaluated using the Black Box Testing method across 15 comprehensive test scenarios covering authentication, data management, transactions, and reporting modules. The testing results demonstrated a 100% success rate across all evaluated scenarios, confirming that all system features function correctly and fully meet the specified user requirements. The developed system successfully streamlines data management, enhances data accuracy, accelerates administrative workflows, and provides real-time operational monitoring capabilities. Ultimately, this digital transformation equips Supermoto Motorcycle Rental with a reliable, secure, and scalable platform to support data-driven decision-making and elevate overall customer service quality and long-term business competitiveness.



**How to Cite:**
A. H. Rahmatulloh and R. Maududy, "Integrated Web-Based Motorcycle Rental System with Real-Time Fleet Availability Tracking," *J. Kecerdasan Buatan dan Teknol. Inf.*, vol. 5, no. 3, pp. 388–404, 2026.
Doi: https://doi.org/10.69916/jkbti.v5i3.526

## KEYWORDS

Motorcycle Rental System
Web-Based Information System
Rapid Application Development
CodeIgniter Framework
Real-Time Fleet Monitoring

## ARTICLE INFO

Received: June 10, 2026
Accepted: August 17, 2026
Available Online: September 1, 2026

## 1. INTRODUCTION

The development of information technology in the digital era has brought significant changes to various business and service sectors. Utilizing information technology enhances operational efficiency, accelerates data management, and supports more effective decision-making. One business sector benefiting from this progress is motorcycle rental services, which provide practical, flexible, and economical transportation solutions for the public [1–3], particularly in tourist destinations, urban areas, and educational environments. As public demand for transportation services continues to rise, implementing an information system has become a crucial factor in supporting motorcycle rental operations to make them more effective and organized. Web-based information

systems enable integrated management of motorcycle rental processes, ranging from vehicle data management, customer data processing, and rental transactions to operational reporting [4].

However, many motorcycle rental businesses still rely on manual operational processes. Managing customer data, recording rental transactions, and generating operational reports are often conducted using logbooks or simple, non-integrated software. This operational bottleneck is currently experienced by Supermoto Motorcycle Rental [5–7]. Despite its high market potential due to its strategic location near the Cititis tourist area, manual administrative processes lead to various issues, such as difficulties in data retrieval, duplicate customer records, vehicle data discrepancies, delays in report generation, and sub-optimal customer service. Furthermore, vehicle availability information cannot be accessed in real time, creating potential risks of misinformation and double-booking schedules. Several previous studies have developed web-based vehicle rental information systems to enhance business management effectiveness. Each study presents distinct approaches, features, and outcomes. A detailed comparison of prior research is summarized in Table 1.

Table 1: Comparison of Previous Research

| Researcher | Method | Research Object | System Features | Research Results | Limitations |
|---|---|---|---|---|---|
| Study [8] | Waterfall | Motorent Motorcycle Rental | Customer data, transactions, reports | Administrative processes became computerized | Lacks real-time vehicle monitoring and data integration is limited |
| Study [9] | SDLC | Vehicle Rental | Online transactions, customer data | Improved service efficiency | Does not utilize an MVC framework and vehicle scheduling is not optimal |
| Study [10] | Prototype | Car Rental | Vehicle reservation, payment | Simplified the rental process | Focused only on four-wheeled vehicles and large-scale enterprises |
| This Study | Rapid Application Development (RAD) | Supermoto Motorcycle Rental | Vehicle management, customer management, transactions, payments, vehicle monitoring, reports | Integrated web-based system | Resolves the limitations of previous studies |

Based on Table 1, previous studies have successfully implemented web-based information systems to improve the effectiveness of vehicle rental management. However, several limitations remain unaddressed, such as the lack of real-time vehicle availability monitoring, incomplete integration of vehicle management, customer records, transactions, payments, and reporting within a single platform, as well as the limited utilization of modern web frameworks combined with the Rapid Application Development (RAD) method. To address these limitations, this study designs an integrated web-based motorcycle rental information system that unifies fleet management, customer records, rental transactions, unit scheduling, and operational reporting into a single platform. The Rapid Application Development (RAD) model was selected due to its agile, iterative nature and strong focus on user requirements [11–15]. Through the RAD methodology, system development can be completed within a short timeframe while maintaining software quality. In light of this background, this research focuses on the design and implementation of a Web-Based Motorcycle Rental Information System. The system is projected to streamline data management, enhance service quality, deliver real-time unit availability information [16], and support management in controlling business operations in a more systematic and structured manner.

## 2. RESEARCH METHODS

### 2.1. Rapid Application Development (RAD) Method

The Rapid Application Development (RAD) model is a software engineering methodology that prioritizes accelerated system development through iterative cycles and intensive prototyping [4]. This method allows users to actively participate throughout the development process, ensuring that the resulting system aligns more closely with user requirements [11].

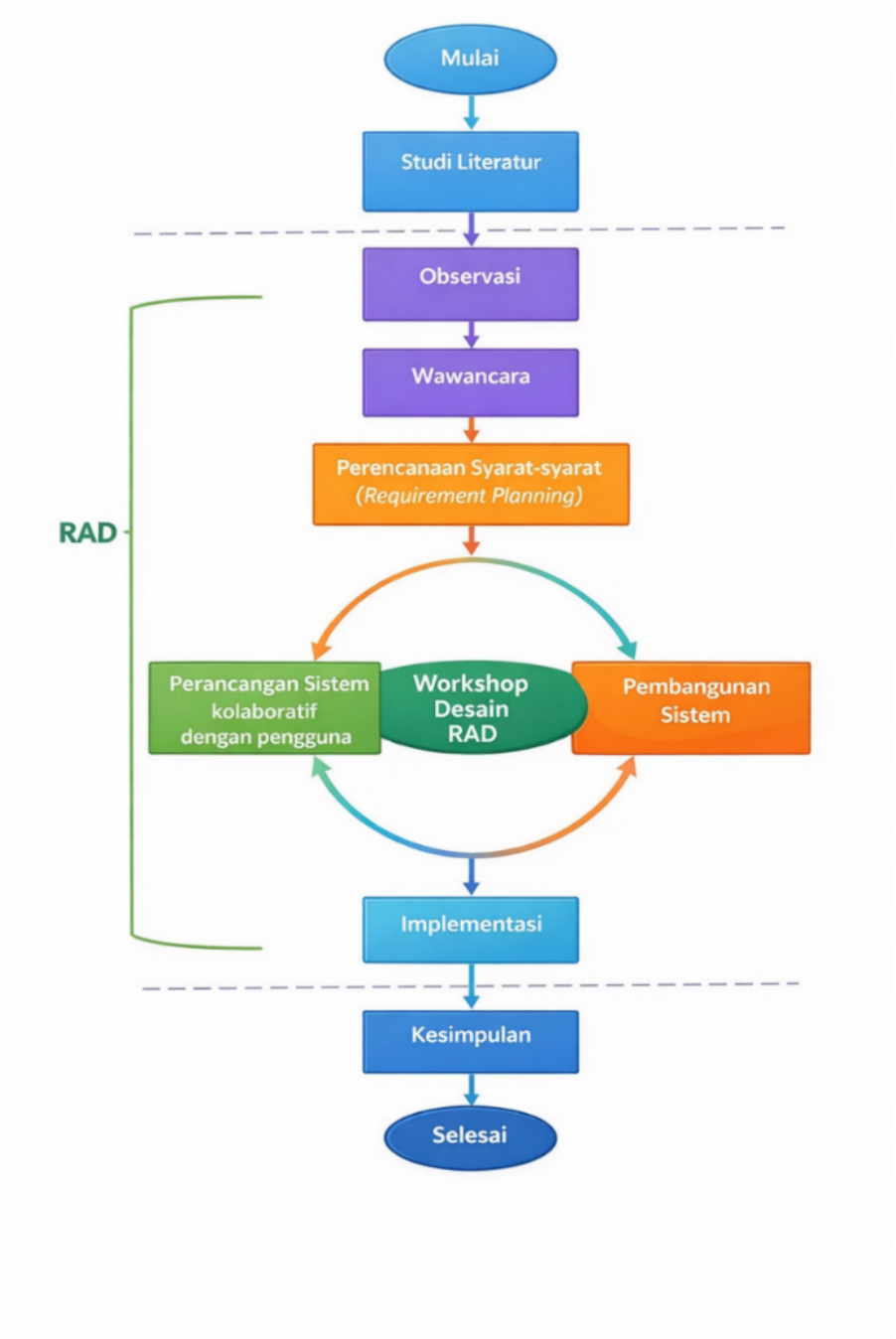


Figure 1. Stages of the Rapid Application Development (RAD) Method

The critical phases of the RAD methodology include Requirement Planning, User Design, Construction, and Implementation. The cycle begins with Requirement Planning to map out system needs, followed by User Design to construct the system architecture and user interface. Subsequently, the Construction phase focuses on application coding according to the design, culminating in Implementation to deploy the system into a live operational environment. Through this workflow, the RAD model can release software significantly faster than conventional approaches.

