Development Phase/Reflection Phase

Introduction

This development phase, or reflection phase, is about a website providing a unique feature to its users. In this development phase of RentOBike project, from conceptual groundwork to the actual implementation. It is guided by insights and feedback received during the conception phase. As mentioned in the conception phase, 'RentOBike' is an online platform where two types of users engage: one who wants to post their bicycle for rent and the second who wants to rent a bicycle. To create this online platform, we use the Flask framework, a simple but powerful tool in Python, to build the backend server. This setup creates a strong system that perfectly fits our website's needs.

Framework and Development Tools

Flask :- As a framework, we choose Flask because of its flexibility and efficiency that helps to develop lightweight web applications.

PyCharm :- As a development environment, we choose PyCharm that provides tools for debugging, testing, and managing our project.

Third-party Libraries

SQLALchemy :- It is for the database management. This library provides a powerful ORM layer for easier interaction with our database. It also helps with the user management, bicycle listings, and bookings.

Werkzeug :- It is for the security function. It includes the password hashing that ensures the safe handling of user credentials.

Flask-Login :- It is for managing the user sessions. It simplifies the process of user authentication and authorization.

Rationale Behind Tool Selection (Researched Tool Selection for Task Implementation)

- Compatibility with Project Architecture
- Community Support and Documentation
- Performance Benchmarks
- Security Features
- Ease of Use and Learning Curve
- Scalability Potential

Comparison and Evaluation Process

- Flask is compared against Django and Express.js because of its minimalistic approach made it ideal for RentOBike website.

- SQLAlchemy is preferred over SQL and other ORMs because of its documentation, ease of use and integration with Flask.
- Werkzeug and Flask-Login is chosen for the security features and seamless integration. It is essential for maintaining a secure and user-friendly platform.

Integration of Feedback

The feedback that we get from the conception phase highlighted the need for scalability and security. It guides us to select the tools that is known for their performance efficiency and security capabilities.

Refinement of Requirements

Following the conception phase, we see the functional requirements as follows:

- User Registration
- Bicycle Listing
- Booking
- User Reviews and Ratings

After getting the feedback related to the non-functional requirements to establish measurable targets that underscore the platform's reliability, security, and user experience are as follows:

Performance :- The target is to load times are set at under 2 seconds for a page load across the website, it ensures the efficiency. It may enhance the user satisfaction and engagement.

Security :- The website emphasizes the security measures. It includes the implementation of data encryption in transit and secure password storage practices. Passwords are hashed using secure algorithms, such as bcrypt, meeting modern security standards to protect user information against unauthorized access.

Scalability :- The website is designed to accommodate up to 5000 concurrent users and 300 to 500 bicycle listings. This type of capability ensures the platform remains accessible and responsive during peak usages of the website.

Implementation of Modules/Components

The development of the website is focused on implementing key modules :

User Module :- This module manages the user interactions with the platform. It includes registration, login, and profile management. As in this website, we are utilizing the Flask-Login, that manage the user sessions, enhancing the security and personalization of the user experience.

Bicycle Module :- This module enables bicycle owners to list their bicycle on the platform. It captures the information related to the bicycle, like name, description, and an availability of the bicycle. It shows the SQLAlchemy's ORM efficiency.

Software and System Documentation Development

There are two types of approach to documentation:

Inline Documentation :- It utilizes Python docstrings for inline documentation. Moreover, it facilitates future development and maintenance.

Sphinx :- we use it for generating the accessible HTML documentation. Sphinx transforms our inline documentation into a structured format.

Documentation of Important Design Decisions

For this website here are the following key design decisions:

Flask :- It is for the simplicity and for the speed in development of the website and it is also suitable for the project scale.

SQLALchemy :- We use this as an ORM for ease of database operations and migrations.

Werkzeug :- It is for the secure password handling, and it prioritizes user security.

Source Code Commenting

At the time of developing the website, we see the comment on our source code. It includes the explanation of the complex logic, documentation of functions and their parameters and the notes that will tell us that why certain approaches were taken. It will help the future developers to understand the system.

