**Capstone Project (CPRO306)**

**Assessment 3: Final SRS Report**

**Project Title:** **Jot Bikes Management System**

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **No.** | **Name** | **SID** |
| **1** | **Thi Ngoc Quynh Ho** | **K220380** |
| **2** | **Erna Halim** | **K221578** |
| **3** | **Hendra Lim** | **K220685** |
| **4** | **Alyxandra Marie Sarmiento** | **K220411** |
| **5** | **Thi Ngoc Quynh Ho** | **K220380** |

[**1. Project Description** 4](#_Toc195816265)

[**1.1. Business Case** 4](#_Toc195816266)

[**1.2. Purpose and Objectives** 4](#_Toc195816267)

[**1.3. Detailed Scope** 4](#_Toc195816268)

[**1.4. Proposed System Design Specifications** 5](#_Toc195816269)

[**1.5. Methodology** 5](#_Toc195816270)

[**2. System Analysis and Requirements** 6](#_Toc195816271)

[**2.1. Software and Hardware Requirements** 6](#_Toc195816272)

[**2.2. Functional and Non-Functional Requirements** 6](#_Toc195816273)

[**2.3. User Requirements (use cases, use case diagram)** 8](#_Toc195816274)

[**3. Database Design** 26](#_Toc195816275)

[**3.1. Entity Relationship Diagram (ERD)** 26](#_Toc195816276)

[**3.2. Data Flow Diagram (DFD)** 27](#_Toc195816277)

[**3.3. Security Implementation** 33](#_Toc195816278)

[3.3.1 Password hashing using Bcrypt 33](#_Toc195816279)

[3.3.2 AES-256 Encryption for data at rest 34](#_Toc195816280)

[3.3.3 Data Anonymization for Analytic 34](#_Toc195816281)

[3.3.4 Role-Based Access Control (RBAC) 34](#_Toc195816282)

[3.3.5 Additional Security Best Practice 34](#_Toc195816283)

[**4. User Interface Design** 34](#_Toc195816284)

[**4.1. User Interface Storyboard** 34](#_Toc195816285)

[**4.2. Input Data Forms** 37](#_Toc195816286)

[**4.3. Output Report Forms** 38](#_Toc195816287)

[**5. Test Plan & System Implementation Plan** 39](#_Toc195816288)

[**5.1 System Implementation Plan** 39](#_Toc195816289)

[**5.2 Test Plan** 40](#_Toc195816290)

[5.3 Data Protection Impact Assessment (DPIA) 40](#_Toc195816291)

[**6. Feedback** 41](#_Toc195816292)

[**7. References** 41](#_Toc195816293)

# **1. Project Description**

## **1.1. Business Case**

Jot Bikes is Australia’s leading provider of electric bikes and scooters, offering both short-term rentals and direct sales. As the business expands, its current manual and fragmented systems for managing bookings, inventory, vendors, and customer data have become inefficient. These issues lead to booking overlaps, stock mismanagement, and customer dissatisfaction. To maintain competitiveness and improve service delivery, Jot Bikes requires a modern, web-based management solution that automates internal operations and provides real-time access to key services for customers, administrators, and vendors. This solution follows industry-recognized best practices in web development, security, and data privacy compliance (OWASP 2024), (OAIC 2024).

## **1.2. Purpose and Objectives**

The purpose of the Jot Bikes Management System (JBMS) is to streamline operations and enhance customer and vendor experiences through digital transformation. The key objectives are:

* Centralised Management: Build a web-based system to manage bike inventory, rental and sales records, vendor listings, and customer accounts in a single platform.
* Automated Booking and Secure Payment: Enable customers to search, book, and securely pay online, improving convenience and accuracy.
* Customer Engagement (CRM): Integrate automated notifications, feedback systems, and booking history to foster better communication and retention.
* Vendor Management Module: Provide a dashboard for vendors to manage bike listings and track bookings in real time.
* Analytics and Reporting: Equip administrators with dashboards for monitoring rental/sales trends, customer behaviour, and vendor performance.

These objectives align with modern web development standards and ensure scalability and data protection in compliance with Australian regulations.

## **1.3. Detailed Scope**

* **Login and Registration**: Admins, Users and Vendors can login and register on JBMS (Jot Bikes Management System) by using secure and unique UserID/ email and password.
* **Manage:**
* **Vendor:** admins can update and view the vendors’ information, including add and delete them in the system.
* **Bike:** both admin and vendor can update and delete bikes’ information, however, only admin can add bike’s data such as: availability.
* **View:**
* **Booking:** both admin and vendor can view the booked products which had been done by the users.
* **Feedback:** admin can view the feedback from users and response.
* **Users:** admin can view the user’s information, including add, delete, and update
* **Users:**
* **Check bikes available:** user can check the number of products available for rent and sale, including search the products filters based on the specific brands, colours, price, etc.
* **Book and pay**: after browsing and checking all the necessary information, users can make a booking and pay online.
* **Encryption & Security:** some security features will be needed for this e-commerce website to protect the sensitive data and session cookies, such as: AES Security Encryption, firewall, SSL certificates, MFA, Payment Gateway, HTTPS. These technique align with best practices that is recommended by OWASP (OWASP 2024) and NIST (Grassi, Garcia & Fenton 2017).
* **Notification System:** the system will need notifications function for Admin, Vendor and User to remind them about the booking and payment to manage.

## **1.4. Proposed System Design Specifications**

The system will be developed using the MERN Stack. The MERN stack was selected for its unified use of JavaScript, which allow seamless integration between front-end and back-end. It also supports rapid prototyping, scalability, and is widely adopted in modern web application development (Vipul A. M; Prathamesh Sonpatki 2016). TailwindCSS is used for fast and responsive design (*Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* n.d.).