### 2.2. Unified Modeling Language (UML)

According to [10], alongside the evolution of object-oriented programming (OOP) methodologies, the Unified Modeling Language (UML) emerged as a formal standard modeling language in software development. The introduction of UML was driven by the essential need for visual instruments to specify, represent, construct, and document system artifacts. As a visual modeling language, UML facilitates technical communication regarding system architecture through integrated diagrams and textual descriptions. Types of UML diagrams include Use Case Diagrams, Class Diagrams, Activity Diagrams, Sequence Diagrams, State Machine Diagrams, Communication Diagrams, Deployment Diagrams, Component Diagrams, and Object Diagrams [17–19].

### 2.3. Black Box Testing

According to [20], Black Box Testing is a software quality evaluation method that focuses on the functional aspects of a system. The primary goal of this testing approach is to identify malfunctions, interface defects, data structure errors, performance issues, as well as initialization and termination failures. This method validates the system externally without intervening in or examining its internal code structure. Testing operations are conducted by providing inputs as stimuli, after which the resulting outputs are analyzed and verified against expected criteria. This approach ensures that all features operate in accordance with user functional specifications. Consequently, Black Box Testing is widely implemented in information system testing due to its reliability in mapping functional defects from an end-user perspective [20–22].

## 3. RESULTS AND DISCUSSION

### 3.1. System Architecture and Design

#### 3.1.1. System Architecture

This Motorcycle Rental Information System is designed using a web-based client-server architecture implementing the Model-View-Controller (MVC) pattern within the CodeIgniter 3 framework. Users access the system via a web browser as the client, after which requests are processed by the web server and routed to the application to communicate with the MySQL database for data management before returning the results to the user. The MVC pattern separates system functions into the Model (data management), View (user interface), and Controller (business logic handling), yielding a structured, maintainable, and scalable application architecture. In addition to the MVC pattern, the system is developed using Object-Oriented Programming (OOP) concepts, where each component is implemented as a class utilizing encapsulation and inheritance to enhance code modularity. During the design phase, Unified Modeling Language (UML), including Use Case, Activity, Sequence, and Class Diagrams, was employed to model functional requirements, process flows, object interactions, and structural components prior to implementation. The integration of client-server architecture, OOP concepts, and UML modeling produces a systematic, extensible platform that effectively supports motorcycle rental operations [23].

#### 3.1.2. Use Case Diagram

The functional relationships of the designed system are visually presented in the Use Case Diagram below.
The use case model serves to project the interactions between users and the system [13]. Through this diagram, the functional boundaries and activities that can be executed by each actor in the Motorcycle Rental Information System are clearly outlined. The diagram identifies two main actors: Customer and Admin. Customers can register, authenticate (login), view detailed fleet information, make rental reservations, and complete payment transactions. Conversely, the Admin actor possesses full authority to manage motorcycle data, brand categories, customer records, transaction histories, and operational reports. To ensure data security and enforce strict access control, user authentication via login is required prior to performing any system activity.

#### 3.1.3. Activity Diagram

An Activity Diagram models the dynamic operational behavior and workflow execution of a system [14]. In this study, the Activity Diagram illustrates the end-to-end operational processes of the Web-Based Motorcycle Rental Information System, detailing the sequential interactions among three main partitions: Customer, System, and Admin. The comprehensive activity workflow is visually depicted in Figure 3.
As illustrated in Figure 3, the operational process is categorized into three distinct execution swimlanes: Cus-

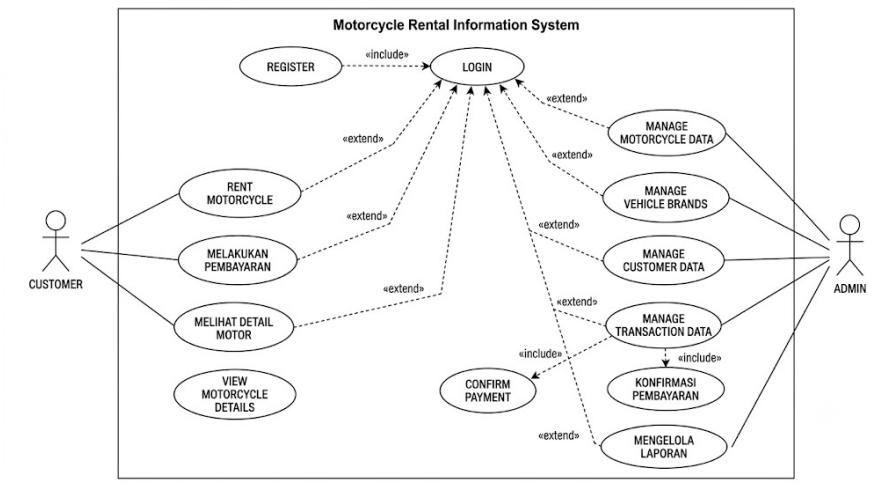


Figure 2. Use Case Diagram

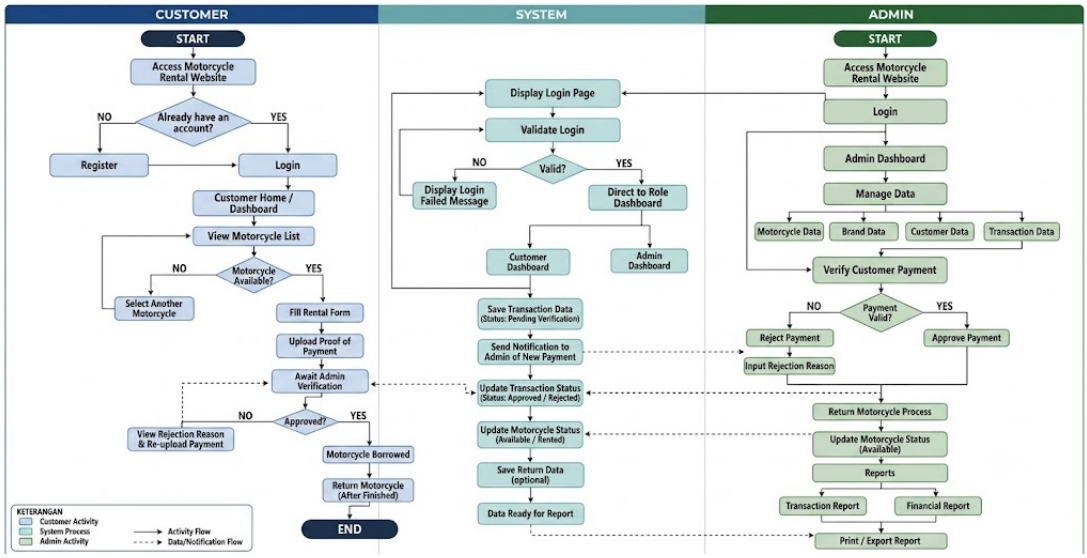


Figure 3. Activity Diagram of the Motorcycle Rental System

tomer activities, System processes, and Admin procedures. The detailed workflow unfolds as follows:

1. **Authentication and Role-Based Redirection:** Both Customer and Admin initiate the process by accessing the rental system website. Unregistered customers are directed through the registration workflow to create an account, while existing users proceed directly to the login interface. The system captures user credentials and performs input validation. If authentication fails, the system displays an error notification and prompts the user to re-enter credentials. Upon successful validation, the system automatically redirects users to their respective dashboards based on assigned access privileges (Customer Dashboard or Admin Dashboard).

2. **Vehicle Selection and Rental Transaction:** From the Customer Dashboard, the customer browses the available motorcycle catalog. The system evaluates vehicle availability; if a selected vehicle is marked as unavailable, the customer is prompted to choose an alternative unit. When an available unit is selected, the customer completes the online rental form and uploads proof of payment. The system records the transaction record under the status "Pending Verification" and dispatches an automated notification to the administrator.

3. **Payment Verification and Approval:** Upon receiving the payment notification, the Admin reviews the transaction details and verifies the payment proof from the Admin Dashboard. If the payment proof is invalid, the Admin rejects the payment and inputs a reason for rejection; the system then updates the

transaction status to "Rejected" and prompts the customer to view the rejection details and upload a corrected payment receipt. If the payment proof is valid, the Admin approves the transaction. The system updates the transaction status to "Approved", marks the motorcycle status as "Rented", and updates the customer interface to reflect the active rental state ("Motorcycle Rented").

