IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.15498

Enhancing Shared Mobility Through AI-Driven Bike Rental Platforms

Prof. Diksha Bansod[1], Aaditi Katole[2], Janvi Aher[3], Sumit Ghoshal[4], Jay Ingole[5]

Professor Department of Computer Science and Engineering,

Nagarjuna Institute of Engineering Technology and Management, Nagpur, Maharashtra, India[1]

UG Student, Department of Computer Science and Engineering,

Nagarjuna Institute of Engineering Technology & Management, Nagpur, Maharashtra, India[2-5]

Abstract: The increasing demand for efficient urban mobility has led to the rise of digital platforms that provide affordable and flexible transportation options. Bike rental systems, in particular, offer a convenient solution for short-distance travel without the burden of ownership costs. *RentBike* is designed as a comprehensive web-based platform that connects riders, bike owners, and administrators within a unified system. The backend is developed using Flask to handle server-side operations efficiently, while MongoDB Atlas provides secure and scalable cloud-based data storage. The frontend is built using Tailwind CSS, ensuring a responsive and adaptive interface across different devices such as smartphones and desktops.

The platform integrates multiple functionalities including user authentication, bike search, booking management, payment processing, and review systems. Role-based access control ensures that users, vendors, and administrators have distinct permissions and capabilities within the system. Additionally, the platform incorporates a recommendation mechanism that analyzes user behavior and booking history to suggest suitable bikes. This personalized approach enhances user experience by presenting relevant options based on past interactions. The system operates seamlessly in the background, dynamically adapting to user roles and preferences without requiring explicit intervention.

From a technical perspective, the system is built using a modular architecture where different components interact through well-defined interfaces. Security is maintained using token-based authentication, while optimized database queries improve search performance and response time. The responsive design ensures accessibility across multiple devices, contributing to a smooth user experience. Performance testing indicates that the system can handle high loads efficiently while maintaining reliability and speed. Overall, RentBike demonstrates how modern web technologies can be leveraged to create scalable and intelligent mobility solutions, with potential future enhancements such as real-time tracking, dynamic pricing, and IoT integration.

Keywords: Flask Framework, JWT Authentication, Urban Mobility, RESTful APIs

I.INTRODUCTION

City movement is speeding up, pushing the need for transport that works well, adapts quickly, yet relies on smart tech. Because buying cars feels heavier now - pricier, clogged roads, harm to nature - people lean more toward sharing rides instead. From this shift, renting bikes has grown into a cleaner, simpler way to cover small trips across town. What follows unpacks how RentBike was built a complete online platform made with Flask, MongoDB Atlas, and Tailwind CSS to let users rent bicycles smoothly through a website.

Bikes sit idle too often. A digital space connects those who own them with people wanting to ride. Searching happens through filters - location matters, so does kind and cost. One click secures a booking. Owners track what's available, update listings, handle reservations. Oversight falls to admins; they check identities, monitor activity patterns. Behind the scenes, tools like RESTful APIs move data smoothly. Security relies on JWT methods, keeping access controlled. Information lives in cloud databases, built to grow without breaking. Web tech forms the backbone, silent but steady. Everything works together, piece by piece.

What stands out about the system Its smart suggestion tool studies how users book to recommend suitable bikes. On top of that, features like handling payments, managing feedback, and confirming user identity help keep things dependable. Trust grows when details are checked.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

