

RESTful Web Services Project

Master's in Intelligent Systems and Applications

Development and Deployment of a Bike Rental and Sales Management Application

Academic year :2024-2025

Course : Web Services

Eiffel Bike Corp

Team Members :

Mahmoud Aziz Ammar

Karim Damak

Siwar Najjar

Supervised by :

Mr Hassan Idoudi

Acknowledgment

We would like to express our deepest gratitude to Dr. Hassan Idoudi for his invaluable guidance and expertise, which were instrumental in the successful development of Eiffel Tutoring Solutions. His unwavering support and thoughtful feedback provided us with clarity and inspiration throughout this journey, enabling us to bring our vision to life.

As a team, we take pride in the collaborative effort that made this project possible. Each member brought their unique skills, dedication, and creativity, fostering an environment of innovation and excellence. This project stands as a testament to our shared commitment and determination. Thank you to everyone who contributed to making this endeavor a success.

Contents

List of Figures	5
1 Introduction	6
1.1 Background and Context	6
1.2 Objectives	6
1.3 Scope of Work	6
1.4 Conclusion	7
2 System Design	8
2.1 Introduction	8
2.2 Overview of System Components	8
2.3 RESTful Web Services Design	8
2.4 UML Class Diagram and Data Flow	9
2.5 System Interaction and User Flows	10
2.6 Conclusion	11
3 Technologies and Tools Used	12
3.1 Overview of Technologies	12
3.2 Java RS (Java API for RESTful Web Services)	12
3.3 Web Service Implementation Tools	12
3.3.1 Currency Converter Web Service	12
3.3.2 FastAPI Integration	13
3.3.3 Mail API Tool	13
3.3.4 Other Web Services	13
3.4 Graphical Interface Development	14
3.5 Conclusion	14
4 Implementation and Development	15
4.1 Bike Rental System Implementation	15
4.2 GustaveBikeService Implementation	15
4.3 Web Service Integration (Bank, Currency Conversion, and Email Notifica- tions)	17
4.4 Testing and Debugging	17
4.5 Conclusion	18
5 Challenges and Considerations	19
5.1 System Architecture Design Challenges	19
5.2 Web Service Integration Challenges	19
5.3 Email API and FastAPI Integration Challenges	19
5.4 User Interface Design Considerations	20
5.5 Performance and Scalability Concerns	20
5.6 Conclusion	21

6	Additional Functionalities and Enhancements	22
6.1	Web Application Enhancements	22
6.2	Deep Learning-based Bike Condition Prediction	22
6.3	Payment Integration Using Currency Converter	23
6.4	Conclusion	23
	Conclusion and Perspectives	24

List of Figures

2.1	UML Class Diagram for the Bike Rental System	10
4.1	Bikes display Interface	16
4.2	Login User Interface	16
4.3	Bike Renting Display	16
4.4	Mail Notification	17

Chapter 1

Introduction

1.1 Background and Context

Eiffel Bike Corp provides a bike rental service aimed at promoting sustainable transportation for students and employees of Gustave Eiffel University. This initiative encourages eco-friendly mobility and strengthens the university community through resource sharing. The goal of this project is to design and implement a distributed Java application, utilizing Java RS (Java API for RESTful Web Services), to efficiently manage the bike rental service.

The system is divided into two key components: a localized bike rental service for the university community and an external service, *GustaveBikeService*, which allows external customers to purchase bikes. The project involves developing a comprehensive set of web services and applications to manage the entire lifecycle of bike rentals, from reservation to feedback collection, as well as handling bike sales to external clients.

1.2 Objectives

The main objectives of this project are as follows:

- To create a RESTful web service for managing bike rentals for students and employees within Eiffel Bike Corp.
- To implement essential features such as bike availability checks, waiting list management, and feedback collection on bike conditions upon return.
- To develop *GustaveBikeService*, a web service that facilitates the sale of rented bikes to external customers, incorporating real-time currency conversion and secure payment processing via an external banking web service.
- To build graphical interfaces for students, employees, and external customers, ensuring seamless interaction with the system.