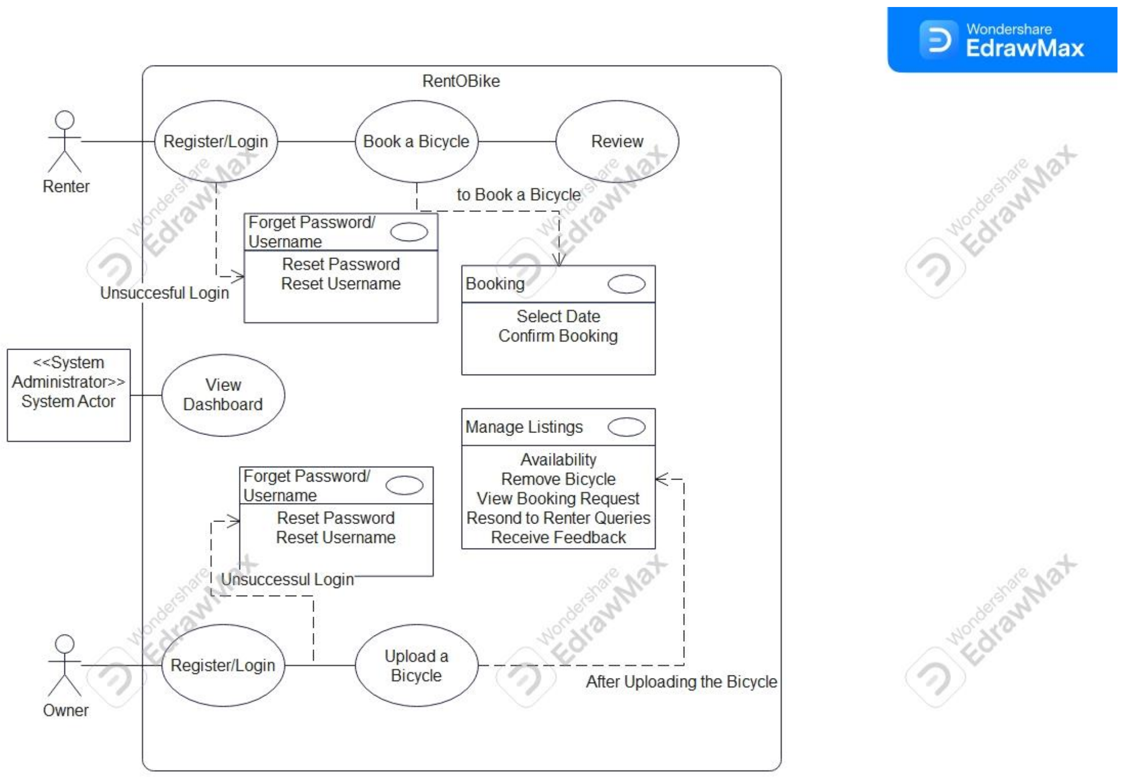
Preparation of Test Data and Software Testing

We have prepared test data that reflects realistic use cases, such as user accounts and bicycle listings. Initial testing has focused on unit tests for individual functions and integration tests to ensure modules work together as expected. Moreover, testing frameworks like PyTest have been utilized for backend testing, it ensures that the website behaves an intended under various scenarios.