4. **Vehicle Return Processing and Report Generation:** After the rental duration concludes, the customer returns the vehicle. The Admin processes the vehicle return entry in the admin portal, prompting the system to update the motorcycle availability status back to "Available" and saving the return logs. Once the transaction cycle is completed, the system organizes the operational data for analytical processing, enabling the Admin to generate, print, and export transaction and financial reports in PDF or Excel formats.

### 3.1.4. Sequence Diagram

A Sequence Diagram is a behavioral UML diagram that models object interactions over time, detailing the step-by-step exchange of messages between actors, user interfaces, controllers, data models, and the underlying database [15]. In this study, sequence diagrams are categorized into six primary functional workflows: User Authentication, Customer Registration, Motorcycle Rental Transaction, Payment Verification, Vehicle Return, and Operational Reporting. The complete interaction sequences are illustrated in Figure 4.

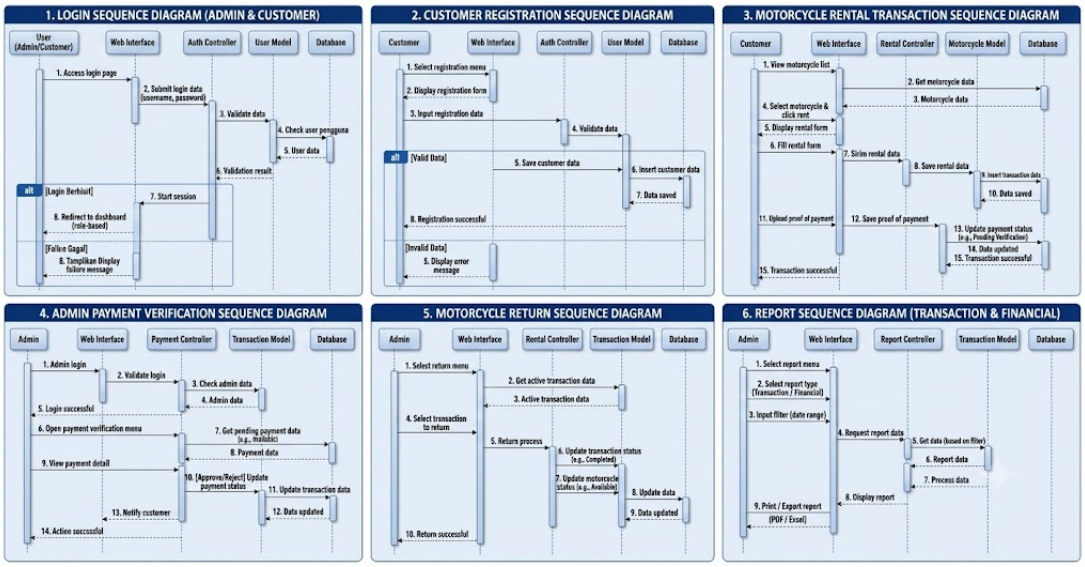


Figure 4. Sequence Diagrams of Core System Functions

As depicted in Figure 4, the dynamic message exchanges across the system components are divided into six main sub-diagrams:

1. **Sequence Diagram 1: Login Process (Admin & Customer):** The user accesses the login page via the Web Interface, which sends credentials (username and password) to the Auth Controller. The controller forwards the request to the User Model to query user data from the Database. Upon returning the query result, an alternative frame (alt) is executed:
    a. Login Success: The Auth Controller initializes a session and redirects the user to the appropriate dashboard based on their role (Customer or Admin).
    b. Login Failure: The controller prompts the Web Interface to render an error message.
2. **Sequence Diagram 2: Customer Registration:** The user requests the registration form through the Web Interface. Upon submitting the filled form, the Auth Controller validates the input:
    a. Data Valid: The controller invokes the User Model to insert the new profile into the Database. Once saved, a confirmation is returned, and the system notifies the customer of a successful registration.

b. Data Invalid: The system bypasses database entry and immediately displays a validation error message on the Web Interface.

3. **Sequence Diagram 3: Motorcycle Rental Transaction:** The customer views the vehicle catalog. The Rental Controller queries active fleet listings from the Motor Model / Database and renders the rental form. The customer submits rental details, which are passed to the Rental Controller to be saved in the Database. Next, the customer uploads payment proof. The controller updates the transaction status to "Pending Verification" and returns a successful booking notification.
4. **Sequence Diagram 4: Payment Verification (Admin):** The Admin authenticates and accesses the payment verification menu via the Web Interface. The Payment Controller fetches pending records from the Transaksi Model / Database. Upon reviewing payment details, the Admin approves or rejects the submission. The Payment Controller updates the payment and transaction status in the database, sends a notification response to the customer, and renders a success message to the Admin.
5. **Sequence Diagram 5: Vehicle Return Process:** The Admin opens the vehicle return interface. The Rental Controller retrieves active transaction logs from the Transaksi Model / Database. When the Admin selects a transaction to process, the Rental Controller updates the transaction status to "Completed" (Selesai) and changes the motorcycle status back to "Available" (Tersedia). The updated state is saved in the database, and a completion notification is displayed.
6. **Sequence Diagram 6: Operational & Financial Reporting:** The Admin accesses the reporting module, selects the report category (Transaction or Financial), and applies date filter parameters. The Laporan Controller requests filtered data from the Transaksi Model, which queries the Database. Once the data is processed, the system renders the report on the Web Interface, allowing the Admin to print or export the records in PDF or Excel formats.

### 3.1.5. Class Diagram

A Class Diagram provides a static structural representation of the system by illustrating its object classes, attributes, methods, and the logical relationships among entities [16]. In this research, the Class Diagram maps out the data models and core operations that form the backend architecture of the Web-Based Motorcycle Rental Information System. The complete class structure and entity relationships are shown in Figure 5.

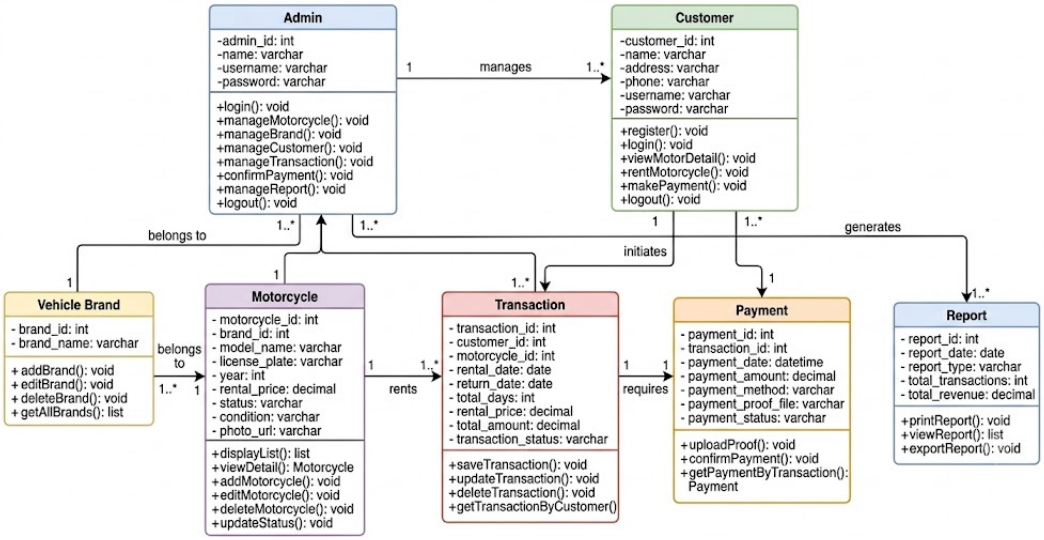


Figure 5. Class Diagram of the Motorcycle Rental Information System

As depicted in Figure 5, the static architecture of the system consists of seven core entities structured to handle data persistence, business logic, and operational attributes. The detailed breakdown of each class and its structural relationships is described as follows:

**Core Classes and Attributes:**

1. Admin Class: Encapsulates system administrative credentials (`admin_id`, `admin_name`, `username`, `password`) and operational methods such as `manageMotorcycle()`, `manageBrand()`, `manageCustomer()`, `manageTransaction()`, `confirmPayment()`, and `manageReport()`.
2. Customer Class: Represents system end-users with personal data attributes (`customer_id`, `customer_name`, `address`, `phone_number`, `username`, `password`) and functional methods including `register()`, `login()`, `viewMotorcycleDetail()`, `rentMotorcycle()`, and `makePayment()`.
3. Vehicle Brand Class: Stores vehicle brand metadata using `brand_id` and `brand_name`, managed through CRUD operations (`addBrand()`, `editBrand()`, `deleteBrand()`, `getAllBrands()`).
4. Motorcycle Class: Manages the rental fleet repository with attributes including `motorcycle_id`, `brand_id`, `motorcycle_name`, `plate_number`, `year`, `rental_price`, `status`, `condition`, and `photo`. It includes methods for fleet management (`displayList()`, `viewDetail()`, `addMotorcycle()`, `editMotorcycle()`, `deleteMotorcycle()`, `updateStatus()`).
5. Transaction Class: Handles rental booking logs, containing transaction parameters (`transaction_id`, `customer_id`, `motorcycle_id`, `rental_date`, `return_date`, `total_days`, `rental_price`, `total_amount`, `transaction_status`) and execution methods (`saveTransaction()`, `updateTransaction()`, `deleteTransaction()`, `getTransactionByCustomer()`).
6. Payment Class: Manages transaction settlement data (`payment_id`, `transaction_id`, `payment_date`, `payment_amount`, `payment_method`, `payment_proof`, `payment_status`) along with verification processes (`uploadProof()`, `confirmPayment()`, `getPaymentByTransaction()`).
7. Report Class: Compiles analytical outputs for operational monitoring (`report_id`, `report_date`, `report_type`, `total_trans`, `total_revenue`) via reporting methods (`printReport()`, `viewReport()`, `exportReport()`).

**Multiplicity and Entity Relationships:**

1. Admin – Customer (1 to 1..*): One Admin entity actively manages multiple Customer records.
2. Admin – Report (1 to 1..*): One Admin generates and manages multiple operational and financial reports.
3. Vehicle Brand – Motorcycle (1 to 1..*): One brand category can be associated with one or many motorcycles in the fleet repository.
4. Motorcycle – Transaction (1..* to 1): One or more motorcycle units can be linked to a single rental transaction.
5. Customer – Transaction (1 to 1..*): One Customer can initiate and hold multiple rental transactions over time.
6. Transaction – Payment (1 to 1..*): Each rental transaction requires one or more payment logs, which are subsequently verified through the payment verification workflow.

### 3.1.6. Entity Relationship Diagram (ERD)

Data architecture and database logical modeling play a crucial role in ensuring data consistency and integrity across system operations [17]. In this study, the relational structure of the backend database is modeled through an Entity Relationship Diagram (ERD), which defines primary keys (PK), foreign keys (FK), entity attributes, and cardinalities among tables. The complete database schema for the Motorcycle Rental Information System is visually presented in Figure 6.

As depicted in Figure 6, the relational database design consists of five main entities structured to handle system authentication, fleet classification, customer profiles, and rental execution logs. The detailed breakdown of entities, primary/foreign keys, and cardinalities is outlined below:

1. **Database Tables and Attributes:**
    a. ADMIN Table: Stores administrative account details with `admin_id` as the Primary Key (PK), along with name, username, and password.

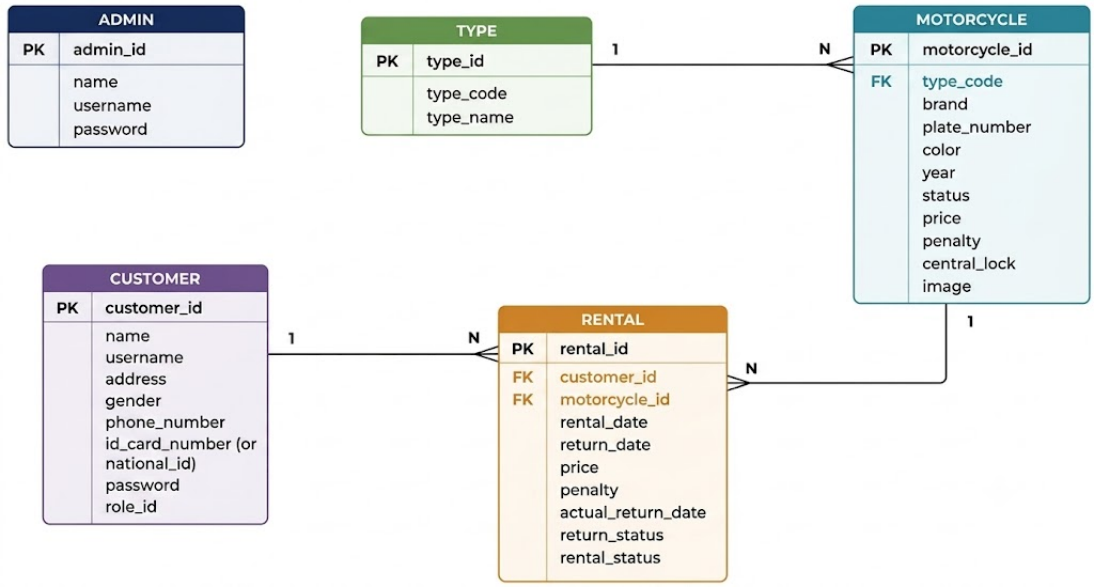


Figure 6. Entity Relationship Diagram (ERD) / Database Schema

b. TYPE Table: Serves as a classification master table storing motorcycle category information, identified by `type_id` (PK), `type_code`, and `type_name`.

c. MOTORCYCLE Table: Represents the fleet inventory records. It utilizes `motorcycle_id` as the Primary Key (PK) and references `type_code` as a Foreign Key (FK) linked to the vehicle category. Additional attributes include physical properties and pricing models such as brand, `plate_number`, color, year, status, price, penalty, `central_lock`, and image.

d. CUSTOMER Table: Manages customer profile records using `customer_id` as the Primary Key (PK). Attributes include identity credentials (name, username, password, `id_card_number` or `national_id`, `role_id`) and contact parameters (address, gender, `phone_number`).

e. RENTAL Table: Acts as the central transactional entity, identified by `rental_id` (PK). It incorporates `customer_id` (FK) and `motorcycle_id` (FK) to link customers and reserved vehicles. Operational metrics logged include `rental_date`, `return_date`, price, penalty, `actual_return_date`, `return_status`, and `rental_status`.

2. **Table Relationships and Cardinalities:**

a. TYPE to MOTORCYCLE (1 to N): A One-to-Many relationship exists where one vehicle type can classify multiple motorcycles, represented by the Foreign Key `type_code` in the MOTORCYCLE table.

b. CUSTOMER to RENTAL (1 to N): A One-to-Many relationship where a single customer can initiate multiple rental transactions over time, mapped via `customer_id` in the RENTAL table.

c. MOTORCYCLE to RENTAL (1 to N): A One-to-Many relationship where an individual motorcycle unit can be involved in multiple rental transactions across different time periods, linked via `motorcycle_id` in the RENTAL table.

### 3.2. System Features Implementation

This section details the practical implementation and graphical user interface (GUI) realizations of the proposed Motorcycle Rental Information System, as outlined during the design phase. The implementation showcases the technical deployment of core functional features across three main system tiers: the frontend user interface, backend logic controllers, and persistent data storage. By integrating the behavioral logic from the sequence diagrams and the structural data models from the class diagram, specific modules were developed to facilitate secure user interactions, transaction management, and administrative oversight. The following subsections

present selected screenshots from the operational web-application, describing the user experience and logical process flow for critical system tasks, beginning with the common entry point for all users. To illustrate the user interface and system capabilities from the client's perspective, this section presents the frontend design of the application. The customer's main interface represents the critical entry point and general landing page where potential users interact with the system's logic before initiating any transactions. The graphical user interface (GUI) was developed based on the interaction flow established in the 'REGISTER' and 'VIEW MOTORCYCLE DETAILS' sequence diagrams and integrates the data attributes from the 'Customer' class diagram. This general frontend interface is operational and accessible via a web browser, with its complete structure and initial content visually detailed in Figure 7.