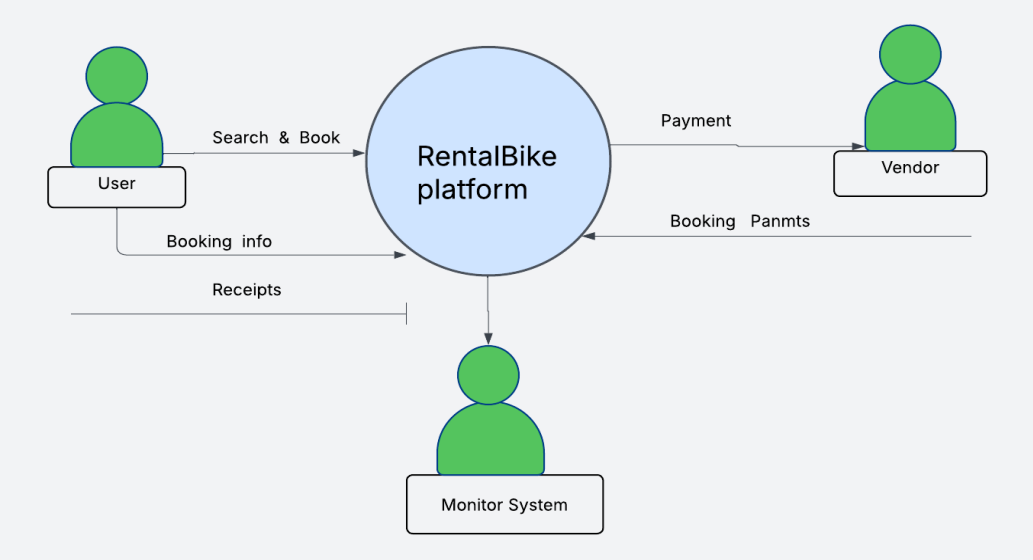
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.15498

Among those involved, smooth communication turns out possible thanks to thought-through blueprints, structured information setups, alongside coordinated task flows. Performance checks happen under conditions close to actual operation, measuring responsiveness, ease of use, scalability at once. One unified setup pulls together diverse tools - RentBike shows what happens when coding covers every layer from front to back. What results tackles everyday movement issues without extra complexity.

II.METHODOLOGY

Starting off, RentBike works across every layer of a website to help city dwellers get around without spending much. Instead of juggling separate tools, people who ride bikes, those who rent them out, and staff share one central space online. Logging in protects personal details. From there, browsing available bikes leads into locking down rentals, passing payments securely, leaving feedback afterward too. Each piece fits together so nothing feels clunky when jumping from screen to screen. Underneath it all, layers split tasks neatly - Flask handles behind-the-scenes actions, databases live safely in the cloud through MongoDB Atlas, while visuals adjust instantly thanks to Tailwind CSS. Depending on your role, you see only what matters, access stays limited appropriately, plus data moves quietly but reliably via standardized links between parts. Smoothness comes from how pieces talk, not just what they do alone. From start to finish, Python shaped how the system came together, backed by tools such as VS Code, Git, and Postman. Each piece was checked carefully - API behavior first, then how parts connect, followed by speed under load - to build confidence it works. When everything lines up, RentBike stands ready: built to grow, keep data safe, stay simple to use, and handle actual rental needs.



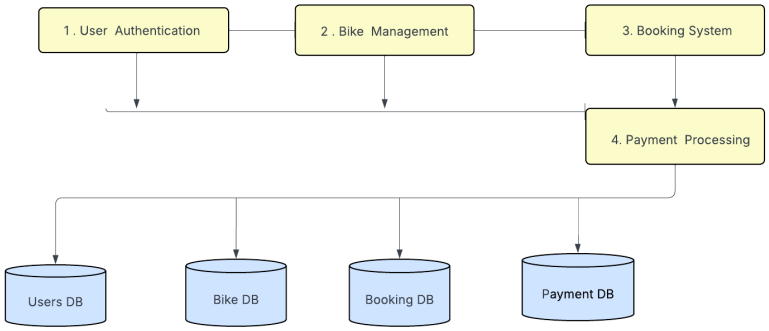
Requirement Analysis: Looking into what people want comes first when building the RentBike setup. This step shapes how the service works so it fits real city travel needs well. People riding bikes must sign up, log in, find cycles, reserve them, pay, then share thoughts afterward. One group adds bikes to the list and checks who booked them. Another watches everything happening across the app plus confirms rider details using official checks. Affordable short-term rides matter more now in busy areas. Each piece grows from what these groups actually do day by day. What the system does matters, yet how it behaves counts just as much. Handling many people at once without slowing down is key, while staying locked tight with login checks keeps things safe. Speed shows up in quick replies when tasks run, besides smooth navigation no matter which device someone uses.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