1.3 Scope of Work

The project is divided into two phases:

1. Phase 1: Bike Rental Service Management

This phase focuses on implementing the bike rental functionality for university members. Key features include:

- User registration and authentication.
- Bike rental requests and waiting list management.
- Feedback and condition reporting for bikes upon their return.

2. **Phase 2: Bike Sales via GustaveBikeService**

In this phase, the project expands to offer rented bikes for sale to external customers. This includes:

- Real-time currency conversion for displaying bike prices in various currencies.
- Integration with a banking web service to process payments securely.
- Creation and validation of customer purchases.

1.4 **Conclusion**

Through this project, we aim to deliver a robust and user-friendly system that meets the needs of Eiffel Bike Corp while supporting sustainable and collaborative mobility solutions for both the university community and external customers.

Chapter 2

System Design

2.1 Introduction

This chapter provides a detailed overview of the system design for the Eiffel Bike Corp. project, focusing on how the bike rental service is structured and how it integrates into a distributed Java-based system. The design incorporates the use of RESTful Web Services to handle bike rental requests, manage availability, and facilitate secure payment processing. The system also includes features for managing waiting lists, collecting feedback on bike conditions, and enabling the sale of bikes through the `GustaveBikeService`. This section outlines the design philosophy, system components, and the architecture of the application.

2.2 Overview of System Components

The system is composed of several key components, each designed to handle different aspects of the bike rental and sales process:

- **Bike Rental Service:** A core service that allows students and employees to rent bikes. This service handles user authentication, bike availability, rental requests, and feedback management.
- **GustaveBikeService:** A web service that allows external customers to view, purchase, and pay for bikes that have been rented at least once by a university member. This service integrates with a real-time currency conversion API to display prices in various currencies and includes payment processing through a Bank web service.
- **Bank Web Service:** This external service is used by the `GustaveBikeService` to verify funds and process payments securely.
- **Graphical User Interfaces (GUIs):** Custom-designed interfaces for university students, employees, and external customers to interact with the system. The interfaces are optimized for ease of use and provide a seamless experience for bike rental and purchase.

2.3 RESTful Web Services Design

The system relies on RESTful Web Services for communication between its components. Each service is designed to be stateless and accessible via HTTP requests. The main services include:

- **Bike Rental Service API:** Provides endpoints for bike availability checks, rental requests, and feedback submissions. It uses HTTP GET for availability checks, POST for renting bikes, and PUT for updating bike conditions.
- **GustaveBikeService API:** Allows external customers to browse available bikes, add them to a shopping cart, and initiate payment. This service integrates with the Currency API and the Bank API for payment processing.
- **Currency API:** Handles currency exchange rates to support multi-currency transactions. It provides HTTP GET endpoints for retrieving exchange rates.
- **Bank API:** Facilitates secure payment processing. It uses HTTP POST for initiating payments and HTTP GET for retrieving the status of transactions.

The design of these services ensures that each functionality is encapsulated within its own API, allowing for modularity and scalability.

2.4 UML Class Diagram and Data Flow

The UML class diagram represents the main objects and their relationships within the system. Key classes include:

- **Bike:** Represents a bike in the system, including attributes like bike ID, condition, and availability status.
- **User:** Represents a user of the system (either student, employee, or external customer), including login details, rental history, and feedback on bikes.
- **RentalRequest:** Represents a request to rent a bike, including the requested bike, user, and rental period.
- **Payment:** Represents the payment process, including transaction details, amount, and payment status.

The data flow shows how data moves between the client, server, and external services, ensuring that each component interacts correctly and efficiently. For example, when a user requests a bike, the system first checks availability through the Bike Rental Service API, then proceeds to the Waiting List API if the bike is unavailable, and notifies the user once the bike becomes available.