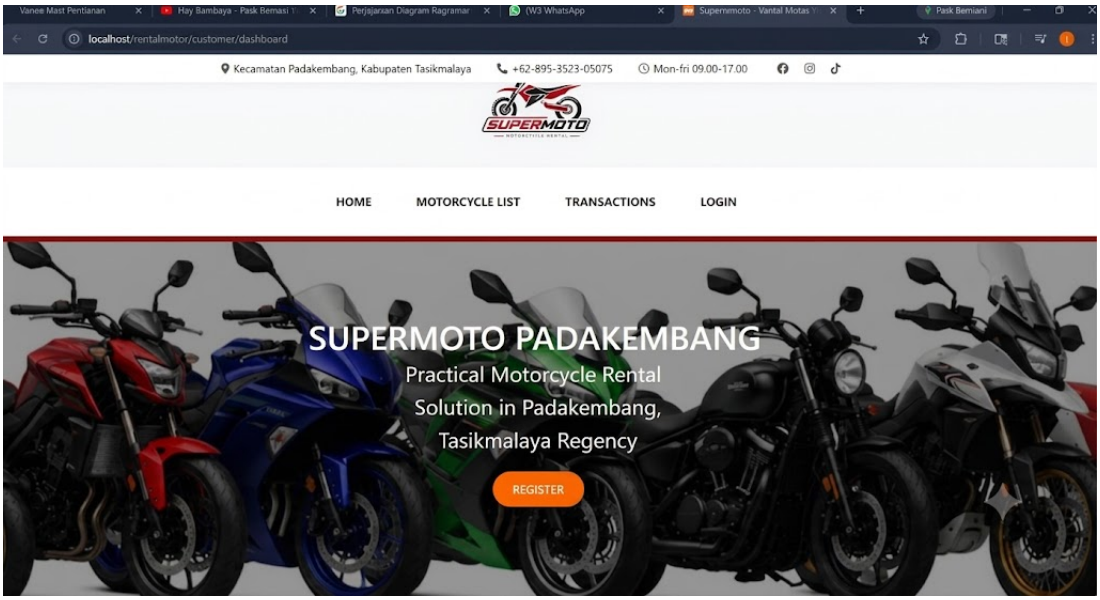


Figure 7. General Backend Customer Frontend Interface

As depicted in Figure 7, the general frontend interface for customers features a minimalist design with a structured layout tailored for operational ease. The interface incorporates several critical design elements derived directly from the underlying data model. First, the header area facilitates user communication by integrating the contact attribute of the primary Admin class, displaying the specific operation address and contact number that the Admin manages in real-time. Below the communication bar, the main navigation menu enables users to execute behavioral methods from the 'Customer' sequence diagram, including basic navigation ('HOME'), fleet exploration ('MOTORCYCLE LIST'), user validation ('LOGIN'), and access to historical logs ('TRANSACTIONS'), where authentication is verified. The central canvas contains the hero section, featuring high-fidelity visuals of the operational fleet. Superimposed white text presents the specialized brand identity, in this instance, 'SUPERMOTO PADAKEMBANG'. This identity is a distinct attribute that separates different branches within the underlying 'Admin' database table. It serves as a clear indicator of the specific regional operations that the general backend system is supporting. A primary action component, namely the 'REGISTER' button, is prominent, designed to initiate the `register()` method within the 'Customer' class diagram and trigger the corresponding sequence for new user onboarding. By integrating these specific attributes and methods directly into the user interface components, the general frontend provides a seamless transactional experience for customer users, connecting frontend interaction to backend database logic without complex transitions.

Following successful authentication, the customer is directed to the transaction module to initiate a vehicle booking. This interface serves as the practical implementation of the 'RENT MOTORCYCLE' use case and operationalizes the `rentMotorcycle()` method defined in the Customer class diagram [18]. The graphical user interface (GUI) was designed to capture essential temporal attributes requisite for transaction processing, thereby linking backend logic controllers to dynamic user input. The operational structure of this primary booking form, accessible via the customer dashboard, is presented in Figure 8.

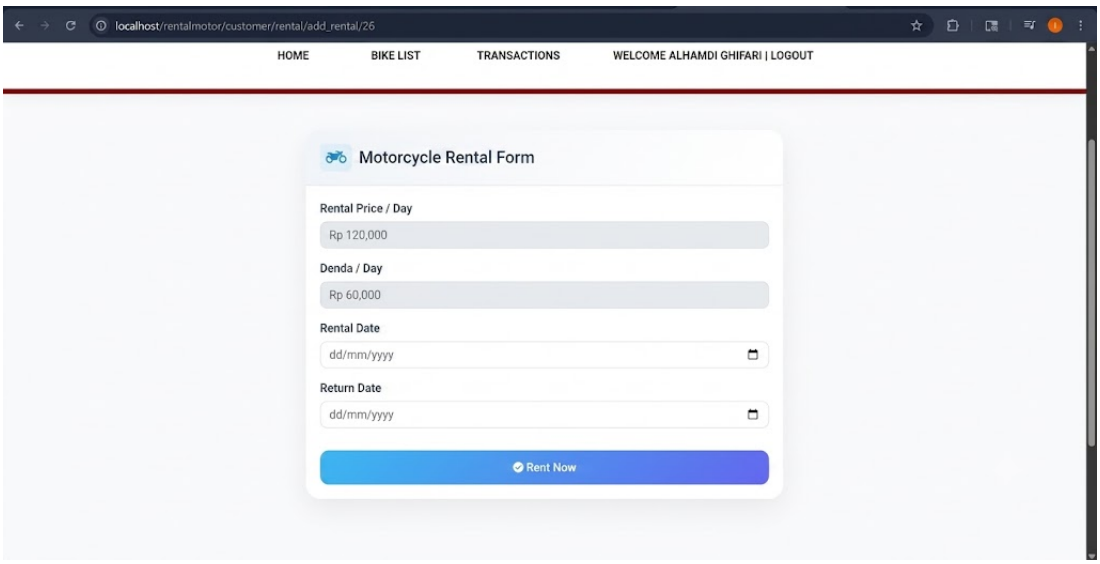


Figure 8. Motorcycle Rental Form Page

As depicted in Figure 8, the page presents a structured 'Motorcycle Rental Form' presented as a focused modal or central element [2]. The interface incorporates specific data attributes retrieved from the underlying 'Motorcycle' database entity to facilitate user decision-making, namely the fixed 'Rental Price / Day' and 'Denda / Day' (Penalty / Day). This realization aligns with the data model's requirement to display vehicle specifics before transaction commitment. For temporal transaction logging, two dynamic input components are provided to capture the 'Rental Date' and 'Return Date' using standard date picker controls (dd/mm/yyyy). The execution of this form is centralized via the prominent blue action component labeled 'Rent Now', which is designed to invoke the `saveTransaction()` method on the backend to validate temporal constraints and instantiate a new pending transaction record, thus transitioning the user from the fleet exploration phase to the active rental queue.

Subsequent to the successful submission of the rental form, the system transitions to the secure payment module. This step represents the direct implementation of the 'MAKE PAYMENT' use case and exemplifies the transaction methods defined in the Class Diagram for creating a rental record. A unique transactional page is dynamically generated to present precise, pre-calculated rental specifics, thereby minimizing operational errors by providing a clear verification summary before the user commits funds. By integrating behavioral logic with the data model, this interface facilitates both data transparency and transactional flow, as visibly detailed in Figure 9.

As depicted in Figure 9, the interface is explicitly structured to facilitate a multi-step verification and payment process [18]. The primary action begins with the central verification summary component. This layout aggregates pre-calculated temporal and financial attributes requisite for transaction finalization. Specifically, the data elements displayed include the unique Bike Brand identifier, the critical Rental Start and End Dates, the specific fixed Rental Fee Per Day, and the calculated Number of Rental Days. Based on this complex data integration, the definitive Total Payment value is dynamically computed and prominently highlighted, acting as a conclusive verification checkpoint derived directly from the Backend logic controllers [2]. To accommodate diverse payment preferences, the operational realization of Figure 9 includes a versatile Payment Method Selection component. This section presents dual options to the customer for transaction completion: conventional account transfers via major Indonesian banking institutions (BRI and BCA) or digital wallet services (DANA) using precise account attributes. For accelerated transaction execution, a specialized Scan QR Payment component is integrated, featuring a unique QRIS-compatible matrix that supports instant validation from standard e-wallet applications. To finalize the method selection and transition the transaction status from 'Pending Payment' to 'Awaiting Verification', a primary action component labeled 'Upload Proof of Payment' is provided, designed to initiate the `confirmPayment()` method and upload the required proof artifact to the Admin tier for manual validation [20].