* Frontend: React.js with TailwindCSS for styling. Responsive design using grid/flex layouts and component-based structure. Axios will be used for API calls.
* Backend: Node.js with Express.js to provide RESTful API endpoints for secure communication, session handling, and validation.
* Database: MongoDB (hosted via MongoDB Atlas) to store bike data, user profiles, booking records, and vendor details in a flexible schema.
* Security: JWT-based session management, secure form validation, XSS/CSRF protection via middleware, and AES encryption for sensitive data.
* Deployment: Hosted on Vercel (frontend) and Render/Heroku or AWS (backend), with GitHub for CI/CD version control and collaboration.

## **1.5. Methodology**

The project follows an Agile methodology using the Scrum framework, enabling iterative development, weekly progress tracking, and flexible change handling. This includes:

* Sprint Planning: Defined goals and backlog grooming every two weeks.
* Daily Stand-ups: Regular team check-ins to track blockers and sync progress.
* Version Control: All work tracked using GitHub with branching and pull requests.
* Design and Prototyping: Wireframes and UI mockups built in Figma, tested iteratively with feedback from team and stakeholders.
* Testing: Unit, integration, and system testing conducted during development using Jest and Postman.
* Deployment & Review: Final system deployed to live environments with stakeholder walkthrough and UAT before final handover.
* Documentation: Admin and user manuals, installation guide, and technical documentation will be delivered at project completion.

# **2. System Analysis and Requirements**

## **2.1. Software and Hardware Requirements**

The hardware requirements for JBMS are:

* Processor: Development and testing require a system with intel core i5 multicore processor (at least), recommended i7.
* RAM: A minimum of 8 GB RAM is recommended.
* External storage: An external hard drive will be included as a backup storage with 256GB minimum.
* Other tools: Keyboard, mouse, monitor, printer, and scanner.

The software requirements for JBMS are:

* Operating system: Windows 10 or later version for Window system and at least MacOS 13 Ventura version or later version for MacBook.
* MongoDB: MongoDB provides a free open source called Community Server, which will be used for this system because it is publicly available to download Team (2025).
* A compatible web browser: Web browser is Chrome and Microsoft Edge recommended or Firefox for Window system and Safari for MacBook.

A live server of JBMS can be accessed by the web browser after deploying.

## **2.2. Functional and Non-Functional Requirements**

The functional requirements for Management and others in JBMS are:

* Inventory Management:
* Tracking Bikes: users can search and filter their search results based on their requirements with accurate information about the products, such as: condition, colour, gear, size, availability, and model.
* Adding/ Removing Bikes: admins and vendors can easily remove and add products into the system for rental and sale based on the stock available.
* Inventory alerts: the system will alert admin and vendor when the stock is low or any products need to be maintained.
* Rental Management:
* Booking and scheduling: user can book online or through the system with specific date, time, and type of products.
* Rental agreements: rental agreements will include customers’ information, rental terms, bikes’ details, date and time, will be generated and stored in the system.
* Payment processing: the payment method – such as cash, credit, check, etc. – and data will be secured in the system.
* Rental check-in and check-out: provide an efficient check-in and check-out process, including product conditions and verify user information before and after processing.
* Extension of rentals: user can easily extend the rental if needed by itself through online and in-store system, also for vendor and admin.
* Sales Management:
* Product catalogue: products for sales or rentals discounts on specific will be displayed with details, visuals and price.
* Sale transactions: Any sale transaction will be processed and stored in the system, including payment and other fulfilment.
* Customer database: customers’ information, including personal, payment and historical and future transactions will be stored in the system.
* Customer Relationship Management (CRM)
* Customer Profiles: customer information – such as: contact details, rental history, and relevant information – will be maintained and stored in the system.
* Communications: customers will receive automated emails and notifications from the company regarding bookings, payments, newsletters and other relevant important information.
* Reporting and Analytics:
* Sales reports: sales reports, including popular types of products, revenue, customer demographics and sales performance, will be generated and stored in the system.
* Rental reports: rental reports are used to track rental activity of the company, including popular rental periods, customer preferable, occupancy rates and popular rental products.
* Inventory reports: demonstrate the fast, slow and non-moving (FSN) products, levels of stock and maintenance information of products.
* Other functions
* Integration with online booking platforms: customer can book products through online booking system of the website.
* User roles and permissions: implement different user roles with multiple levels of access to the system.
* Security: JWT is used for session management, bcrypt for password hashing, and RBAC for enforcing access levels while also ensuring that the system follow OWASP and NIST best practices (OWASP 2024).
* Customization: JBMS is modular by design, which allow future enhancements such as vendor specific pricing models, rental extension policies, and promotional features without having to do system-wide changes.

The non-functional requirements of JBMS are:

* Responsiveness: The JBMS will be responsive on various devices with different dimensions.
* Privacy: prevent customers’ data from being accessed by unauthorized participants.
* Security: Implement necessary and newest security features to protect the sensitive information and session cookies.
* Scalability: The ability of the system to expand in the future when the growth of users, transactions, products, vendors, admins and volume of data if the company upgrade to a bigger size.
* Reliability: the performance of the system needs to be consistency.
* Quality: the system is performing effectively and bug-free.
* Documentation: documents that involve in the project such as user and technical guidelines, and specifications.
* Response Time: how fast the company response to the users’ actions and requests for example: live chat, emails, phone calls, social media, etc.
* Maintainability: the system can be easily updated, developed, improved and expanded for the future needs.
* Performance: the response time, and resources’ quality demonstrate the effectively and efficiency of the JBMS.