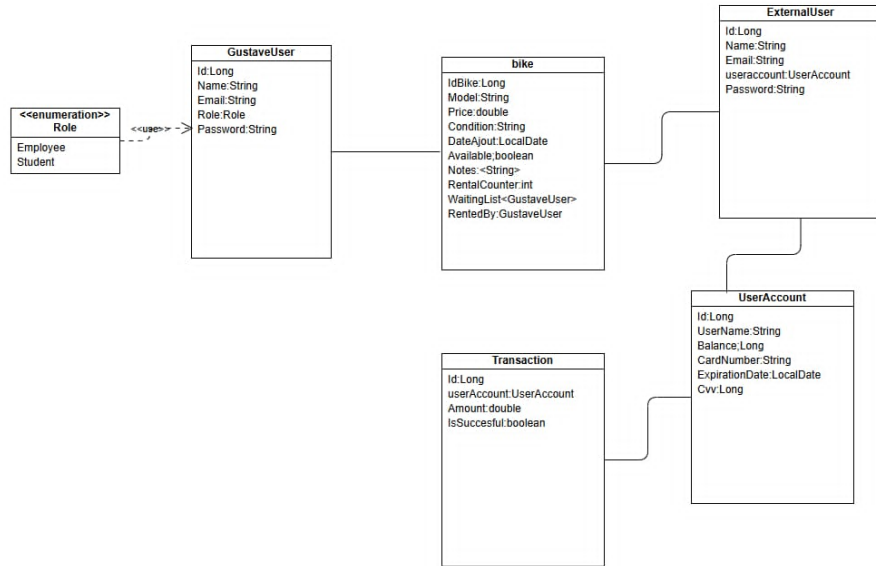


Figure 2.1: UML Class Diagram for the Bike Rental System

2.5 System Interaction and User Flows

The system interaction and user flows are designed to ensure a smooth and intuitive experience for all users. Below is a description of the typical user flow for both bike rentals and purchases:

1. Bike Rental Process:

- A user logs in to the bike rental system.
- The user searches for available bikes and selects one.
- If the bike is available, the user can proceed with the rental.
- If the bike is not available, the user is placed on a waiting list.
- Once the bike becomes available, the user is notified and can complete the rental process.

2. Bike Purchase Process via GustaveBikeService:

- A customer accesses the GustaveBikeService to browse bikes available for purchase.
- The customer is presented with the option to view prices in different currencies.
- After selecting a currency, the system calculates the total price using real-time exchange rates.
- The customer proceeds with the payment via the Bank Web Service.
- Once payment is confirmed, the transaction is completed, and the bike is sold.

These user flows ensure that the system is intuitive and that users can complete their transactions with minimal effort.

2.6 Conclusion

This chapter outlined the system design for the Eiffel Bike Corp. project, detailing the architecture, components, and interactions within the system. The use of RESTful web services enables modularity and scalability, while the integration of external services, such as the Bank Web Service and real-time currency conversion, enhances the functionality of the bike rental and sales process. The next chapter will focus on the implementation details, including the technologies used and how the system components were developed and integrated.

Chapter 3

Technologies and Tools Used

3.1 Overview of Technologies

In this chapter, we outline the technologies and tools employed to develop the Bike Rental Service and the accompanying web services. The system leverages modern web technologies to ensure scalability, performance, and ease of use. The following sections detail the primary technologies used for building the RESTful web services, graphical interfaces, and other components of the system.

3.2 Java RS (Java API for RESTful Web Services)

The core of our system is built using the Java API for RESTful Web Services (Java RS). Java RS is a set of APIs provided by Java to create RESTful web services, which are lightweight and scalable. We used Java RS to implement the backend services for the Bike Rental Service, including endpoints for managing bike availability, user rentals, and interactions with external services like the currency converter.

Key features and advantages of Java RS used in the project:

- Provides a simple, scalable architecture for creating web services.
- Supports a wide range of HTTP methods (GET, POST, PUT, DELETE), making it ideal for implementing the Bike Rental and GustaveBikeService APIs.
- Ensures high performance and low overhead, essential for handling numerous simultaneous requests.
- Integrated with other Java technologies (JAX-RS) to enhance functionality and ease of development.

3.3 Web Service Implementation Tools

To implement the various functionalities required by the project, we utilized a combination of tools and libraries to develop the web services. Below are the primary tools and approaches used:

3.3.1 Currency Converter Web Service

To enable users to purchase bikes from the GustaveBikeService in various currencies, we integrated a Currency Converter Web Service. This web service provides real-time

exchange rates for different currencies, allowing the system to convert the bike price into the user's preferred currency.