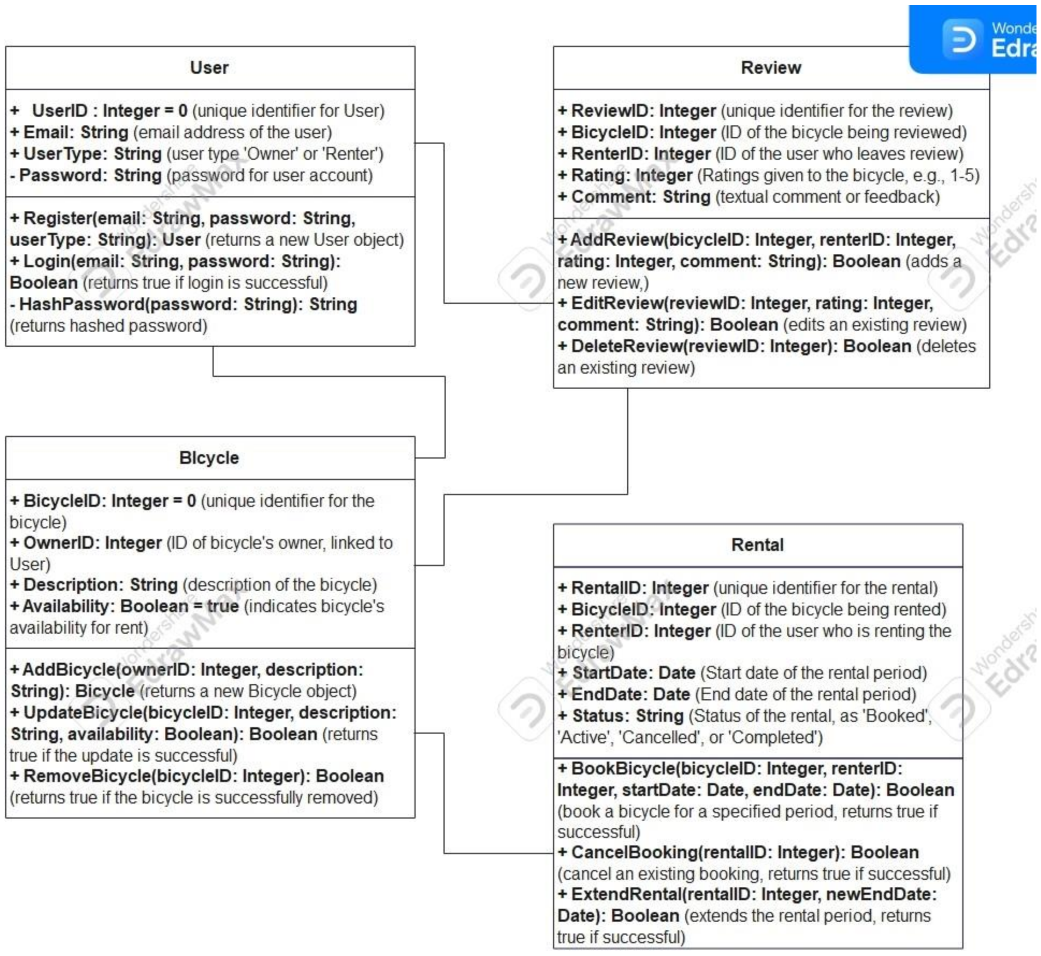
UML Diagrams

UML Diagrams provides a visual representation of the system architecture and data flow. It enhances comprehension of the platform’s design. We will see the UML Diagrams below:

UML Use-case Diagram



UML Class Diagram



User

+ UserID : Integer = 0 (unique identifier for User)
+ Email: String (email address of the user)
+ UserType: String (user type 'Owner' or 'Renter')
- Password: String (password for user account)

+ Register(email: String, password: String, userType: String): User (returns a new User object)
+ Login(email: String, password: String): Boolean (returns true if login is successful)
- HashPassword(password: String): String (returns hashed password)

Review

+ ReviewID: Integer (unique identifier for the review)
+ BicycleID: Integer (ID of the bicycle being reviewed)
+ RenterID: Integer (ID of the user who leaves review)
+ Rating: Integer (Ratings given to the bicycle, e.g., 1-5)
+ Comment: String (textual comment or feedback)

+ AddReview(bicycleID: Integer, renterID: Integer, rating: Integer, comment: String): Boolean (adds a new review,)
+ EditReview(reviewID: Integer, rating: Integer, comment: String): Boolean (edits an existing review)
+ DeleteReview(reviewID: Integer): Boolean (deletes an existing review)