## **2.3. User Requirements (use cases, use case diagram)**

The use cases of JBMS demonstrate how the users (such as: admin, vendor, and customers) and stakeholders are involved in the system.

**Admin**

* Create id & password

|  |  |
| --- | --- |
| CREATE ID & PASSWORD Use Case | |
| Name: | Create id & password |
| Actor/Role: | Admin |
| Description: | Demonstrate how to create the users’ account |
| Successful completion: | 1. Log into the system. 2. Go to Users section. 3. Choose a New User button. 4. Add new User details. 5. Click create. |
| Alternative: |  |
| Precondition: | Admin should have the details of the user |
| Postcondition: | Users are customers and vendors |
| Assumptions: | User does not have an account. |

A group of white circles with text

AI-generated content may be incorrect.

Fig. Admin creating id & password use case

* Log into the system

|  |  |
| --- | --- |
| LOGIN Use Case | |
| Name: | Login |
| Actor/Role: | Admin |
| Description: | Demonstrate how admin log into the system through their account. |
| Successful completion: | 1. Admin logs into JBMS and is redirected to their Admin dashboard based on access role. 2. Admin can view booking, users’ details and feedback in the ‘View Bookings’, and ‘View Customers’ section. 3. Admin can manage vendor and products information in ‘Manage Vendor’ and ‘Manage Bikes’ section. |
| Alternative: |  |
| Precondition: | Admin have their own username, and password |
| Postcondition: | Admin log in into the dashboard to manage vendor and products, also view users’ details and feedback, and booking. |
| Assumptions: | * Admin want to manage stock, vendor and check for booking and feedback. |

A group of white circles with text

AI-generated content may be incorrect.

Fig. Admin login use case

* View bookings: View booked products.

|  |  |
| --- | --- |
| VIEW BOOKINGS Use Case | |
| Name: | View booking |
| Actor/Role: | Admin |
| Description: | Demonstrate how admin view the booked products. |
| Successful completion: | 1. Admin log into the system and access the "View Bookings" section successfully. 2. Admin view the list of booked products. 3. Admin can view the details of the bookings such as: start – end dates, number of booked products, type of products and customers details. |
| Alternative: |  |
| Precondition: | Admin should be logged in. |
| Postcondition: | Admin should be able to view the complaint. |
| Assumptions: | Customers booked the bikes. |

A group of white circles with text

AI-generated content may be incorrect.

Fig. Admin view booking use case

* View Users and Feedback: view users details and their feedback

|  |  |
| --- | --- |
| VIEW USERS and FEEDBACK Use Case | |
| Name: | View Users and Feedback |
| Actor/Role: | Admin |
| Description: | Demonstrate how admin view the users’ details and their feedback. |
| Successful completion: | 1. Admin log into the system and assess ‘Customers’ section successfully that included the customer’s details and feedback. 2. Admin view the customer’s details and feedback after accessing the ‘Customers’ section. 3. Admin can update the users’ details and reply to the feedback (if needed). |
| Alternative: |  |
| Precondition: | Admin should be able to view the customers’ details and feedback. |
| Postcondition: | Admin should be able to reply to the customers’ feedback |
| Assumptions: | Admin receive feedback from the customers. |

A group of white circles with text

AI-generated content may be incorrect.

Fig. Admin view customers’ details and feedback use case

* Manage Vendor: Admin can manage vendors information.

|  |  |
| --- | --- |
| MANAGE VENDOR Use Case | |
| Name: | Manage vendor |
| Actor/Role: | Admin |
| Description: | Demonstrate how admin manage vendor information |
| Successful completion: | 1. Admin log into the system and access the ‘Manage Vendor’ section successfully. 2. Admin view the details of all vendors. 3. Admin can update, delete and add vendors and vendors’ information. |
| Alternative: |  |
| Precondition: | Admin should be working for JB |
| Postcondition: | Admins can add or delete or update the vendor details. |
| Assumptions: | Vendors changed or vendor’s information has been changed. |

A group of white circles with text

AI-generated content may be incorrect.

Fig. Manage Vendor use case

* Manage Bikes: Admin can update, add and delete products (information).

|  |  |
| --- | --- |
| MANAGE BIKES Use Case | |
| Name: | Manage Bikes |
| Actor/Role: | Admin |
| Description: | Demonstrate how admin manage the products. |
| Successful completion: | 1. Admin logs into the system and access the "Manage Bikes" section successfully. 2. Admin can view and check the bikes information and conditions. 3. Admin can update, delete and add the bikes information and conditions. |
| Alternative: |  |
| Precondition: | Admin are on shift while making any change. |
| Postcondition: | Admin can view, update, delete and add information and condition for bikes. |
| Assumptions: | Admin have access to the system. |

A group of white circles with text

AI-generated content may be incorrect.

Fig. Manage Bikes use case

**Users**

* Registration: Sign up for member.

|  |  |
| --- | --- |
| REGISTRATION Use Case | |
| Name: | Registration |
| Actor/Role: | Customers |
| Description: | Demonstrate how the customer register to the membership |
| Successful completion: | 1. Customers access the registration page successfully. 2. Customers enter all the needed information (such as name, contact details, and address) through the registration form. 3. Customers submit the registration form and verify account through personal email. 4. The system creates an account successfully. |
| Alternative: |  |
| Precondition: | Customers should have all required details. |
| Postcondition: | Customers created an account with JBMS. |
| Assumptions: | Customers need to buy/ book any products from the shop. |

A group of symbols on a black background

AI-generated content may be incorrect.