- **Currency Exchange API:** We used the external API "openexchangerates" to fetch real-time exchange rates and ensure the accuracy of currency conversions.
- **JSON and XML:** The Currency Converter Web Service communicates with other systems using JSON and XML formats, ensuring seamless integration and efficient data exchange.
- **Security:** All data transmissions within the currency conversion web service are encrypted and secured via HTTPS.

3.3.2 FastAPI Integration

The implementation of web services was carried out using FastAPI, a modern and efficient Python web framework. Its asynchronous nature and robust support for dependency injection allowed us to build scalable and high-performance APIs.

- **API Design:** FastAPI was used to design and implement the core API endpoints, ensuring low-latency interactions for bike rentals, currency conversion, and payment processing.
- **Model Integration:** A Transformer-based model (inspired by Facebook's research) was integrated into the system to handle intelligent processing tasks, further enhancing the services' capabilities.
- **Data Serialization:** For seamless data transformation between objects and JSON, we utilized the Jackson Annotation package in combination with FastAPI.

3.3.3 Mail API Tool

To enhance communication between the system and its users, we implemented a mailing service using the Javax Mail API. This service allows the system to send automated emails for various purposes, such as booking confirmations, payment receipts, and notifications.

- **Javax Mail API:** The library was used to configure and send emails securely over SMTP servers.
- **Integration with FastAPI:** The mail service was integrated into the FastAPI framework, enabling asynchronous email dispatch triggered by key user actions.
- **Security Features:** Emails are sent via secured SMTP connections, ensuring the confidentiality and integrity of communication.

3.3.4 Other Web Services

In addition to the currency converter, FastAPI integration, and mailing service, several other web services were implemented to manage the core functionalities of the Bike Rental Service. These include services for checking bike availability, managing waiting lists, handling user data, and interacting with the external bank service for payment processing.

3.4 Graphical Interface Development

A key aspect of the system is the graphical interface developed for users (students, employees, and customers) to interact with the Bike Rental Service. The interface is designed to be intuitive and user-friendly, allowing seamless navigation for renting bikes, viewing available bikes, and making payments.

The following tools and technologies were used for graphical interface development:

- **HTML5/CSS3 (for web version):** For the web-based client, HTML5 and CSS3 were employed to create a responsive design compatible with various devices (mobile, tablet, desktop).
- **JavaScript (with AJAX):** JavaScript was used to enable dynamic updates, with AJAX calls facilitating real-time fetching of bike availability and payment handling.

3.5 Conclusion

This chapter outlined the primary technologies and tools used in developing the Bike Rental Service system and its associated web services. By leveraging FastAPI, Transformer-based models, Jackson Annotation, Javax Mail API, external APIs, and modern web development technologies, we implemented a robust, secure, and scalable solution. The next chapter will detail the implementation process, including how these technologies were integrated into the final system.

Chapter 4

Implementation and Development

4.1 Bike Rental System Implementation

The Bike Rental System was implemented using RESTful web services, FastAPI, and a user-friendly interface. Without a traditional database, the system uses file-based data management to store and retrieve information. The key components are:

- **Bike Availability:** The system checks the availability of bikes by reading from JSON files that store the current inventory. When a user requests a bike, the system updates the relevant file to reflect the new state.
- **Bike Rental Logic:** If no bikes are available, the system places users on a waiting list stored in a file. Users are notified via email (using the Javax Mail API) when the requested bike becomes available.
- **Backend:** The backend was developed using Java RS, implementing RESTful APIs for efficient interaction with the file-based storage system.
- **Data Flow:** The system integrates external services (e.g., Currency Converter API) to ensure accurate pricing and diverse payment options. File-based logs are used to track service interactions and user transactions.

4.2 GustaveBikeService Implementation


The GustaveBikeService extends the bike rental system to external users, enabling them to browse available bikes, add items to a shopping cart, and make purchases.

