Software Engineering Project
“BikeGuard : Smart Bicycle Parking System”

Kelompok 6

Andrew Kristofer Jian	**2206059673**
Azriel Dimas Ash-Shidiqi	**2206059414**
Irfan Yusuf Khaerullah	**2206813290**

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK KOMPUTER

UNIVERSITAS INDONESIA

DEPOK 2024/2025

Table of Contents

List of Pictures

List of Tables

CHAPTER I
INTRODUCTION

1.1 Background

In Universitas Indonesia, bicycles are still a vehicle option that students and academics rarely use. This can happen because there is still a lack of supporting facilities for bicycle transportation, such as the limited number of bicycle parking spaces and lack of bicycle security. To overcome this problem, our group proposed developing a bicycle parking system with features such as QR codes, lock control, real-time monitoring, bike security, and data analytics. Our goal in developing this idea is to support the UI Green Metric program, especially in the Transportation indicator, to increase the number of bicycle users on campus and the Facilities indicator by providing safe and comfortable parking facilities.

1.2 Project aim

Our Primary Object is to develop a comprehensive bicycle parking system for Universitas Indonesia, to solve a challenge that is currently happening in Universitas Indonesia Such as limited bike parking spaces and fewer security issues in the current parking system, By implementing QR code, lock control, real-time monitoring, and data analytics, one of our goals to support the UI Green Metric Program.

In addition to solving these challenges, our goal for developing this project is to increase bicycle usage among the students and university civitas and academics by promoting environmentally friendly transportation options alongside motorcycles and cars, so there will be enhancing the transportation and facilities indicator by providing safe and convenient parking facilities. and our hope is to reduce reliance on motorcycles and cars so that reduce pollution and foster a sustainable and healthy university environmentally

Project Objective

- Reduce pollution
  This Project aims to support one of the GreenMetric UI indicators in the Transportation parameter, With the increasing bicycle usage, it can reduce pollution caused by cars and motorcycles
- Increase the Security of bike
  An alarm system implemented on barriers or locks to prevent theft and illegal parking, so it ensures the bicycles remain safe from unwanted actions
- Real-Time Security Monitoring

The project will include real-time monitoring for bicycle security, This will give user notification if there are any threats to the bicycle's safety, such as attempted theft or other unauthorized access actions

- Analyze Data
  The system will collect and store the data on bicycle parking security aim to provide administrators insights, analyze trends, and enhance safety features to prevent security threats that could happen and improve other features that could be needed it

1.3 Similar Competitor

- Bikeep Parking System

Bikeep bicycle parking system is a software engineering implementation designed by Bikeep, This software engineering implementation is designed to meet the needs of cyclists facing challenges such as shortage of security and accessible parking spaces in urban areas. Besides that, Bikeep has the purpose of reducing global pollution levels by encouraging more people to use bicycles instead of Private transportation such as private cars and private motorcycles. With trusted security features, bikes solve a safety concern problem that has deterred people from using a bike.

The Bikeep Parking system aims to provide a safe and user-friendly solution that encourages bicycle use. Bikeep bicycle parking system uses an implementation of the Internet of Things (IoT) such as a real-time monitoring system, and smart lock to ensure bicycles are kept safe from theft. The Bikeep Parking system is planned for urban areas and high-traffic areas to reduce pollution and reduce congestion in urban environments

- Oonee Smart Bicycle Parking System

Oonee offers a modular and scalable smart bike parking system designed for urban environments, especially in dense metropolitan areas. Oonee focuses not only on providing secure bike parking but also on integrating it with public spaces. Oonee pods are equipped with monitoring, smart locks, and real-time user access via a mobile app, ensuring a high level of safety and convenience for cyclists.

Oonee aims to make cycling an attractive transportation option by overcoming the lack of secure parking in cities. By integrating its system with public transportation hubs and busy urban areas, Oonee encourages people to switch from using motorized vehicles, such as cars and motorcycles, to bicycles, which are more environmentally friendly by reducing pollution and traffic congestion. The company also emphasizes inclusivity, offering customizable options for different communities and urban landscapes.