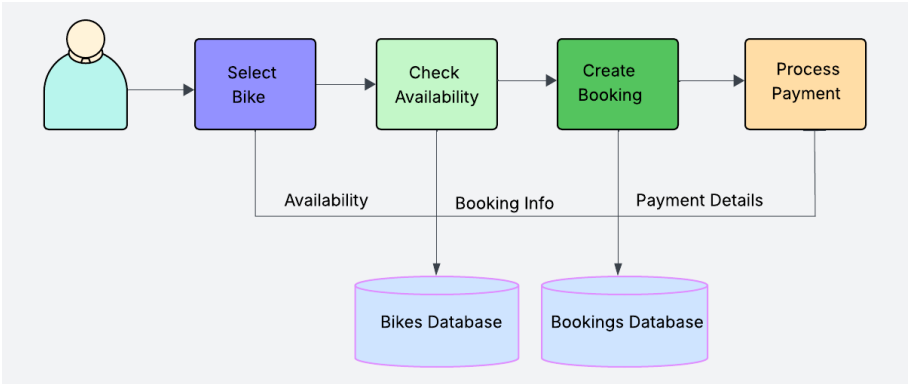
International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

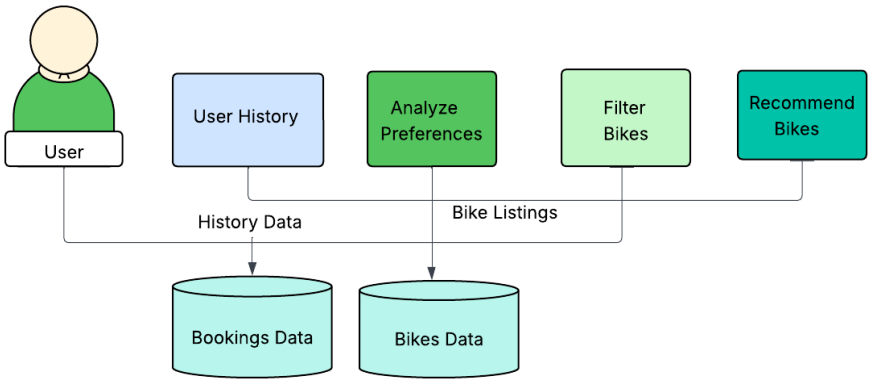
DOI: 10.17148/IJARCCE.2026.15498



System Design: The system design of RentBike follows a three-tier architecture consisting of the presentation layer, application layer, and data layer. The presentation layer is developed using HTML, JavaScript, and Tailwind CSS, providing a responsive and intuitive user interface. The application layer is implemented using Flask, which handles business logic and API communication. The data layer uses MongoDB Atlas, a cloud-based NoSQL database that ensures flexible and scalable data storage. The system is designed using a modular approach, dividing functionalities into separate components such as authentication, bike management, booking, payments, reviews, and administration. RESTful APIs are used to enable communication between the frontend and backend. Additionally, role-based access control is implemented to manage permissions for users, vendors, and administrators. The system also includes a recommendation mechanism that analyzes user behavior to provide personalized suggestions.



Development Environment: The development environment for the RentBike system includes a combination of modern tools and technologies that support efficient development and deployment. The backend is built using Python and the Flask framework, along with libraries such as flask-jwt-extended for authentication, pymongo for database interaction, and bcrypt for password hashing. MongoDB Atlas is used as the cloud-based database, allowing secure and scalable data storage. The frontend is developed using HTML, CSS, and JavaScript, with Tailwind CSS used for responsive design and styling. Development tools such as Visual Studio Code are used for coding, Git is used for version control, and Postman is used for testing APIs. The application runs locally using a Flask development server and can be served via a simple HTTP server for frontend testing.



IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.15498

Testing and Implementation: The testing and implementation phase ensures that the RentBike system functions correctly and efficiently under various conditions. Different types of testing are performed, including unit testing for individual API endpoints, integration testing to verify interaction between modules, and user interface testing to ensure proper functionality of frontend components. Performance testing is conducted to evaluate the system's ability to handle multiple users and maintain fast response times. Error handling mechanisms are implemented to manage invalid inputs and unexpected failures gracefully. During implementation, the database is seeded with sample data to simulate real-world usage. The system is deployed locally for testing and is prepared for production deployment with necessary configurations. Overall, this phase ensures that the system is reliable, secure, and capable of providing a smooth user experience.