- **API Design:** GustaveBikeService adheres to RESTful principles, providing endpoints for operations such as viewing bikes, checking availability, and making purchases. Data is stored and retrieved from JSON files.
- **Integration with External Services:** The service interacts with external APIs, such as the Bank API for secure payments and the Currency Conversion API for international pricing options.
- **Notification System:** An email notification system, implemented using Javax Mail API, ensures users receive updates on waiting list status.
- **Graphical Interface:** The web-based graphical interface, developed using HTML5, CSS3, and JavaScript (with AJAX), delivers a seamless and responsive shopping experience.

Bike Availability					
ID	Model	Condition	Notes	Status	Actions
1	Model A	Needs minor repairs	<ul style="list-style-type: none"> - First bike - Needs repair - Popular model 	Free ●	<button>Rent</button>
2	Model B	Good condition	<ul style="list-style-type: none"> - Second bike - Recently serviced - Good condition 	Free ●	<button>Rent</button>

Figure 4.1: Bikes display Interface

Welcome Back to Eiffel Bike Corp



Login to Continue

Login

Email

Password

Login

Don't have an account? No problem. [Create one.](#)

Figure 4.2: Login User Interface

Bike Availability					
ID	Model	Condition	Notes	Status	Actions
1	Model A	Needs minor repairs	<ul style="list-style-type: none"> - First bike - Needs repair - Popular model 	Unavailable ●	<button>Rent</button>
2	Model B	Good condition	<ul style="list-style-type: none"> - Second bike - Recently serviced - Good condition 	Free ●	Return your current bike first
3	Model C	Good condition	- ena eli behc yrenlini	Free ●	Return your current bike first
4	Model D	Good condition	- ena num 4	Free ●	Return your current bike first

Figure 4.3: Bike Renting Display



Figure 4.4: Mail Notification

4.3 Web Service Integration (Bank, Currency Conversion, and Email Notifications)

The integration of external services was a key aspect of the system's functionality. File-based data management was leveraged to ensure smooth operation. Key integrations include:

- **Bank API Integration:** The Bank API was integrated to validate payment details and process transactions securely. Communication was managed through JSON data exchanges, with transaction logs stored in files.
- **Currency Conversion API:** The external Currency Conversion API (OpenExchangeRates) was used to fetch up-to-date exchange rates. Converted prices were stored in files for reuse during the session.
- **Email Notifications:** Using Javax Mail API, the system sends automated notifications for waiting list updates.

4.4 Testing and Debugging

The system underwent extensive testing to ensure reliable performance and seamless functionality. Key aspects included:

- **Unit Testing:** Individual modules and functions were rigorously tested, particularly the file-based storage and retrieval logic.
- **Integration Testing:** Interactions between the system's components and external APIs were tested to confirm compatibility and smooth data flow.
- **Error Handling:** Comprehensive debugging was conducted to manage edge cases, such as incomplete file writes or missing files, ensuring robustness and reliability.

4.5 Conclusion

The implementation of the Bike Rental System and GustaveBikeService successfully delivered a scalable and user-friendly platform. By utilizing FastAPI and a file-based storage approach, the system efficiently managed bike availability, rentals, and external API integrations. The inclusion of Currency Conversion, Bank API, and email notifications ensured accurate pricing, secure transactions, and timely updates. Extensive testing confirmed the system's reliability and robustness, making it well-suited for real-world use and user needs.

The addition of user interfaces, such as the GustaveBikeService web interface and notification system, further enhances the service's accessibility and user experience. The seamless integration of external services also positions the system for future scalability and global reach. The graphics added in this chapter provide a clear visual representation of the system's components, data flow, and user interface design, contributing to a comprehensive understanding of the overall implementation.

Chapter 5

Challenges and Considerations

5.1 System Architecture Design Challenges

Designing the system architecture for the bike rental service posed several challenges:

- **Scalability:** Ensuring that the system can handle a large number of simultaneous users required careful design of the backend services and system interactions.
- **Fault Tolerance:** Implementing a fault-tolerant system to ensure that the service remains operational even in the event of network failures or service disruptions.
- **Data Consistency:** Maintaining consistency across different components, such as ensuring that bike availability is accurately reflected in real-time.