- Airside Smart Bicycle Parking System

Airside provides a futuristic smart bike parking system by combining advanced IoT and AI technologies to create safe and nice bike parking. Designed in public areas, especially in airports, large corporate campuses, and city centers, Airside's parking system offers automatic bike docking and undocking, allowing for hands-free operation.

Airside offers a parking system with modern features such as smart locks, real-time monitoring,

and AI-driven predictive maintenance. With this, Airside will ensure that user's bikes remain safe while optimizing the use of available parking spaces. The system can be accessed through a mobile app, allowing users to easily find, book, and manage parking spaces. By implementing Airside's parking system in locations with heavy traffic, the company aims to promote the use of bicycles as an environmentally friendly alternative to transportation, reducing congestion and pollution in the environment.

1.4 Literature Review

Smart Parking Systems has become an innovative solution to overcome parking problems in urban areas. The increasing action to use environmentally friendly vehicles has increased the use of bicycles. With the increasing use of bicycles, a system is needed to ensure safety and efficiency in providing parking spaces. Internet of Things (IoT) technology plays a very significant role in the development of Smart Bike Parking Systems, with its ability to connect technology with existing infrastructure. Miorandi et al. (2012), explain that IoT facilitates communication between physical objects such as bicycles and parking infrastructure, enabling more effective automation and monitoring.

Security and efficiency are the main priorities in the Smart Bike Parking System. The use of technology such as sensors is used to detect the presence of bicycles which will later provide real-time monitoring information to users through the application. This sensor technology will greatly assist in creating efficiency in the system, such as information regarding the detection of parking space availability. In addition, the use of the Smart Lock System that can be controlled through the IoT application will strengthen the security of this system. Integration between the locking system and the IoT application to open, lock, and monitor the locking is the main aspect of the development of the Smart Bike Parking System.

In the urban scope, the development of the Smart Bike Parking System has a positive impact on environmental sustainability. With the increasing efficiency of the use of this parking space, people are more encouraged to use bicycles as their vehicles. Ultimately, this is related to reducing pollution and realizing an environmentally friendly city. The use of the Smart Bike Parking System can also be integrated with other public vehicles, as well as public places, which will support efficient mobility in big cities.

1.5 Functions and Modules

Our Project has several Modules and features such as

- Web application
  The system will consist of A Web Application with the system including a user-friendly interface to make it easy to access for users
- QR Code
  The project will be equipped with QR codes so the user can use them to easily lock and unlock the bicycles and ensure their bicycle have secure access
- Automated Lock system

The system will be integrated with smart features to control the Hardware locking system such as automatically locking and unlocking their bicycles via their mobile phone

- Security
  The alarm system on bike locker prevents from forced actions and legal actions enhancing bicycle safety

- Real-time monitoring system
  The system will include a sensor that monitors the condition of user bicycles, user will be able to view their bicycle in real-time and receive notifications along with the administrator if the security of their bicycle is in danger
- Data analytic system
  Data that be analyzed by the system will be used to track and analyze application usage in greater detail, including security aspects and other features that could potentially be improved

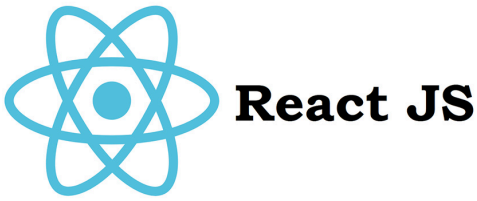
1.6 Tools

Software

Backend
For our smart bike parking system , In the backend we are using Node.js, We chose Node.js because of its speed and scalability to handle large-scale real-time applications and ease in managing HTTP requests

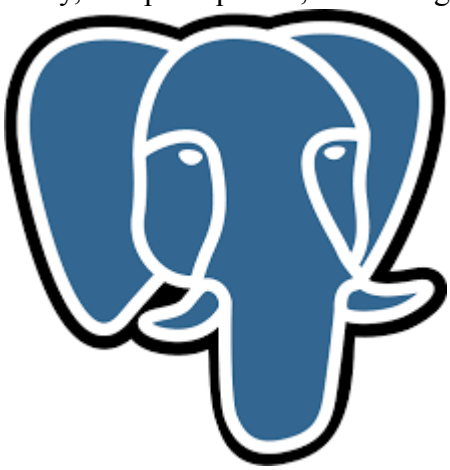
Picture 1 Node Js

Frontend
For our smart bike parking system, in the frontend, we are developing a user-friendly React JS for building the user interface and user experience system for our website, React JS provides the ease of creating Responsive and interactive UI