BIcycle

+ BicycleID: Integer = 0 (unique identifier for the bicycle)
+ OwnerID: Integer (ID of bicycle's owner, linked to User)
+ Description: String (description of the bicycle)
+ Availability: Boolean = true (indicates bicycle's availability for rent)

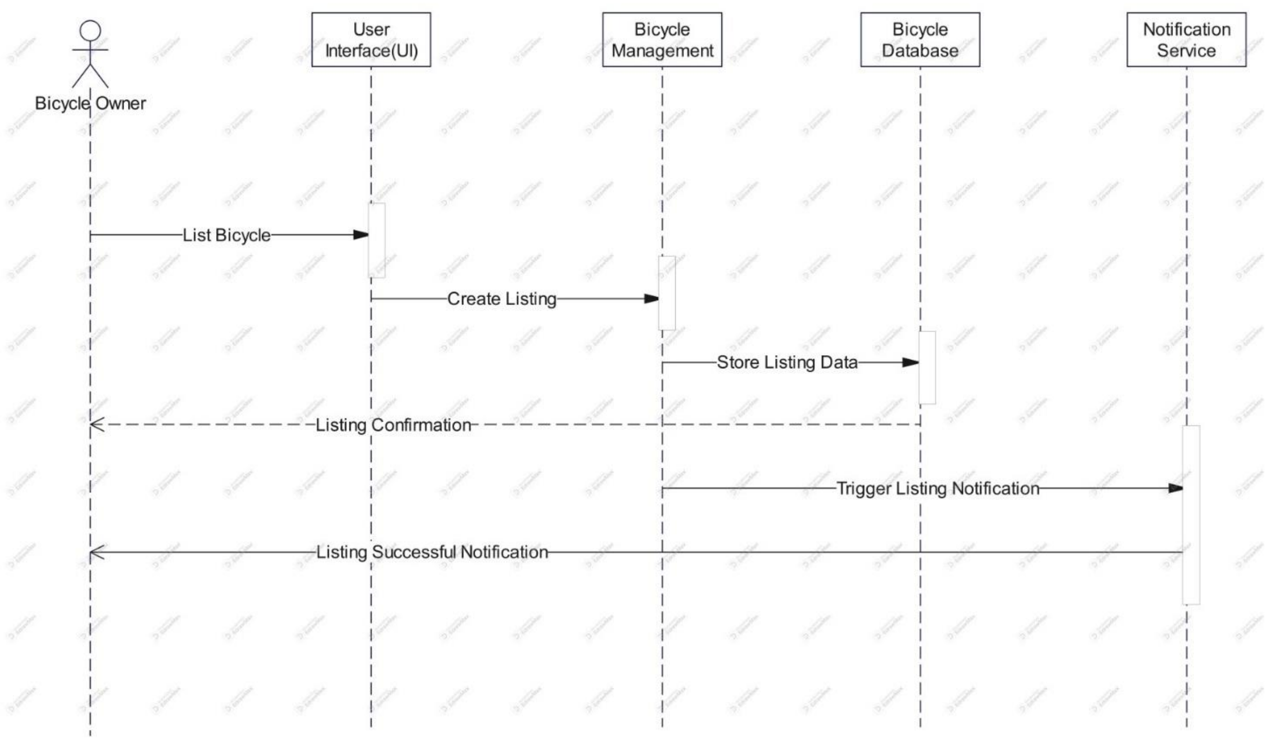
+ AddBicycle(ownerID: Integer, description: String): Bicycle (returns a new Bicycle object)
+ UpdateBicycle(bicycleID: Integer, description: String, availability: Boolean): Boolean (returns true if the update is successful)
+ RemoveBicycle(bicycleID: Integer): Boolean (returns true if the bicycle is successfully removed)