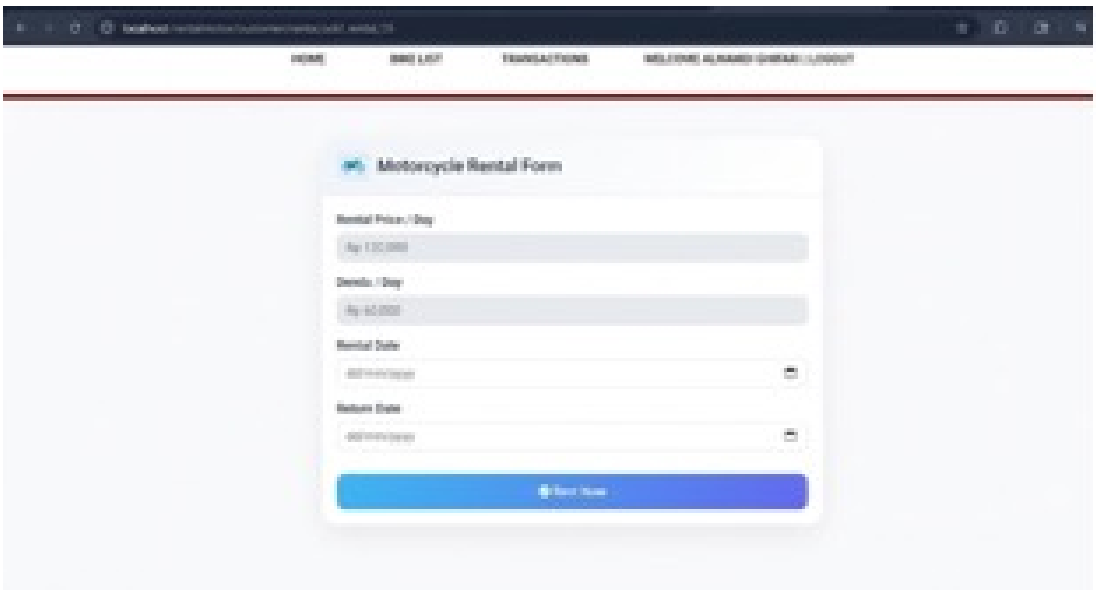


Figure 9. Motorcycle Rental Payment Verification and Method Selection Page

To operationalize the exploration of the vehicle fleet, the system provides a specialized interface focusing on singular vehicle specifics. This page acts as the practical realization of the 'VIEW MOTORCYCLE DETAILS' use case and implements the `viewDetails()` method defined in the Motorcycle class diagram [18]. By navigating from the general fleet listing to this unique view, users can verify critical attributes that are essential for transaction finalization, thereby bridging the behavioral logic with structural data integrity. The operational structure of this detailed motorcycle specification page, derived directly from the Backend database tiers, is presented in Figure 10.

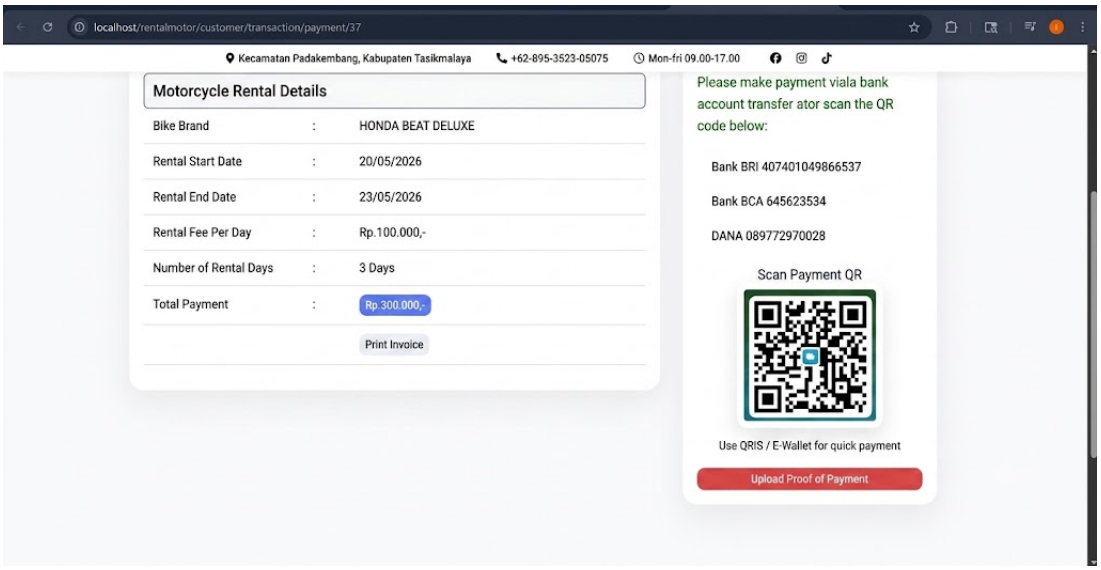


Figure 10. Motorcycle Detail Page Interface

As depicted in Figure 10, the detailed motorcycle specification page is structured with a minimalist layout, focusing the user's attention on specific data attributes requisite for transaction processing [2]. The graphical user interface (GUI) centralizes high-fidelity visual representations and technical data fields derived from the underlying data model. Specifically, the data displayed includes the unique `motorcycle_id` (reflected in the browser URL bar), the 'Brand' entity identifier, the physical 'Plate Number', the vehicle 'Color', the 'Manufacturing Year', and the critical operational 'Status'. By presenting the precise 'STATUS: AVAILABLE' value as a highlighted component, the backend logic tier dynamically confirms vehicle availability, which is the necessary prerequisite to initiate a rental request. To proceed with the transaction based on this information, a primary

action component labeled 'RENT NOW' is integrated, designed to invoke the `rentMotorcycle()` method and transition the user from the view details phase to the booking form phase, linking frontend interactions to stateful system logic without complex multi-step forms [20].

### 3.3. System Security Implementation

Beyond the realization of core functional features, ensuring the integrity and security of data and system operations is a critical requirement. To achieve this, several multi-layered security aspects were implemented in the proposed Motorcycle Rental Information System, focusing on preventing unauthorized access, ensuring data validation, and mitigating common cyber threats. This logical and operational security framework, derived directly from the Backend security configuration and logic tier, is comprehensively summarized in Table 2, detailing each aspect, its corresponding implementation technique, and its specific goal within the system architecture.

To ensure the system operates safely and protects user data from potential vulnerabilities, several security measures have been integrated into the application. Table 2 outlines the key security aspects, their technical implementations, and their respective objectives within the system.

Table 2: System Security Implementation Aspect Summary

| No | Security Aspect | Implementation | Objective / Purpose |
|---|---|---|---|
| 1 | Authentication | Login using username and password. | Ensures that only registered users can access the system. |
| 2 | Authorization | Access privileges are differentiated between Admin and Customer. | Restricts system access based on user roles. |
| 3 | Session Management | Sessions are actively maintained while the user is logged in. | Maintains login state and prevents unauthenticated access. |
| 4 | Input Validation | Input data is validated prior to processing and storage. | Reduces input errors and maintains data consistency. |
| 5 | Database Security | Utilizing CodeIgniter Query Builder. | Mitigates the risk of SQL Injection attacks. |
| 6 | XSS (Cross-Site Scripting) | Filtering user input data and output rendering. | Reduces the risk of malicious script injection into the application. |
| 7 | Upload Validation | Restricting allowed file types and file sizes for payment proofs. | Prevents uploading invalid or unauthorized files. |
| 8 | Logout | Session data is destroyed when the user logs out. | Prevents unauthorized account misuse. |
| 9 | Data Backup | Performing periodic database backups. | Reduces the risk of data loss. |

The overall architecture of the system's security features, as outlined in Table 2, is designed to cover various critical entry points and data processing stages within the application:

1. Authentication: As listed in item 1 of Table 2, the system implements a form-based authentication mechanism requiring a valid username and password. This acts as the primary gatekeeper to ensure that system features are accessible only to legitimately registered users.
2. Authorization: In item 2, role-based access control is enforced to separate user privileges, specifically between Administrators and Customers. This guarantees that ordinary customers cannot access sensitive administrative tools, maintaining strict boundaries based on user roles.
3. Session Management: Item 3 highlights session tracking while users remain logged in. By actively verifying session state on protected pages, the system prevents unauthenticated users from bypassing the login screen through direct URL access.
4. Input Validation: In item 4, input validation is executed before any submitted data is processed or stored in the database. This step prevents invalid data formats, reduces system logical errors, and guarantees data consistency.
5. Database Security: Item 5 details the utilization of CodeIgniter's built-in Query Builder. By leveraging parameterized queries automatically provided by the framework, the application neutralizes potential SQL

Injection attacks that attempt to manipulate raw database queries.

6. XSS (Cross-Site Scripting) Prevention: As described in item 6, strict filtering of user input and safe output rendering are enforced. This prevents attackers from injecting malicious client-side JavaScript code into input fields that could otherwise be rendered in other users' browsers.
7. Upload Validation: Item 7 addresses file upload safety, specifically for payment proof receipts. The application strictly checks the extension type and file size limits before saving uploads, preventing malicious scripts disguised as image files from being uploaded to the server.
8. Logout: In item 8, the system handles session destruction when the user clicks logout. All session credentials and tokens are completely cleared from memory, effectively stopping unauthorized access via the same browser session.
9. Data Backup: Finally, item 9 covers the data backup strategy. Periodic database backups are scheduled to safeguard core system data, minimizing the impact of potential data loss due to unexpected server or hardware failures.