III.MODELLING AND ANALYSIS

Inside RentBike, design spreads across three levels: display, operations, then storage. User screens make up the outer face, letting people engage via websites. Flask runs the middle section, sorting rules and managing incoming calls. Information flows into MongoDB Atlas at the base, kept ready for quick access**.**

Every piece of information fits into groups like people, bicycles, reservations, feedback, or money transactions. Because each item carries its own ID tag, connections form naturally across categories. A reservation ties a person to a bicycle ride. Once that happens, any comment left points back to that specific rental moment.

From the start, data moves through REST-style links connecting user screens to server functions. Instead of one path, each address manages its own job - adding reservations, moving money tasks, pulling bicycle facts. Speed gets a boost when searches use smart sorting, quick picks, or narrowed fields behind the scenes.

Out of past rentals, habits begin to show. That pattern feeds into a system built to guess what ride fits best. Preferences around cost come alive through math that watches spending over time. Suggestions pop up only after sorting through those details quietly behind the scenes.

Looking at how fast the system responds, grows, and stays steady is key. Because it runs on cloud platforms, heavier traffic won't slow things down much. Built in separate pieces, adding fresh functions feels natural. Change comes easier since parts connect cleanly. What works now keeps working later.

IV.RESULTS AND DISCUSSION

The RentBike system demonstrates how multiple technology layers work together to deliver a smooth and efficient bike rental service. During testing, it successfully handled core operations such as user registration, bike search, booking, and payment processing without performance issues, even under multiple simultaneous requests. Optimized APIs and efficient database management using MongoDB Atlas ensure fast response times and secure data handling. The responsive frontend design further enhances user experience by adapting seamlessly across different devices, making navigation simple and intuitive.

The system also provides personalized bike recommendations based on user history, while new users are shown popular options, improving decision-making and engagement. Role-based access ensures that users, vendors, and administrators have appropriate controls, with vendors managing bikes and bookings and admins overseeing platform activities. Although features like real-time tracking, dynamic pricing, and live payment gateways are not yet implemented, the system remains stable, reliable, and scalable. Overall, RentBike proves to be a consistent and efficient platform capable of supporting real-world bike rental services with strong potential for future enhancements.

V.CONCLUSION

The study presents RentBike as a comprehensive web-based platform developed to address the growing need for flexible and efficient urban transportation. Built using modern technologies such as Flask, MongoDB Atlas, and Tailwind CSS, the system integrates multiple features into a seamless environment. Its design focuses on real user needs, ensuring smooth interaction across all layers. The modular architecture allows different components to function independently while maintaining overall system harmony, supported by secure authentication and efficient data handling.

The platform incorporates role-based access control, enabling users, vendors, and administrators to perform specific tasks efficiently. Features such as booking management, secure login systems, and intelligent recommendation mechanisms

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

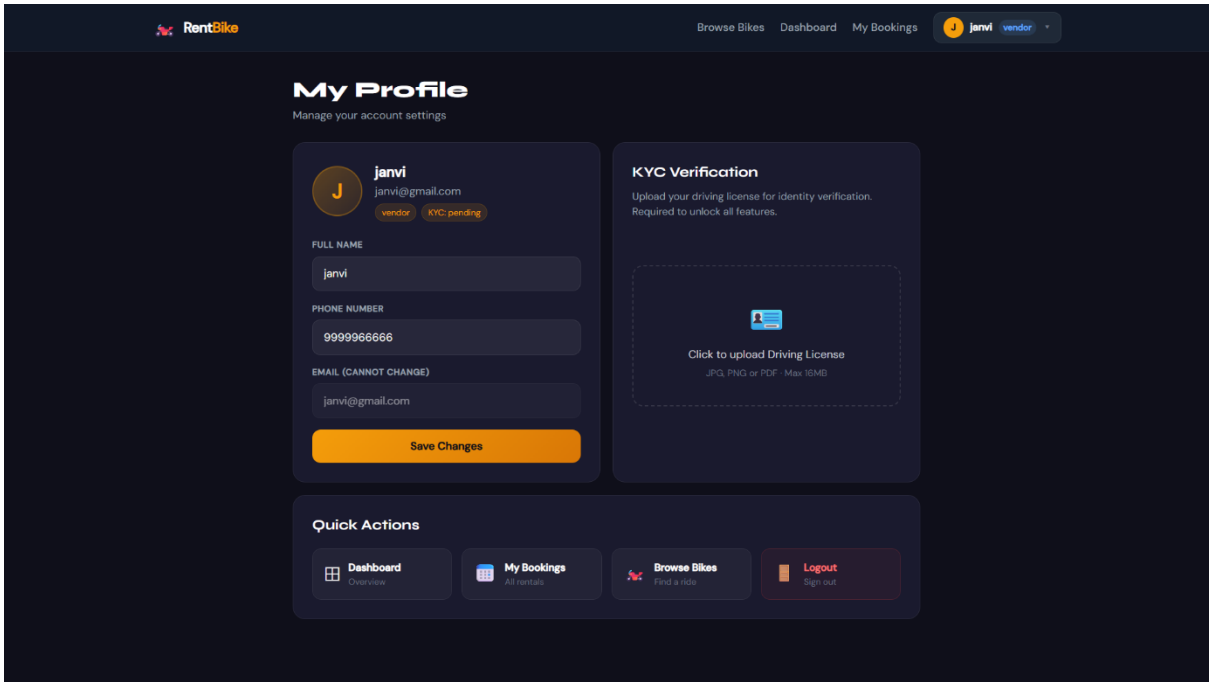
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.15498