Picture 2 React Js

Database
For our smart bike parking system, In the database, We are using PostgreSQL as our database because of its scalability, complex queries, and strong security system



Picture 3 PostgreSQL

Hardware and IoT

Our Project system will integrate a combination of hardware and IoT. This integration enables automation and monitoring systems in real time. We are using FreeRTOS to manage real-time tasks that support our system features.

- Smart Locks
  We are using a smart lock system integrated with the website to control auto-locking for the bike. This system will allow users to unlock and lock their bikes remotely

- Alarm System
  We are using an alarm system on the barrier to alert the user of any attempted theft or unauthorized access

- Parking Sensor
  We are using a parking sensor to detect the presence of bicycles. So users can see information about existing parkin real-time through the application or website

1.7 Resources

To implement our Smart bike parking system, several key resources and components are required
Our project utilizes these resources

- Software: Node.js for the backend, React native or React JS for frontend, PostgreSQL for the database, FreeRTOS for managing real-time tasks, and Visual Studio code for code development.
- Hardware: Smart locks, parking sensors, alarm system, and microcontroller ESP32.
- Communication: Wi-Fi modules, Notification system with Real-time system.
- Development Project: Visual Studio code for backend and frontend code development, Android studio for frontend code development, Arduino IDE for IoT code development. And GitHub for our version control and digital collaboration workspace for our group.

1.8 Risk Analysis

1.8.1 Technical Risks

a. System Integration Failures: challenge for integrating the system of smart bicycle parking with university infrastructure such as transportation systems and internet connection.
b. Software Failures: Bugs on web applications or backend and frontend systems could lead to the system and application running abnormally and not working properly.
c. Hardware Malfunction: Hardware plays a crucial role in our project, so a failure or malfunction of hardware components such as damage to the sensor, smart lock, and monitoring system can disrupt the operations of the project.

1.8.2 Security Risks

a. Unauthorized Access: the system could be accessed by irresponsible third parties by hacking the application and system or exploiting the QR code and smart locks system.
b. Data Violation: the leakage of data users, such as parking information, payment data, and parking history. These problems can threaten the users of smart bicycle parking applications. This could damage the system's reputation and lead to a legal issue.
c. Vandalism: The system can be a target for theft or be damaged intentionally by irresponsible third parties such as damaging hardware components in the system such as the system sensor and system smart lock.

1.8.3 Operational Risks

a. System Maintenance: The system requires regular maintenance and there could be downtime potential at unpredictable times that could disrupt the daily operation, so it needs more attention to system maintenance.
b. User Adoption: Users may be slow to adopt the new system; it may take them more time than expected.
c. Insufficient Training For Operational Staff: Those who manage the bicycle parking system may lack understanding of the system technology which could lead to operational error, thus requiring training to solve these problems.

1.8.4 Budgetary Risks

a. Cost Overrun: Estimated project costs exceed the planned budget due to poor planning or there could be problems during the project development process.
b. Unforeseen Expenses: There are unexpected costs in project development, such as additional hardware components, software maintenance, or costs in dealing with problems that occur in project development.
c. Inefficient Resource Utilization: Inefficient use of resources such as purchasing inappropriate and suboptimal hardware components, and wasting time in the project development process. These things could lead to higher costs.

1.8.5 Schedule Risks

a. Development Delays: System development is delayed due to technical issues or lack of worker resources that result in the project not being completed on time.
b. Underestimation of Task Complexity: The complexity of a project task may be overlooked or underestimated leading to unexpected challenges or issues during the development project. As a result, the project could take longer to finish than expected. These can lead to a delay in the overall development project schedule.
c. Unforeseen Events or Delays: Unforeseen disruptions or problems such as technical issues with the systems or hardware components, or health issues with the members involved in developing the project. These unforeseen disruptions could disrupt the workflow and cause the project not to be completed on time.