### 3.4. Functional Testing Results

To evaluate the overall functionality and operational behavior of the application, Black Box Testing was conducted. This testing strategy focuses exclusively on verifying the functional requirements and external software behaviors without inspecting the underlying source code. Each test scenario is designed to validate specific core features, ranging from user authentication and master data management to rental transactions and financial reporting. By inputting both valid (positive testing) and invalid (negative testing) datasets, the system's ability to correctly process user actions, enforce business rules, handle data entry errors, and display appropriate feedback messages is systematically evaluated. Table 3 details the executed test scenarios, input data, expected outputs, and their corresponding test results.

Table 3: Black Box Functional Testing Results

| No | Feature | Test Scenario | Input Data | Expected Result | Status |
|---|---|---|---|---|---|
| 1 | Login | User enters a valid username and password | Valid username and password | The system successfully authenticates the user and displays the dashboard based on their access role. | Passed |
| 2 | Login | User enters an incorrect password | Valid username, incorrect password | The system rejects the login attempt and displays the message "Username or Password incorrect". | Passed |
| 3 | Customer Registration | Customer completes all required registration fields | Full Name, Username, Password, Address, Phone Number, ID Number (KTP), Gender | The system successfully saves the customer data and displays a success notification. | Passed |
| 4 | Customer Registration | Customer leaves a mandatory field empty | Name field left blank | The system rejects the submission and displays a message stating that mandatory fields must be filled. | Passed |
| 5 | Motorcycle Data Management | Admin adds new motorcycle data | Complete motorcycle details | The system successfully stores the new motorcycle data in the database. | Passed |
| 6 | Motorcycle Data Management | Admin deletes a motorcycle entry | Selected motorcycle record | The system successfully removes the selected motorcycle record from the database. | Passed |
| 7 | Brand Data Management | Admin adds a new motorcycle brand | Motorcycle brand name | The system successfully saves the new brand entry. | Passed |
| 8 | Brand Data Management | Admin updates existing motorcycle brand data | New brand name | The system successfully updates the brand details in the database. | Passed |
| 9 | Rental Transaction | Customer rents an available motorcycle | Available motorcycle, rental date, rental duration | The system successfully saves the transaction, calculates total cost, and updates motorcycle status to Rented. | Passed |
| 10 | Rental Transaction | Customer attempts to rent an unavailable motorcycle | Motorcycle status set to Rented / Unavailable | The system rejects the transaction and displays the message "Motorcycle is not available for rent". | Passed |
| 11 | Rental Transaction | Customer returns the rented motorcycle according to schedule | Motorcycle return details | The system updates the return log and changes the motorcycle status back to Available. | Passed |
| 12 | Transaction Report | Admin filters transaction reports by a valid period | Valid start date and end date | The system displays transaction records corresponding to the selected period and enables print functionality. | Passed |
| 13 | Financial Report | Admin selects a valid period for financial reporting | Valid start date and end date | The system generates the financial report showing total revenue and provides a print option. | Passed |
| 14 | Financial Report | Admin leaves the date filter blank | Start date and end date left empty | The system displays a validation warning stating that the reporting period must be specified. | Passed |
| 15 | Financial Report | Admin inputs an invalid date range | Start date is set later than end date | The system cancels processing and displays the error message "Invalid date range". | Passed |

Based on the Black Box testing results presented in Table 3, all 15 test scenarios yielded expected results without encountering critical system errors or unhandled execution exceptions. The analysis of the test outcomes can be categorized into three primary execution domains:

1. Authentication and Access Control (Test Cases 1–4): The application correctly restricts access to unauthorized entities while granting smooth login transitions for verified Admin and Customer roles. The form validation mechanism effectively intercepts incomplete registration forms and incorrect password attempts, displaying precise error messages to guide user corrections.
2. Master Data & Transactional Processing (Test Cases 5–11): Data persistence operations (Create, Read, Update, Delete) across the motorcycle fleet and brand management modules executed seamlessly. Furthermore, the core rental state engine worked reliably during transactions: renting a motorcycle automatically shifts its availability status from Available to Rented and calculates total pricing accurately, whereas returning a vehicle properly resets its status to Available, preventing double-booking anomalies.
3. Reporting & Input Validation Integrity (Test Cases 12–15): The report generation modules demonstrate robust parameter validation. When provided with valid date boundaries, the system renders complete transactional and financial summary data with printable views. Crucially, negative validation rules were successfully triggered when handling empty date fields or inverted date ranges (e.g., start date exceeding end date), neutralizing logic errors and ensuring reporting integrity.

In conclusion, the system's functional performance demonstrates high reliability and compliance with specified requirements, confirming that the business logic, input validation filters, and database interactions function correctly across all key modules.

### 3.5. Discussion

The development of the Web-Based Motorcycle Rental Information System for Supermoto Motorcycle Rental has successfully addressed the core operational inefficiencies previously associated with manual management. By leveraging the Rapid Application Development (RAD) method and the CodeIgniter 3 framework, the system effectively unifies fleet management, customer data processing, rental transactions, and operational reporting into a single, cohesive platform. The 100% success rate achieved during Black Box Testing validates that all functional requirements, from secure user authentication and real-time vehicle availability tracking to dynamic report generation, operate reliably and meet the specified business needs.

When compared to prior research, this study offers distinct and measurable advancements. While earlier systems often lacked real-time monitoring capabilities or relied on non-MVC architectures (as noted in Studies [1] and [2]), this research successfully implements a structured Model-View-Controller (MVC) pattern that ensures long-term scalability and code maintainability. The inclusion of real-time status updates—such as dynamically shifting a motorcycle's status from "Available" to "Rented" upon transaction approval—directly mitigates the double-booking risks and data discrepancies highlighted in the introduction. This provides a significantly more robust and integrated solution than previous prototypes or basic SDLC implementations.

Furthermore, the integration of multi-layered security measures ensures the system is not only functional but also resilient against common cyber threats. Features such as role-based authorization, CodeIgniter Query Builder for SQL injection prevention, strict file upload validation, and XSS filtering guarantee data integrity and protect sensitive user information. This comprehensive security framework, combined with intuitive graphical user interfaces for both customers and administrators, significantly accelerates administrative workflows, reduces human error, and enhances the overall customer service experience.

Despite these achievements, the current system has certain limitations that present opportunities for future development. Currently, the payment verification process requires manual intervention by the administrator to validate uploaded payment proofs, which may introduce slight delays during peak rental periods. Future research could address this by integrating automated third-party payment gateways (e.g., Midtrans or Xendit) to enable instant, real-time transaction validation. Additionally, future iterations of this system could expand its capabilities by developing a companion mobile application (Android/iOS) for greater user accessibility and integrating GPS tracking modules to provide real-time geographical monitoring of rented motorcycles, thereby further elevating operational oversight and asset security.

## 4. CONCLUSION

This study successfully developed and implemented an Integrated Web-Based Motorcycle Rental Information System for Supermoto Motorcycle Rental to overcome the operational inefficiencies associated with manual data and transaction management. By utilizing the Rapid Application Development (RAD) method and the CodeIgniter 3 framework with a Model-View-Controller (MVC) architecture, the system provides a centralized, scalable platform for managing motorcycle fleets, customer profiles, rental transactions, payment processing, and operational reporting. The implementation of real-time vehicle availability tracking has effectively eliminated the risks of data discrepancies, delayed reporting, and double-booking schedules that previously hindered business operations. Furthermore, the integration of structured payment verification workflows and automated reporting features has significantly accelerated administrative processes and improved overall data accuracy. The comprehensive Black Box Testing results demonstrated a 100% success rate across all 15 test scenarios, confirming that the system is highly reliable, secure against common vulnerabilities, and fully aligned with user functional requirements.

Ultimately, this digital transformation equips Supermoto Motorcycle Rental with a systematic and efficient tool to enhance service quality, streamline business operations, and support data-driven decision-making. For future development, it is recommended to integrate automated third-party payment gateways (e.g., Midtrans or Xendit) to enable instant payment validation, develop a companion mobile application to broaden customer accessibility, and incorporate GPS tracking modules to provide real-time geographical monitoring of the rental fleet, thereby further elevating asset security and operational oversight.

## ACKNOWLEDGMENTS

The authors would like to express their gratitude to Universitas Cipasung Tasikmalaya and Supermoto Motorcycle Rental for supporting this research project.