Fig. Registration use case

* Check Bikes Availability: Search for available products to buy/ rent.

|  |  |
| --- | --- |
| CHECK BIKES AVAILABILITY Use Case | |
| Name: | Check Bikes Availability |
| Actor/Role: | Customers |
| Description: | Demonstrate how the customers check the availability of the needed products |
| Successful completion: | 1. Customers access the Main Page successfully. 2. Customers search for products that they want based on colours, conditions, sizes, brands, etc. through filters. 3. Customers can check for the products availability before making any booking/ buying. |
| Alternative: |  |
| Precondition: | Customers should know what types of products they want. |
| Postcondition: | Customers should be able to find the bikes by using the filter and understand and accept its condition and information. |
| Assumptions: | Customers know what type of bikes they want. |

A group of symbols on a black background

AI-generated content may be incorrect.

Fig. Check Bikes Availability use case

* Book and Pay: users can book and pay to rent the products.

|  |  |
| --- | --- |
| BOOK and PAY Use Case | |
| Name: | Book and Pay |
| Actor/Role: | Customers |
| Description: | Demonstrate how the customers book and pay for the products they want to rent/ buy on the website. |
| Successful completion: | 1. Customers will be led to the book and pay page after deciding the products they want to rent. 2. Customers fill the form with the payment information and agreements. 3. Customers click on the ‘book’ button and the system will save all the details and sent them to the admin dashboard. |
| Alternative: |  |
| Precondition: | Customers decided the products they want to rent. |
| Postcondition: | Customers are ready for booking. |
| Assumptions: | Customers are happy and understand all the conditions and information of the products. |

A group of symbols on a black background

AI-generated content may be incorrect.

Fig. Book and Pay use case

* View Profile: customers can update personal information and booking (if needed).

|  |  |
| --- | --- |
| VIEW PROFILE Use Case | |
| Name: | View profile |
| Actor/Role: | Customers |
| Description: | Demonstrate how the customers can view and update their information in the system. |
| Successful completion: | 1. Customers log into the system and access the "Profile" section. 2. Customer successfully view their personal details and make any update if needed. 3. Customer can check and update their booking (if needed) directly on the website. |
| Alternative: |  |
| Precondition: | Customer should have an account and able to log into the system. |
| Postcondition: | Customers are able to view and update their personal details and bookings. |
| Assumptions: | Customers want to check their personal details and bookings. |

A black background with white text

AI-generated content may be incorrect.

Fig. View Profile use case

* Log into the system

|  |  |
| --- | --- |
| LOGIN Use Case | |
| Name: | Login |
| Actor/Role: | Customers |
| Description: | Demonstrate how customer log into the system through their account. |
| Successful completion: | 1. Customer log into JBMS and is redirected to their User dashboard based on access role. 2. Customers can search and check the products availability. 3. Customers can book and pay for the products after satisfying with everything. |
| Alternative: |  |
| Precondition: | Customers have their own username, and password |
| Postcondition: | Customers log in into the dashboard where they can view and update their profile and booking. |
| Assumptions: | * Customers want to book/ manage personal or booking details. |

A group of symbols on a black background

AI-generated content may be incorrect.

Fig. Customers login use case

**Vendor**

* Log into the system

|  |  |
| --- | --- |
| LOGIN Use Case | |
| Name: | Login |
| Actor/Role: | Vendors |
| Description: | Demonstrate how vendor log into the system through their account. |
| Successful completion: | 1. Vendors. log into JBMS and is redirected to their Vendor dashboard based on access role. 2. Vendor can view bookings and manage bikes in their dashboard. |
| Alternative: |  |
| Precondition: | Vendors have their own username, and password |
| Postcondition: | Vendors log in into the dashboard for viewing bookings and managing bikes. |
| Assumptions: | * Vendors want to view booking and manage products. |

A black background with white text

AI-generated content may be incorrect.

Fig. Vendor login use case

* View bookings: View booked products.

|  |  |
| --- | --- |
| VIEW BOOKINGS Use Case | |
| Name: | View booking |
| Actor/Role: | Vendor |
| Description: | Demonstrate how vendor view the booked products. |
| Successful completion: | 1. Vendor log into the system and access the "View Bookings" section successfully. 2. Vendor view the list of booked products. 3. Vendor can view the details of the bookings such as: start – end dates, number of booked products, type of products and customers details. |
| Alternative: |  |
| Precondition: | Vendor should be logged in. |
| Postcondition: | Vendor should be able to view the complaint. |
| Assumptions: | Customers booked the bikes. |

A black background with white text

AI-generated content may be incorrect.

Fig. Vendor view booking use case

* Manage Bikes: Vendor can update, add and delete products (information).

|  |  |
| --- | --- |
| MANAGE BIKES Use Case | |
| Name: | Manage Bikes |
| Actor/Role: | Vendor |
| Description: | Demonstrate how vendor manage the products. |
| Successful completion: | 1. Vendor logs into the system and access the "Manage Bikes" section successfully. 2. Vendor can view and check the bikes information and conditions. 3. Vendor can update, delete and add the bikes information and conditions. |
| Alternative: |  |
| Precondition: | Vendor are on shift while making any change. |
| Postcondition: | Vendor can view, update, delete and add information and condition for bikes. |
| Assumptions: | Vendor have access to the system. |

A black background with white text

AI-generated content may be incorrect.

Fig. Manage Bikes use case

* + 1. Use Cases

**Admin**

A screenshot of a computer

AI-generated content may be incorrect.

Fig. Admin use cases

**Vendor**

A diagram of a company

AI-generated content may be incorrect.

Fig. Vendor use cases

**Customer**

A diagram of a person with white text

AI-generated content may be incorrect.