CHAPTER 2
PROJECT MANAGEMENT

2.1 Roles and Responsibilities

Name	Roles	Responsibilities
Andrew Kristofer Jian	Frontend Developer	- Develop User interface for web application using React JS - Ensure the frontend design is user-friendly dan responsive
Azriel Dimas-Ash Shidiqi	Backend & Frontend Developer	- Develop the backend infrastructure using Node.js - Ensure Scalability and the performance of the backend and database system - Help Frontend Code
Irfan Yusuf Khaerullah	Project Coordinator, IoT & Frontend Developer	- Coordinate project activities - Ensure alignment of project goals and timelines - Integrate IoT components and manage the development of real-time monitoring and data analytics system - Help Frontend Code

Table 1 Team Roles and Responsibilities

2.2 Schedule

2.2.1 Overview Schedule

Project Activity	September				October				November				December			
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Planning																
Pre-Production																
Production																
Testing																
Pre-Launch																
Launch																
Post-Production																

Picture 4 Schedule

2.2.2 Detail Schedule

Our Group uses the waterfall model to create our schedule. The Waterfall model is straightforward and idealistic because all software development life cycle models are based on the classical waterfall model. These are the details of our schedule timeline.

2.2.2.1 Planning (Week 2 - Week 4 of September)

Before All of the members started to do their tasks, our group determined the idea of our project that related to UI GreenMetric. After a brainstorming session, we had an idea to create a smart security system for bicycles at Universitas Indonesia to increase bicycle use and decrease the use of cars and motorcycles. At this stage, we brainstormed together and identified what would be needed in the project later, such as hardware and software components, and we also determined each task for our project and made estimates of risks and mitigation that might occur.

2.2.2.2 Pre-Production (Week 4 of September - Week 1 of October)

In This stage, we finalize the project and realize the idea that we laid out in the planning stage. At this stage, our group prepares everything that might be needed when making the project**.**

2.2.2.3 Production (Week 2 of October - Week 3 of November)

This Stage is the longest process and takes a long time. This stage is the most difficult stage of the project such as creating a prototype, creating project design, creating code for the backend and frontend, and integrating IoT components to implement the real-time system and where every other part is created.

2.2.2.4 Testing (Week 4 of November)

In this stage, We do testing on all our project components from software, hardware, and database. We also test communication integration between software and hardware components, also we resolve our project bugs that happened during the testing stage.

2.2.2.5 Pre-Launch (Week 1 of December)

In this stage, we made adjustments to the testing stage, we also got an overthinking attack about the response of people who are going to use our project later. We also do user acceptance tests for certain individuals to validate our project. In the end, we are preparing for the deployment system to be launched.

2.2.2.6 Launch (Week 2 of December)

In this stage, our project will be officially released, Our Smart Parking Bicycle system will be launch-worthy, We will keep monitoring the system for

initial feedback from users.

2.2.2.7 Post Production (Week 3 of December)

In this stage, the work is not over even after the system has been released and launched. It is a common thing when a system that has just been launched has bugs that were not caught during the testing stage. Therefore our group will continue to optimize the system based on user feedback and gathering data such as the effectiveness of our system features to ensure a long-term maintenance strategy.

2.3 Risk Management

The BikeGuard project has identified several key risk categories and developed mitigation strategies for each. Here's a summary of our risk management approach.

2.3.1 Technical Risks

We've addressed potential system integration challenges, software bugs, and hardware malfunctions. Our strategies include:

a. Conducting thorough compatibility assessments and developing phased integration plans
b. Implementing rigorous testing protocols and quality assurance processes
c. Establishing regular maintenance schedules and keeping spare parts on hand
d. Implementing redundancy for critical systems to avoid single points of failure
e. Implementing continuous integration and deployment practices

2.3.2 Security Risks

To combat unauthorized access, data breaches, and vandalism, we plan to:

a. Implement strong encryption and multi-factor authentication
b. Conduct regular security audits and staff training on data handling
c. Use tamper-evident seals on hardware
d. Implement strict access controls following the principle of least privilege
e. Regularly update and patch all system components