## DECLARATIONS

**Author Contributions:**

Arrizqi Hilman Rahmatulloh: Conceptualization, Methodology, Software, Writing – Original Draft.
Robby Maududy: Data Curation, Validation, Supervision, Writing – Review & Editing.

**Funding:**

The authors received no financial support for the research, authorship, and/or publication of this article.

**Conflicts of Interest:**

The authors declare no conflict of interest regarding the publication of this paper.

**Data Availability:**

The data and source code that support the findings of this study are available from the corresponding author upon reasonable request.

**AI Use Statement:**

During the preparation of this work, the authors used Qwen AI in order to improve the language and readability of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the final content of the published article.

**Additional Information:**

No additional information is available for this paper.

## REFERENCES

[1] A. Nurmayanti, W. J. Lestari, A. Sevtiana, and R. Rahimah, "Aplikasi Akuntansi Penerimaan Kas Atas Sewa Alat Berat Pada PT. Jasa Transportasi Yala Githa Tama Cirebon," *Ekono Insentif*, vol. 15, no. 2, pp. 57–66, 2021, doi: 10.36787/jei.v15i2.495.

[2] P. A. Nugroho and D. Hernandi, "Perancangan Sistem Informasi Untuk Penyewaan Jasa Fotografi Berbasis Web Pada Appa Project," *Jris J. Rekayasa Inf. Swadharma*, vol. 4, no. 1, pp. 10–17, 2024, doi:

10.56486/jris.vol4no1.399.

[3] H. Yudiastuti, I. Irwansyah, F. Panjaitan, and D. Rumanti, "Sistem Informasi Sebagai Media Promosi pada Wedding Gallery Berbasis Website," *J. Softw. Eng. Ampera*, vol. 3, no. 2, pp. 84–98, 2022, doi: 10.51519/journalsea.v3i2.212.

[4] A. Subki, B. Imran, and S. Erniwati, "Pengembangan Sistem Informasi Geografis Berbasis Android Pada Wisata Daerah Lombok, Nusa Tenggara Barat," *Infotek J. Inform. dan Teknol.*, vol. 4, no. 2, pp. 259–269, 2021, doi: 10.29408/jit.v4i2.3667.

[5] S. Kuncoro and Ariansyah, "Rancang Bangun Sistem Informasi Pada Studio Foto Berbasis Web (Studi Kasus Arra Photo Cinema)," *ITeCS (Indonesian J. Inf. Technol. Comput. Sciense)*, vol. 1, no. 3, pp. 114–118, 2023.

[6] Y. Azhar, G. A. Mahesa, and M. C. Mustaqim, "Prediksi pembatalan pemesanan hotel menggunakan optimalisasi hiperparameter pada algoritme Random Forest," *J. Teknol. dan Sist. Komput.*, vol. 9, no. 1, pp. 15–21, 2021, doi: 10.14710/jtsiskom.2020.13790.

[7] A. A. Afolorunso, B. E. Aimuel, A. O. Abiodun, A. O. Adesina, and S. A. Ajagbe, "Development of a Machine Learning–Enabled Decision Support Framework for Hotel Booking Cancellation Prediction," *Adv. Multidiscip. Sci. Res. J. Publ.*, vol. 16, no. 3, pp. 23–36, 2025, doi: 10.22624/aims/cisdi/v16n3p2.

[8] H. Humaidi and I. F. Hanif, "Perancangan Sistem Informasi Keuangan dan Aset Berbasis Web (Studi Kasus: PT Informatika Media Pratama)," vol. 1, no. 1, pp. 25–33, 2022.

[9] R. Triyanto, "Rancang Bangun Aplikasi Penjualan Berbasis Website (Studi Kasus: Toko Waroeng Bola)," *J. Sist. Inf. dan Sains Teknol.*, vol. 2, no. 1, pp. 1–9, 2020, doi: 10.31326/sistek.v2i1.670.

[10] N. Aminudin and I. Susilo, "Perancangan Sistem Aplikasi Ujian Online Berbasis Web Pada SMA Negeri 1 Kalirejo," *Aisyah J. Informatics Electr. Eng.*, vol. 1, no. 1, pp. 81–88, 2019, doi: 10.30604/jti.v1i1.14.

[11] M. W. Tjia, J. Hendrik, and S. Time, "Web-Based Inventory System Design Using Rapid Application Development Method at PT. ALFA SCORPII," *J. Artif. Intell. Eng. Appl.*, vol. 4, no. 3, 2025.

[12] H. T. Sadiah, M. Saad, and N. Ishlah, "Design of the Inventory Application of CV Diva Karya Mandiri Using RAD (Rapid Application Development)," *Int. J. Quant. Res. Model.*, vol. 4, no. 2, pp. 82–89, 2023.

[13] A. Suriyana and L. Junaedi, "Rancang Bangun Sistem Informasi Penjualan Online (E-Commerce) pada Toko Cindyah Collection dengan Metode Rapid Application Development," *J. Adv. Inf. Ind. Technol.*, vol. 2, no. 2, pp. 1–9, 2020, doi: 10.52435/jaiit.v2i2.65.

[14] Sahidi and Z. Mutaqin, "Sistem Informasi Pariwisata Desa Senaru Berbasis Website Menggunakan Metode RAD," *J. Comput. Technol.*, vol. 1, no. 1, pp. 1–5, 2023.

[15] I. Dimentieva, E. Lutfina, G. W. Saraswati, and R. M. Caturkusuma, "Integrating RAD and Design Thinking for Developing a Web-Based POS and Inventory Management System for MSMEs: A Case Study," *Inf. J. Ilm. Bid. Teknol. Inf. dan Komun.*, vol. 11, no. 1, pp. 80–88, 2026.

[16] M. Syani, Y. P. Jabir, F. Laia, and E. A. Firdaus, "Sistem Informasi Manajemen Pengawasan Dan Pengendalian (WASDAL) Menara Telekomunikasi (Studi Kasus: Dinas Komunikasi Dan Informatika Kota Cimahi)," vol. 18, pp. 167–179, 2024.

[17] A. F. Qadafi and A. D. Wahyudi, "Sistem Informasi Inventory Gudang Dalam Ketersediaan Stok Barang Menggunakan Metode Buffer Stok," *J. Inform. dan Rekayasa Perangkat Lunak*, vol. 1, no. 2, pp. 174–182, 2020, doi: 10.33365/jatika.v1i2.557.

[18] H. Putri, F. Rini, and A. Pratama, "Salah satu teknologi yang berkembang adalah teknologi informasi, dapat dilihat dari banyaknya," *J. Pustaka Data*, vol. 2, no. 1, pp. 5–10, 2022.

[19] F. R. Mulyadi and Y. Syahidin, "Rancang Bangun Sistem Informasi Kepegawaian Dengan Metode Waterfall," *Explor. Sist. Inf. dan Telemat.*, vol. 12, no. 2, p. 186, 2021, doi: 10.36448/jsit.v12i2.2056.

[20] I. P. G. A. Putra, I. M. B. Adnyana, and D. R. Putri, "Rancang Bangun Ruang Baca Digital Berbasis Web Pada Perpustakaan Desa Mengwi," *J. Sutasoma*, vol. 2, no. 2, pp. 105–113, 2024, doi: 10.58878/sutasoma.v2i2.286.

[21] Subardin, E. Sekar Lembayung, F. Findryani, and S. Safal, "Perancangan Sistem Informasi Pengarsipan Surat (SIPS) Berbasis Web Di Kantor Kepolisian Daerah Sulawesi Tenggara," *J. Kecerdasan Buatan dan Teknol. Inf.*, vol. 3, no. 1, pp. 17–25, 2024, doi: 10.69916/jkbti.v3i1.106.

[22] C. Fauziah, F. Sawitri, F. N. Azahara, and H. Permatasari, "Implementasi Pengujian Form Transaksi Laporan Penjualan Sistem Kasir POS Codekop CV Darut Taqwa Ujung Harapan Menggunakan Metode White Box Testing Dan Black Box Testing," in *Pros. Sem. Nas. Teknol. Inf. dan Bisnis (SENATIB)*, 2023, pp. 456–464.

[23] A. S. Alfain, A. A. N. Fajrillah, and R. Hanafi, "Analisis Dan Perancangan Arsitektur Enterprise Pada Badan Kepegawaian Daerah (BKD) Pemerintah Provinsi Jawa Barat Menggunakan TOGAF ADM 9.2," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.)*, vol. 8, no. 1, pp. 127–139, 2023, doi: 10.29400/jipi.v8i1.3311.