Fig. Customer use cases

Summary

|  |  |
| --- | --- |
| Role | Permissions |
| Admin | Manage users, vendors, bikes, bookings, and feedback |
| Vendor | Add/edit bikes, view own bike bookings |
| User | Check bikes availability, make bookings, view profile |

# **3. Database Design**

## **3.1. Entity Relationship Diagram (ERD)**

A screenshot of a computer

AI-generated content may be incorrect.

Fig. ER Diagram

The database is designed using a NoSQL schema structure using MongoDB. Collections such as Users, Products, Bookings, and Feedback are stored as a JSON-like documents, allowing schema flexibility and fast retrieval. This document-based design is ideal for application like JBMS where field may vary such as vendor specific bikes and scalability is crucial Team (2025).

A diagram of a computer

AI-generated content may be incorrect.

Fig. Class Diagram

## **3.2. Data Flow Diagram (DFD)**

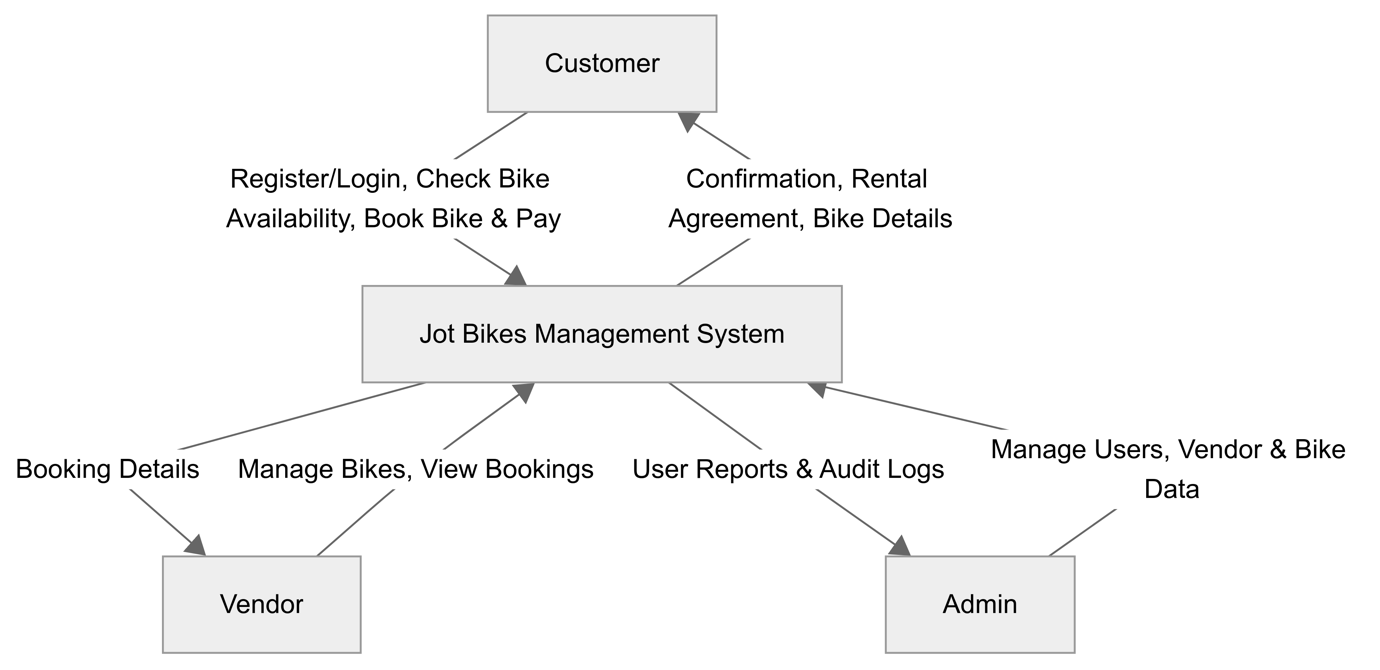


Fig. Data Flow Diagram Level 0

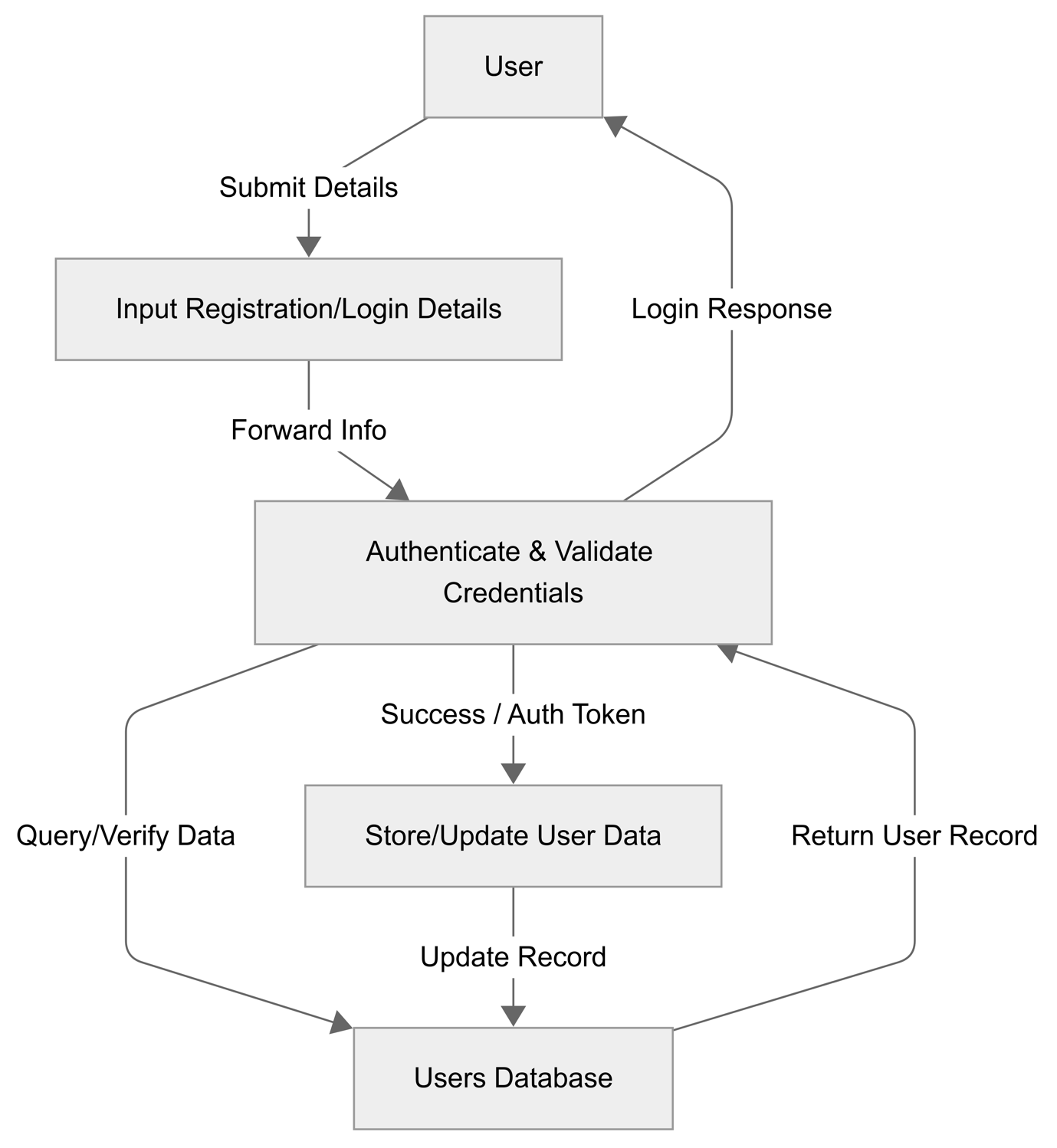


Fig. Data Flow Diagram Level 1 - User Registration and Login

A diagram of a computer

AI-generated content may be incorrect.

Fig. Data Flow Diagram Level 1 – Check Bike Availability

A diagram of a business

AI-generated content may be incorrect.

Fig. Data Flow Diagram Level 1 – Book and Pay

A diagram of a vendor

AI-generated content may be incorrect.

Fig. Data Flow Diagram Level 1 – Vendor Manage Bikes

A diagram of a vendor

AI-generated content may be incorrect.

Fig. Data Flow Diagram Level 1 – Vendor View Bookings

A diagram of a system

AI-generated content may be incorrect.

Fig. Data Flow Diagram Level 1 – Admin Manage Vendor

A diagram of a computer program

AI-generated content may be incorrect.

Fig. Data Flow Diagram Level 1 – Admin Manage Bikes

Relationships between entities such as User to bookings, product to category are enforced using MongoDB ObjectID references. For example, each booking document include user\_id and product\_id fields that store references to ObjectID in their respective collections. This allow MongoDb to maintain referential integrity across collections even without relational constraint (MongoDB 2024).