2.3.3 Operational Risks

We're tackling system maintenance issues, slow user adoption, and staff training gaps by:

a. Scheduling maintenance during off-peak hours and implementing redundant systems
b. Conducting user workshops and creating user-friendly guides
c. Implementing a user feedback system for continuous improvement
d. Creating a knowledge base for common issues and solutions

2.3.4 Budgetary Risks

To manage potential cost overruns, unforeseen expenses, and resource inefficiencies, we will:

a. Implement strict budget monitoring and include contingency funds
b. Conduct thorough initial cost analyses and maintain flexible budget allocations
c. Use resource tracking tools and promote a culture of cost-consciousness
d. Implement s just-in-time inventory system for hardware components
e. Implement a prioritization system for expenses

2.3.5 Schedule Risks

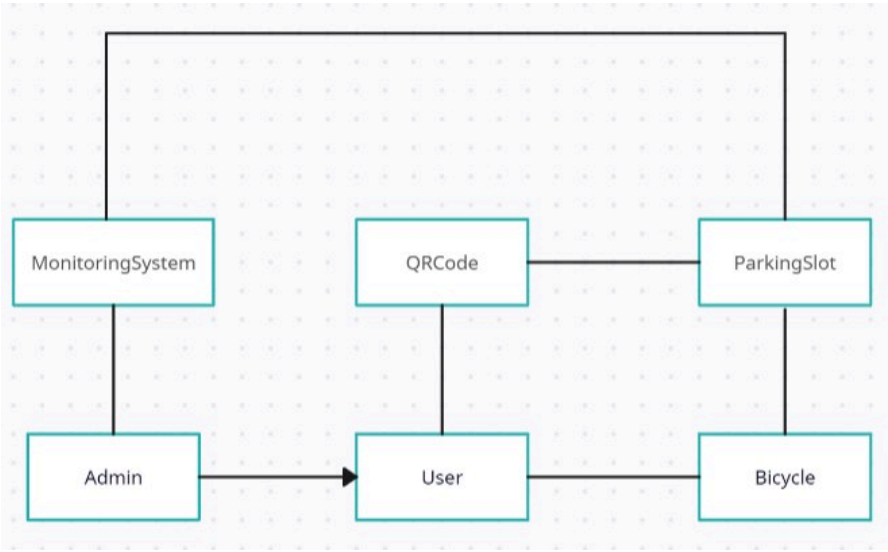
We're addressing potential delays and task complexity issues through:

a. Careful planning and detailed documentation at each phase before moving to the next
b. Implementing strict change control processes to manage scope creep
c. Regular progress tracking against predefined milestones and deliverables
d. Using project management tools to visualize the critical path and identify potential bottlenecks
e. Implementing a resource forecasting system to anticipate and address potential resource shortages

By implementing these strategies within our waterfall model, the BikeGuard project aims to minimize disruptions to our sequential development process and ensure smooth progression through each phase of the project lifecycle.