enhance usability and system performance. The recommendation system, driven by user behavior, improves engagement by providing relevant suggestions. Testing results indicate that the system maintains high speed and reliability while handling data effectively, demonstrating its capability to support real-world applications.

Looking ahead, the system has strong potential for future enhancements, including integration with live payment gateways, GPS-based bike tracking, mobile application development, and improved recommendation algorithms. Overall, RentBike represents a scalable and secure solution that transforms digital infrastructure into practical urban mobility, enabling users to access bike rental services quickly and efficiently while maintaining system stability under increasing demand.

VI.OUTPUT

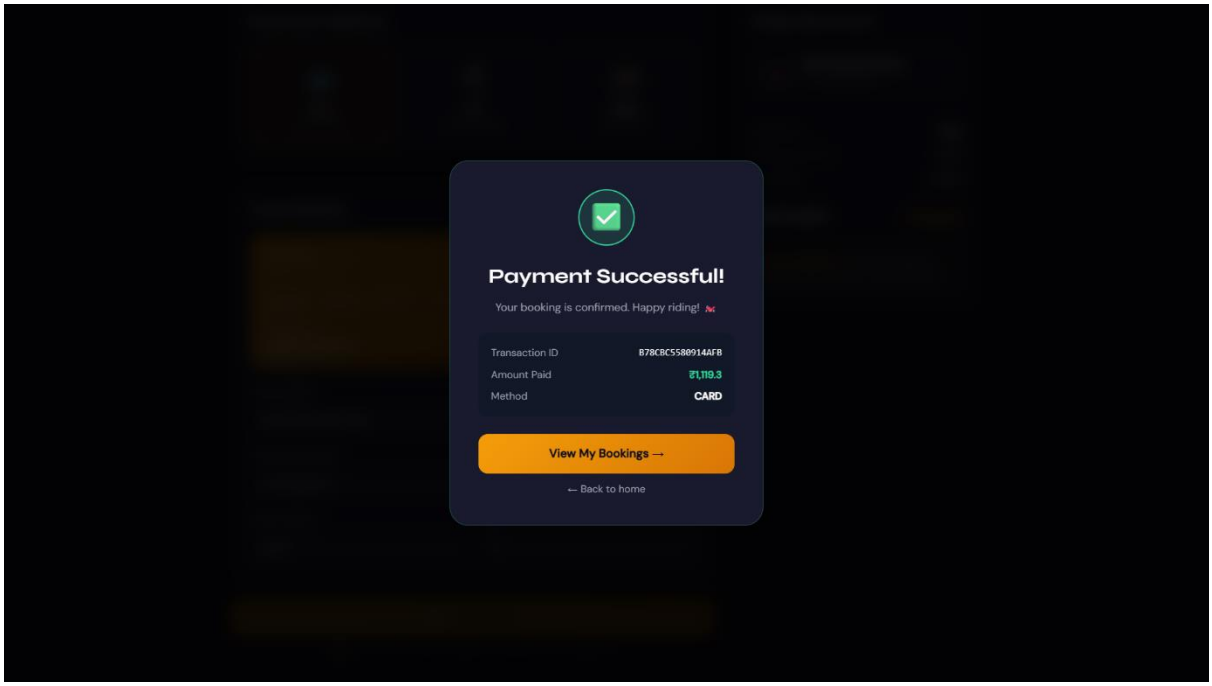


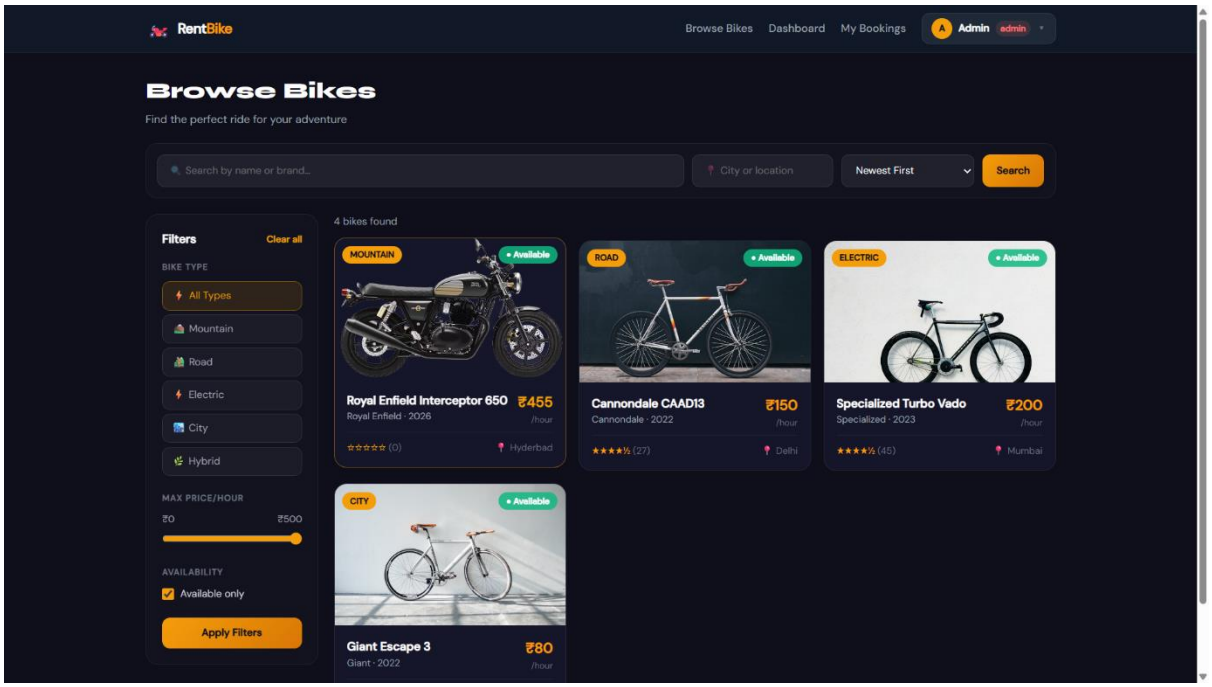
IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.15498