## **3.3. Security Implementation**

CIA (Confidentiality, Integrity, and Availability) must be adhered to when designing and implementing a website. For Jot Bikes Rental Management System, a multi-layered security approach has been implemented. This system consists of BCrypt which is used for secure password hashing, AES-256 encryption for sensitive data, data anonymization for privacy-compliant analytics, and RBAC (Role-Based Access Control) to ensure user has access based on their assigned role. Additionally, following industry best practice as recommended by OWASP and NIST. Secure Communication (HTTPS/SSL), environment-based key management, and having a proper admin activity login for audit trail will also be implemented. Precaution has been taken to ensure that JBMS is compliance and resilience in an unlikely event of a security breach.

### **3.3.1 Password hashing using Bcrypt**

All password in JBMS is stored using the BCrypt hashing algorithm as it is highly resistant to brute-force and rainbow table attacks. BCrypt adds automatic salting to prevent identical passwords having the same hash and it is supported by all major language and framework this is highly recommended by OWASP for securely storing password OWASP (2024).

Implementation

* User password are hashed during registration using bcrypt.hashpw().
* The salted hash result is stored in the database.
* During login, the password is verified using bcrypt.checkpw().

### **3.3.2 AES-256 Encryption for data at rest**

In this modern era, it is the utmost importance to safeguard sensitive user information or PII (Personally Identifiable Information) and payment metadate. JBMS will be using the AES-256 encryption as it is recognized globally for its strength and speed this is also recommended by NIST for protection of data at rest (Grassi, Garcia & Fenton 2017).

Implementation

* Data will be encrypted using AES-256 algorithm via the backend.
* Ensure that decryption will only be permitted after role-based authentication is validated.

### **3.3.3 Data Anonymization for Analytic**

To maintain user privacy, PII (Personally Identifiable Information) such as names, email addresses, and contact number must be anonymized. This is achieved through pseudonymization which will replace name with a coded identifier and email masking. Which will convert [K11111@student.kent.edu.au](mailto:K11111@student.kent.edu.au) to [K\*\*\*\*\*@student.kent.edu.au](mailto:K*****@student.kent.edu.au) , this will align with global data protection standards for privacy-enhancing data de-identification.

### **3.3.4 Role-Based Access Control (RBAC)**

RBAC ensure that user only access feature that is relevant to their role as defined by NIST RBAC model (Sandhu ' et al. 1996).