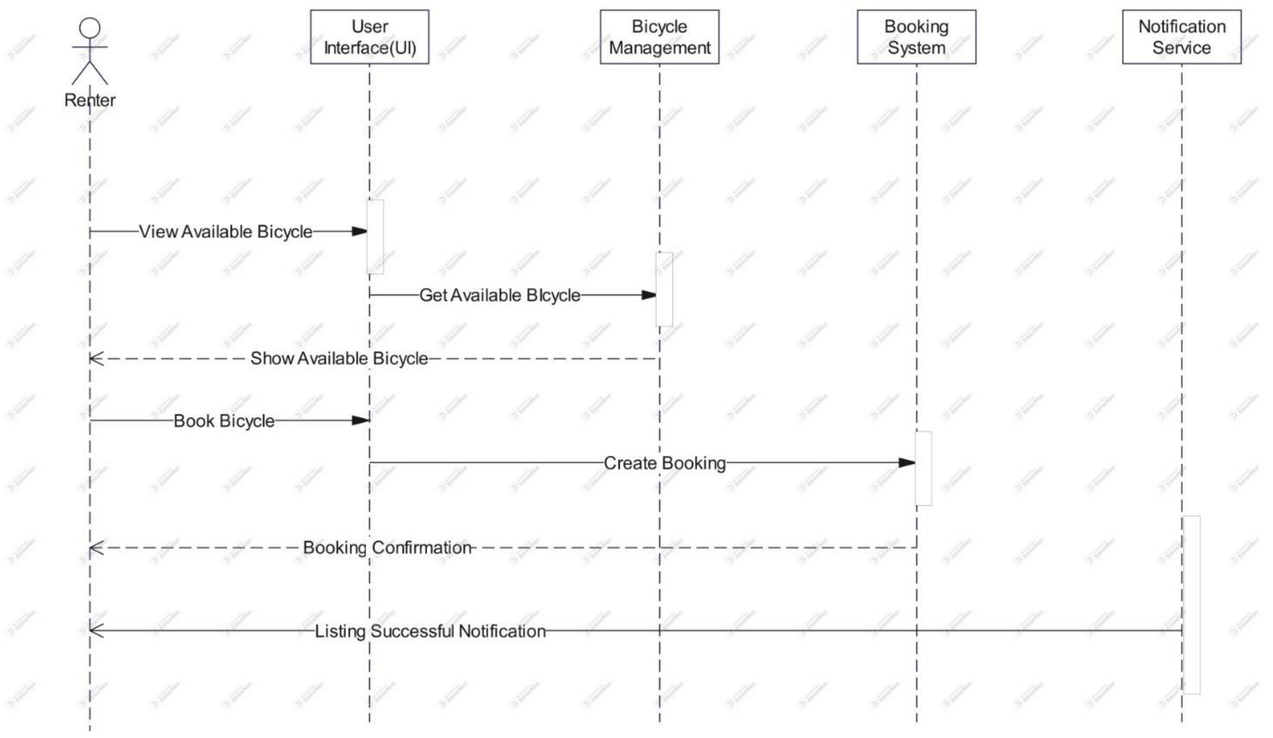
Rental

+ RentalID: Integer (unique identifier for the rental)
+ BicycleID: Integer (ID of the bicycle being rented)
+ RenterID: Integer (ID of the user who is renting the bicycle)
+ StartDate: Date (Start date of the rental period)
+ EndDate: Date (End date of the rental period)
+ Status: String (Status of the rental, as 'Booked', 'Active', 'Cancelled', or 'Completed')