5.2 Web Service Integration Challenges

Integrating third-party services, including the Currency Conversion API and the Mail API, presented several difficulties:

- **Data Synchronization:** Ensuring that the exchange rates from the Currency Conversion API were synchronized correctly with the bike pricing system to provide accurate pricing.
- **Security Concerns:** Implementing secure communication protocols (such as HTTPS) to protect sensitive user data and financial transactions.
- **Email Delivery Reliability:** Ensuring the delivery of emails (booking confirmations) despite possible delays or failures in email services.

5.3 Email API and FastAPI Integration Challenges

Email API Challenges

Integrating the email service using the Javax Mail API introduced several challenges:

- **Configuration Complexity:** Setting up the Javax Mail API required careful configuration of SMTP servers, including handling authentication and encryption protocols.
- **Email Delivery Failures:** Ensuring reliable email delivery faced obstacles such as spam filters, server rate limits, and handling invalid email addresses.

- **Security Concerns:** Protecting sensitive user information in email content and ensuring secure communication through encrypted SMTP connections.
- **Asynchronous Operations:** Email dispatch needed to be asynchronous to avoid blocking critical application processes, requiring additional integration efforts with FastAPI's asynchronous capabilities.

FastAPI Challenges

Building the core functionality with FastAPI also presented some challenges:

- **Dependency Management:** Managing dependencies in FastAPI for complex workflows like email dispatch and external API calls while maintaining code simplicity.
- **Asynchronous Programming:** Ensuring proper handling of asynchronous requests and responses to prevent performance bottlenecks or deadlocks.
- **Error Handling:** Implementing robust error handling mechanisms to gracefully manage failures from external services, including the email API and currency conversion API.
- **Scalability:** Ensuring the FastAPI application could handle a high volume of concurrent requests efficiently without degrading performance, especially during peak periods.
- **Integration with External Libraries:** Integrating FastAPI with various external libraries (such as Javax Mail for email dispatch and the Currency Converter API) required careful management to ensure seamless interaction between components.

5.4 User Interface Design Considerations

The graphical user interface (GUI) for both the bike rental system and the GustaveBike-Service was designed with the user experience in mind. Considerations included:

- **Simplicity:** The interface was kept simple and intuitive to allow users to easily browse, rent bikes, and make purchases.
- **Responsiveness:** The design was made responsive to ensure compatibility across various devices, including desktops and mobile phones.
- **User Feedback:** The system provides real-time feedback to users regarding the status of their bike rentals and purchases.

5.5 Performance and Scalability Concerns

To ensure the system can handle high traffic and large volumes of data, several performance and scalability considerations were made:

- **Load Balancing:** The backend was designed to support load balancing to distribute traffic evenly across servers and improve performance during peak usage.
- **Caching Mechanisms:** Caching was implemented to reduce load times and enhance system responsiveness, particularly for frequently accessed data like bike availability.

5.6 Conclusion

The challenges encountered during the design and implementation of the bike rental system highlighted the complexity of building a scalable, reliable, and secure service. Addressing issues such as scalability, fault tolerance, and data consistency required careful planning and implementation. The integration of third-party services, including the Currency Conversion API and Mail API, presented challenges in data synchronization, security, and email delivery reliability. Additionally, the integration of FastAPI with asynchronous operations demanded careful dependency management and error handling. Despite these challenges, the system was designed to provide a smooth user experience with a responsive interface and robust performance during high traffic periods, ensuring the success of the overall architecture.

Chapter 6

Additional Functionalities and Enhancements

6.1 Web Application Enhancements

While the bike rental service is currently accessible through a web-based interface, future enhancements could further improve user experience and service delivery:

- **Bike Availability Tracking:** Users could access real-time updates on bike availability through the website, allowing them to plan rentals more efficiently.
- **Notification System:** The system could be enhanced to send email or web-based notifications when a reserved bike becomes available or when there are updates regarding the rental process.