* Admin: Full access to the whole system
* Vendor: Access to the bike catalogue to add, update, and delete
* User: Access to their profile, manage booking, and provide feedback.

### **3.3.5 Additional Security Best Practice**

With the evolving security landscape additional security measure will also be implemented to build a future-proof system.

* Secure Communication (HTTPS/SSL): network communication is conducted over HTTPS using TLS encryption to prevent any data leakage and interception during transit.
* Admin activity logging: administrative actions are being logged to ensure accountability, operational transparency and preparation in case of future audit this aligns with ISO/IEC 27001:2022 standards for security auditing (ISO 2022).

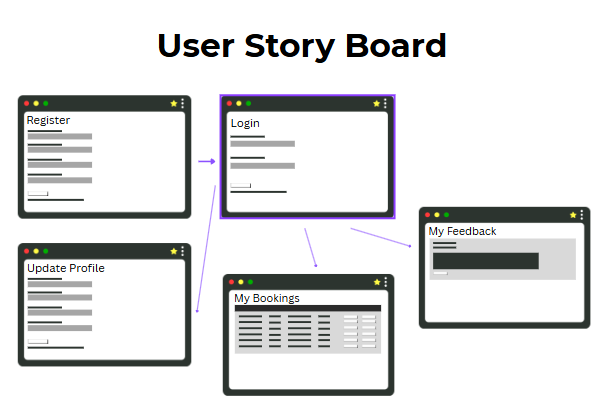
# **4. User Interface Design**

## **4.1. User Interface Storyboard**

This story board illustrates how different types of users will interact with the system. It shows different screen representations on how the system works. The UI is developed using React.js with Tailwind CSS. Tailwind provides responsive grid and flex layouts, which improve mobile responsiveness and consistency in design. Below is the storyboard of Jot Bikes System storyboard.







## **4.2. Input Data Forms**

Client-side form validation is implemented using React Hook Form. Common validations include required fields, email format, and inline error display. Following the data created in MongoDB by NoSQL database the input form will be as following:

Pre-AES

Users

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| user\_id | username | email | password | isAdmin | isVendor | createdAt | UpdatedAt |

Products

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| product\_id | description | brand | price | category | countIn  Stock | image | reviews | Num  Reviews | createdAt | UpdatedAt |

Categories

|  |  |
| --- | --- |
| category\_id | name |

Product Bookings

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| bookings\_id | user | product | startDate | ednDate | status | totalPrice | createdAt |

Feedbacks

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| feedback\_id | user | type | subject | message | status | priority | category | attachments | createdAt | updatedAt |

## **4.3. Output Report Forms**

Following the data created in MongoDB by NoSQL database the input form will be as following:

Post AES

Users:

{

"\_id":{"$oid":"67f2fa6bcdd0be224f4af416"},

"username":"erna",

"email":"erna@gmail.com",

"password":"$2b$10$jVn4z85CFj61pkbbVeRiYeaNaKTtBtbvrMfmPJPKKpziZDATp7a v2",

"isAdmin":true,

"createdAt":{"$date":{"$numberLong":"1743977067696"}},

"updatedAt":{"$date":{"$numberLong":"1743977067696"}}

,"\_\_v":{"$numberInt":"0"}

}

Products:

{

"\_id":{"$oid":"67faf5c28ce8a8ca8e4d4269"},

"name":"Segway",

"description":"The Segway electric scooter offers a sleek design with intelligent features like cruise control and mobile app connectivity. Its smooth acceleration and long battery life make it perfect for zipping around town or commuting with ease.",

"brand":"Segway",

"price":{"$numberInt":"85"},

"quantity":{"$numberInt":"1"},

"category":{"$oid":"67f8920156194929d9b145bd"},

"countInStock":{"$numberInt":"20"},

"image":"/uploads/1744751015355-Bike1.webp",

"numReviews":{"$numberInt":"0"},

"reviews":[],

"createdAt":{"$date":{"$numberLong":"1744500162069"}},

"updatedAt":{"$date":{"$numberLong":"1744751015357"}},

"\_\_v":{"$numberInt":"0"}

}

Categories:

{

"\_id":{"$oid":"67f8920156194929d9b145bd"},

"name":"Bike Rental",

"\_\_v":{"$numberInt":"0"}

}

Product Bookings:

{

"\_id":{"$oid":"67fed568d59f6ddf5d8fff0f"},

"user":{"$oid":"67fde730e28f040be6cf6c53"},

"product":{"$oid":"67fb468a958a3ef3a02ae3ae"}

,"startDate":{"$date":{"$numberLong":"1744761600000"}},

"endDate":{"$date":{"$numberLong":"1745366400000"}},

"status":"pending",

"totalPrice":{"$numberInt":"80"},

"createdAt":{"$date":{"$numberLong":"1744754024996"}}

,"\_\_v":{"$numberInt":"0"}

}

Feedbacks:

{

"\_id":{"$oid":"67ff04a744fe755c3a232434"},

"user":{"$oid":"67f2fa6bcdd0be224f4af416"},

"type":"general",

"subject":"Exceptional",

"message":"Exceptional service and user experience. Keep it up!",

"status":"pending",

"priority":"low",

"category":"website",

"attachments":[],

"createdAt":{"$date":{"$numberLong":"1744766119138"}},

"updatedAt":{"$date":{"$numberLong":"1744766119138"}},

"\_\_v":{"$numberInt":"0"}

}

# **5. Test Plan & System Implementation Plan**

## **5.1 System Implementation Plan**

The System Implementation plan is divided to 5 phase from development to production.