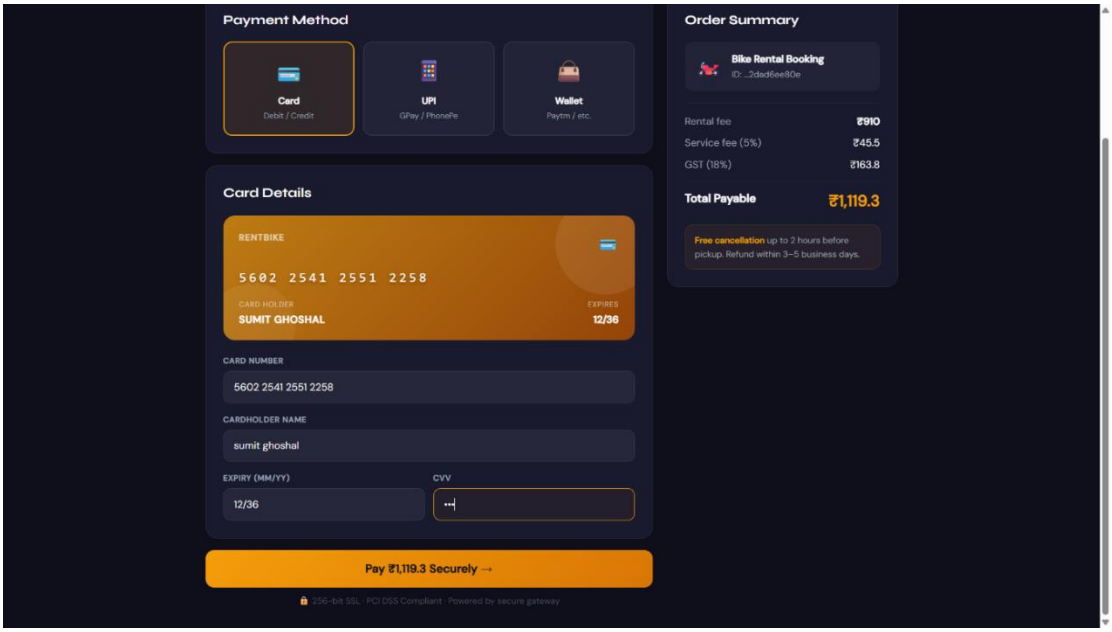


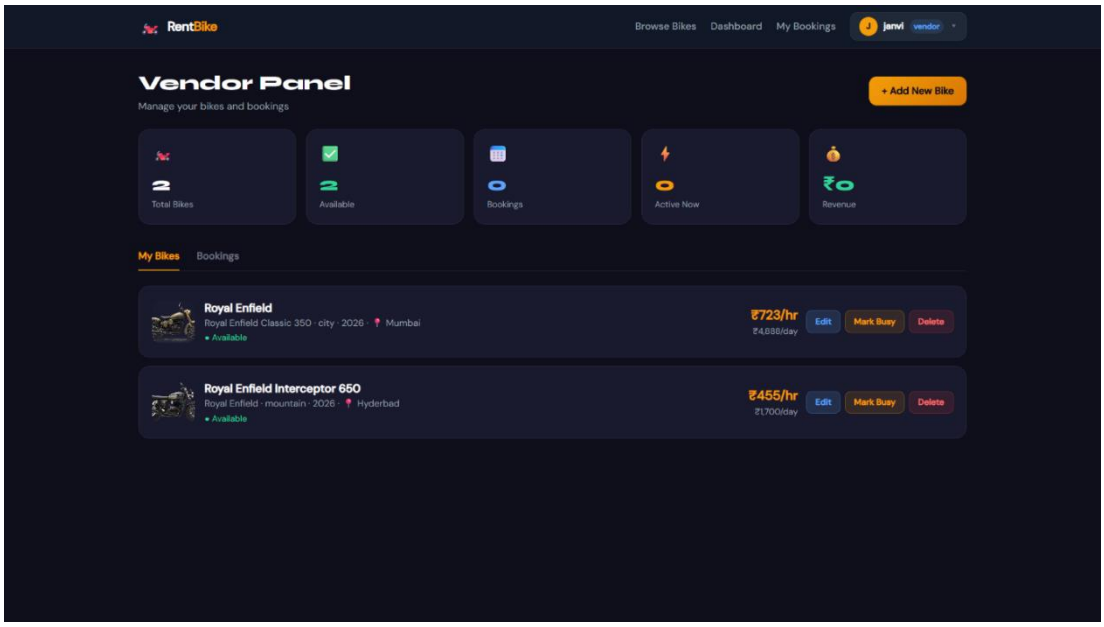
IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.15498





REFERENCES

[1]. R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures (REST)," Doctoral Dissertation, University of California, 2000.
[2]. MongoDB Inc., "MongoDB Atlas Documentation," MongoDB,
[3]. M. Grinberg, "Flask Web Development," O'Reilly Media, 2018.
[4]. Mozilla Developer Network, "JavaScript and Web APIs Documentation," MDN Web Docs
[5]. OWASP Foundation, "Web Application Security Guidelines," OWASP
[6]. JWT.io, "JSON Web Token Introduction and Best Practices,"
[7]. Tailwind Labs, "Tailwind CSS Documentation,"
[8]. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley, 1994.
[9]. R. S. Pressman, "Software Engineering: A Practitioner's Approach," McGraw-Hill, 2014.
[10]. IEEE, "Cloud-Based Web Applications and Microservices Architecture," IEEE Research Papers