6.2 Deep Learning-based Bike Condition Prediction

Instead of relying on a traditional user rating system, the system leverages a sophisticated deep learning model to predict the condition of bikes based on user-provided notes. This approach ensures more accurate and consistent assessments, which are crucial for effective bike maintenance and service reliability. The deep learning model, based on Facebook's transformer-based architecture, plays a central role in processing and interpreting textual data to generate these predictions. Key features include:

- **Condition Prediction Model:** A deep learning model fine-tuned for bike rentals that predicts bike conditions (e.g., excellent, good, fair, or poor) from user notes.
- **Transformer-based Architecture:** Facebook's transformer model captures contextual relationships between words in user feedback, allowing it to understand complex phrases and assess bike conditions accurately.
- **Fine-tuning for Specific Use Cases:** The model is fine-tuned on a dataset tailored to bike rental scenarios, helping it specialize in bike-related feedback, such as mechanical issues, comfort concerns, or wear and tear.
- **User Notes Analysis:** The model processes user notes about bike condition, such as "The front wheel was squeaky and hard to steer," extracting relevant details that signal mechanical issues. This enables deeper and more accurate condition assessments.

- **Continuous Learning:** The model improves over time by learning from new user feedback, refining its predictions to keep pace with changing user behavior and bike conditions.

This approach ensures accurate, scalable, and efficient analysis of user-provided data, enhancing the user experience and streamlining maintenance operations, ensuring bikes remain in optimal condition.

6.3 Payment Integration Using Currency Converter

To make the bike rental service more accessible to a global audience, we introduce a payment currency converter feature. This system allows users to make payments in their preferred currency by automatically converting the amount to the local currency at real-time exchange rates. The key features include:

- **Multi-currency Support:** The payment system supports multiple international currencies, ensuring that users from different regions can pay in their local currency, making the service more accessible to a wider audience.
- **Real-time Currency Conversion:** The system uses real-time exchange rates to convert the rental fee into the local currency, providing transparency and accurate pricing for international users.
- **User-friendly Interface:** The currency converter is integrated seamlessly into the checkout process, providing an intuitive interface for users to view and confirm the converted price before finalizing the payment.

This currency converter integration enhances the user experience by simplifying international payments and ensuring that the bike rental service can easily accommodate users from diverse financial backgrounds.

6.4 Conclusion

In conclusion, the additional functionalities and future enhancements outlined in this chapter aim to provide users with a more seamless and efficient bike rental experience. The integration of deep learning-based bike condition prediction, along with expanded payment options through the currency converter, will ensure the service remains competitive and adaptable to evolving user needs. By continuously improving user engagement and system performance, the bike rental service can expand its reach and enhance its value proposition, making it more accessible and convenient for a global audience.

Conclusion and Future Perspectives

Conclusion

This dissertation presents the development of a Bike Rental System that integrates key technologies to enhance user experience and system efficiency. The main contributions include the implementation of a scalable RESTful system for bike rentals, integration of external services for accurate payment processing, and the design of a user-friendly interface. These advancements ensure reliable bike rentals, seamless payment handling, and a responsive user experience. Future improvements, such as mobile apps and IoT integration, offer potential for further enhancing the system's capabilities.

Perspectives

To extend the current work and further enhance the Bike Rental System, the following perspectives are suggested:

1. **Mobile Application Development:** A dedicated mobile app could be developed to extend the system's reach, providing users with a more convenient and efficient way to rent bikes, track availability, and make payments.
2. **Advanced Features for User Feedback:** Integrating an advanced feedback and review system could provide valuable insights for users and help in improving the service. Additionally, a deeper integration with sentiment analysis could help identify and address user concerns in real-time.
3. **Integration with IoT Devices:** Future work could involve integrating Internet of Things (IoT) devices to provide real-time tracking of bikes and enhance the system's capabilities with advanced features like bike location tracking and automated rentals. This could also include bike diagnostics, giving users and operators more precise information about the bike's condition.
4. **Machine Learning for Dynamic Pricing and Demand Forecasting:** Implementing machine learning algorithms for dynamic pricing and demand forecasting could optimize rental prices based on demand, time, and location. This feature would help improve revenue generation and customer satisfaction.
5. **Sustainability Features:** Future versions of the system could integrate sustainability metrics, allowing users to track the environmental impact of their bike usage. This could align with growing eco-conscious trends and encourage more sustainable transportation choices.