Phase 1 (Environment Setup)

* Timeline of 1-2 week
* Deploy frontend (React + TailwindCSS)
* Deploy backend (Node.js + Express)
* Configure MongoDB Atlas with access restrictions and backups
* Enable environment variables and secure communication (HTTPS)

Phase 2 (System Configuration)

* Timeline of 1 week
* Set up user roles and permissions: Admin, Vendor and User
* Configure bike categories, payment options, and rental rules.
* Ensure data is displaying as intended in the website

Phase 3 (User Training)

* Timeline of 1 week
* Host a live demo session to the member of the different access, and how to populate the data.

Phase 4 (System Testing)

* Rigorously perform testing to identify and resolve any bugs before going live
* Perform UAT (User Acceptance Testing) to gather feedback from the user and make relevant changes.
* Timeline of 2-3 week

Phase 5 (Go-Live and Post launch support)

* Communicate launch details to stakeholder
* Create a communication channel for support tickets, feedback, and error logs.

## **5.2 Test Plan**

Test plan is very crucial due to the timeline of this project to validate core system functions which include login, booking, payments, and role-based access. Test plan also ensure that it will identify and resolve any bugs before release and ensure system reliability, security, and usability.

Phase 1 (Unit Testing)

Tests individual React component, API endpoints, and backend logic in isolation.

|  |  |
| --- | --- |
| Module | Sample Tests |
| Login/Register | Reject invalid emails, hash passwords, MFA enforcement |
| Booking | Prevent double bookings, validate time slot conflicts |
| Inventory API | Return correct filtered results, handle out of stock cases |

Phase 2 (Integration Testing)

Tests how modules interact, such as flow between user booking and payment gateway.

|  |  |
| --- | --- |
| Integration | Sample Scenario |
| Booking & Inventory | Inventory updates after a confirmed booking |
| Vendor Management | Vendor can view and manage their own listings |
| Payment Gateway | Successful payment triggers booking confirmation email |

Phase 3 (System Testing)

Tests the entire system for end-to-end functionality.

* Perform an actual booking from signing up, search, book and pay.
* Ensure consistent display from computer/laptop to mobile devices and tablets

Phase 4 (User Acceptance Testing)

* Gather qualitative feedback from user via interview

## **5.3 Data Protection Impact Assessment (DPIA)**

Since the project is based on Sydney Australia it is important to comply with local regulations such as Australian Privacy Act 1988 and Australian Privacy Principles (APPs), a Data Protection Impact Assessment has been conducted for JBMS this is to ensure that JBMS are safeguarding sensitive user data. The data are processed to enable bookings, payments, and rental operations. Also to authenticate users and manage roles securely.

Data Collected

|  |  |
| --- | --- |
| Data Type | Example fields |
| Identity Data | Full Name, Email, Phone, Account Role |
| Transactional Data | Booking Dates, Payment Status, Product Info |
| Technical Data | IP Address, Device info, Login Session Tokens |

Identified Risks and Mitigation/Security Control for each risk

|  |  |  |
| --- | --- | --- |
| Risk | Impact | Mitigation/Security Control |
| Unauthorised access to PII | High | RBAC (Role Based Access Control) |
| Intercept data in transit | High | HTTPS |
| Weak password storage | High | Password hashing with bcrypt |
| Data loss from server error | Medium | Automated backups of MongoDB Atlas |
| Over Collection of data | Medium | Minimal collection of required data for booking |
| Insider misuse | High | Admin activity logging |

Legal and Ethical Compliance

* Ensure it is following Australian Privacy Act (1998) and APPs 1-13 OAIC (2024)
* Ensure user are providing consent before data collection
* User can request account/data deletion
* Ensure data are safely stored and not share to any third-party without consent
* Privacy policy are publicly available on the website platform

# **6. Feedback**

Based on the feedback received from the lecturer, changes have been made accordingly.

1. Feedback: “Provide description for each table and diagram not just pasting the picture in without any explanation”

Action taken: Brief explanation has been provided to ensure the reader know and understand what the purpose of the picture, table and diagram is.

1. Feedback: “Start working on the actual website development based on this report”

Action taken: Website has been generated based on the content of the document though it is still a rough start, but progress has been made.

1. Feedback: “Ensure adequate reference is provided”

Action taken: Beside the 5 minimum reference as required by the assignment adequate amount of reference has been included to ensure the document are citing relevant sources.

Link to GitHub Repository: <https://github.com/HendraLim1/ProjectCapstone/blob/28d3a4b2c355a4da7fefa8fa0c972255912246fe/WIP%20Report/Group%204%20Final%20SRS.docx>

Link to Website: <https://jot-bikes.vercel.app/>

# **7.** **References**

Grassi, PA, Garcia, ME & Fenton, JL 2017, ‘Digital Identity Guidelines’, *NIST Special Publication 800-63-3*.

ISO 2022, *ISO/IEC 27001 standard – information security management systems*, ISO.

MongoDB 2024, *MongoDB Documentation*, https://github.com/mongodb/docs-bi-connector/blob/DOCSP-3279/source/index.txt.

OAIC 2024, *The Privacy Act*, OAIC, Australian Government.

OWASP 2024, *OWASP Top Ten*, owasp.org, OWASP.

Sandhu ', R, Coyne, E, Feinstein, H & Youman, C 1996, ‘Role-Based Access Control Models’, *IEEE Computer*, vol. 29, no. 2, pp. 38–47.

*Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* n.d., tailwindcss.com.

Team, MD 2025, *Install MongoDB Community Edition*, Mongodb.com, viewed 14 April 2025, <https://www.mongodb.com/docs/manual/administration/install-community/>.

Vipul A. M; Prathamesh Sonpatki 2016, *ReactJS by Example - Building Modern Web Applications with React*, Packt Publishing.