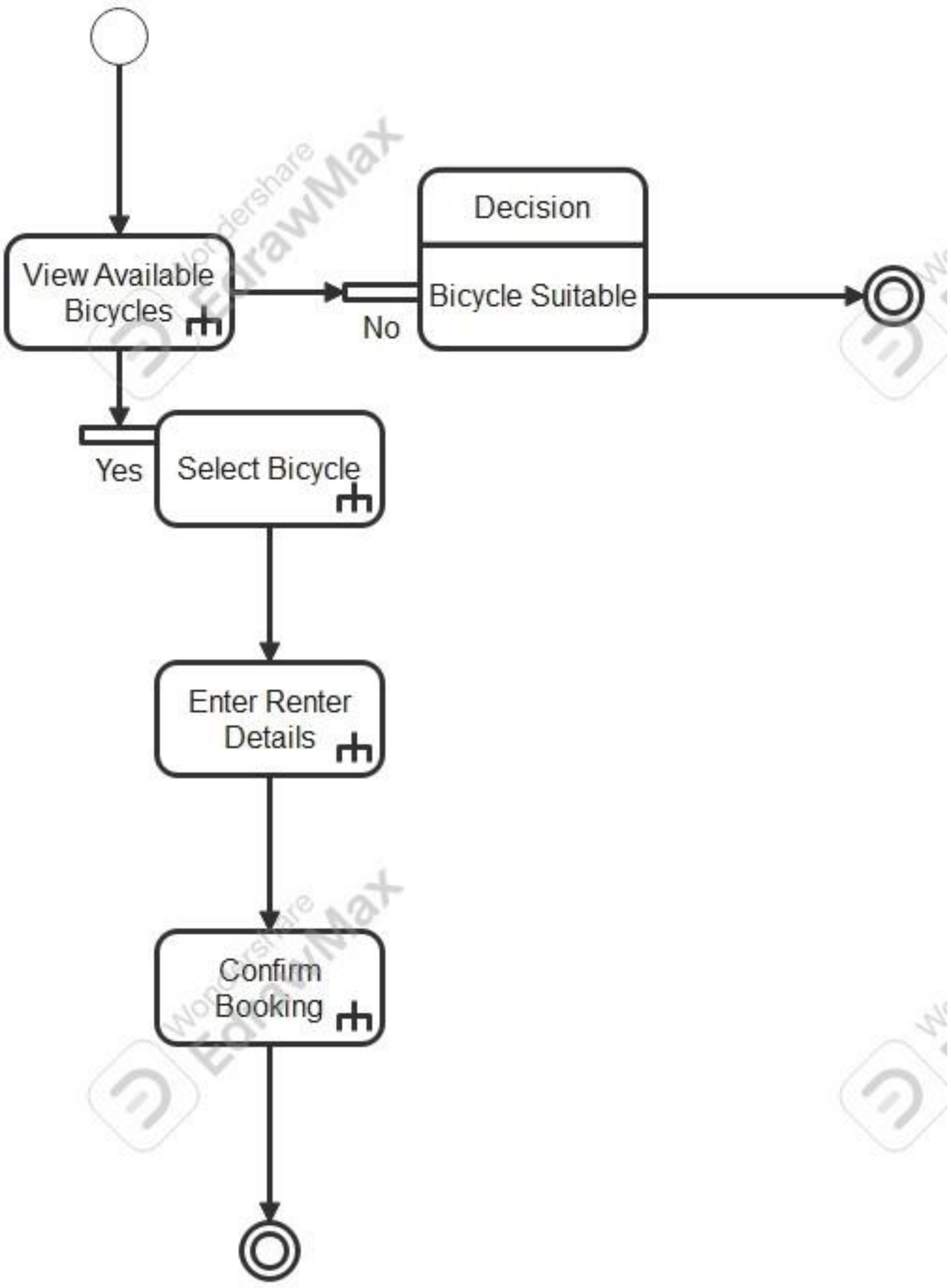
+ BookBicycle(bicycleID: Integer, renterID: Integer, startDate: Date, endDate: Date): Boolean (book a bicycle for a specified period, returns true if successful)
+ CancelBooking(rentalID: Integer): Boolean (cancel an existing booking, returns true if successful)
+ ExtendRental(rentalID: Integer, newEndDate: Date): Boolean (extends the rental period, returns true if successful)

UML Sequence Diagram (Updated)





UML Activity Diagram


View Available Bicycles
Decision
Bicycle Suitable
No
Yes
Select Bicycle
Enter Renter Details
Confirm Booking