CHAPTER 3
DESIGN

3.1 Class Diagram



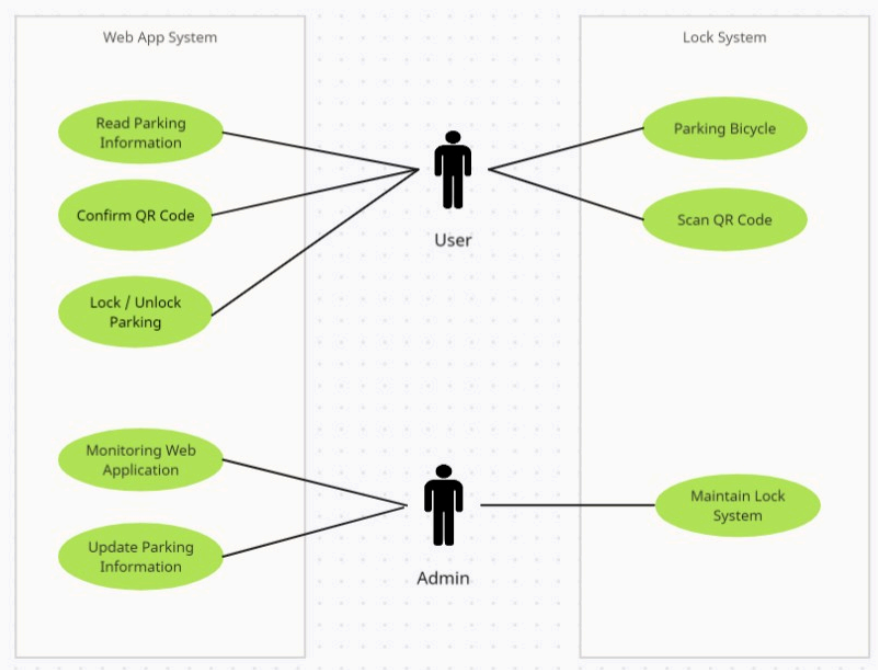
Picture 5 Class Diagram

The class diagram shown above illustrates the structure of the web application that we will create later. Each class in the diagram describes an object that exists in the web application:

1. User
   User Class will represent the general user of the BikeGuard system. This Class will store user's information, such as UserID, name, email, and password. The User Class will be related to the Bicycle Class as its owner and QRcode Class for the locking system.
2. Admin
   Admin Class is a type of User class, but has additional rights to manage the system. This class will store admin's information, such as adminID, name, email, and password. Admin Class will be related to the Monitoring Class to monitor and manage the system and parking lot.
3. Bicycle
   Bicycle Class represents the bicycles parked in the system. Each bicycle is associated with a specific user. So, this class will be related to User Class and also ParkingSlot Class to detect whether there is a bicycle parked in the parking lot or not.
4. QRCode
   QRCode Class will represent a unique QR code given to each parking lot. Used to lock and unlock bicycles in certain parking lots that match the QR Code. This class will be related to the User Class as the one who scans the QR Code and the ParkingSlot Class as the place where the QR Code is located.
5. ParkingSlot
   ParkingSlot Class will represent a parking slot for bicycles. Each slot can be managed by the admin and filled by one bicycle. This class will be related to the QRCode Class as the locking system of the parking slot, the Bicycle Class as the object that fills this class, and the MonitoringSystem Class as the manager and monitor of the parking slot.

6. Monitoring System
   MonitoringSystem Class will monitor the bike condition in real-time, such as the bike's safety status and the availability of parking spaces. Therefore, the Monitoring Class will relate to the Admin Class as the system administrator and the ParkingSlot Class.

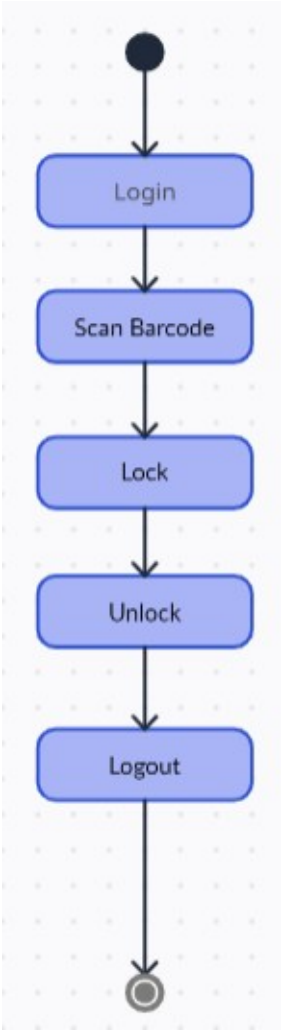
3.2 Use Case Diagram



Picture 6 Use Case Diagram

The use case diagram above illustrates the main features of the BikeGuard system. Users can access the web application to read parking information, scan QR codes to lock or unlock their bikes, and monitor bike conditions in real-time. On the other hand, Admin has access to monitor parking status through the Monitoring Web Application feature and update parking information by adding or managing parking slots. Admin is also responsible for maintaining the lock system, ensuring that the bike locking system is functioning properly. All features in this system are carried out to maintain and provide security and comfort in using the BikeGuard web application.

3.3 State Diagram



Picture 7 State Diagram

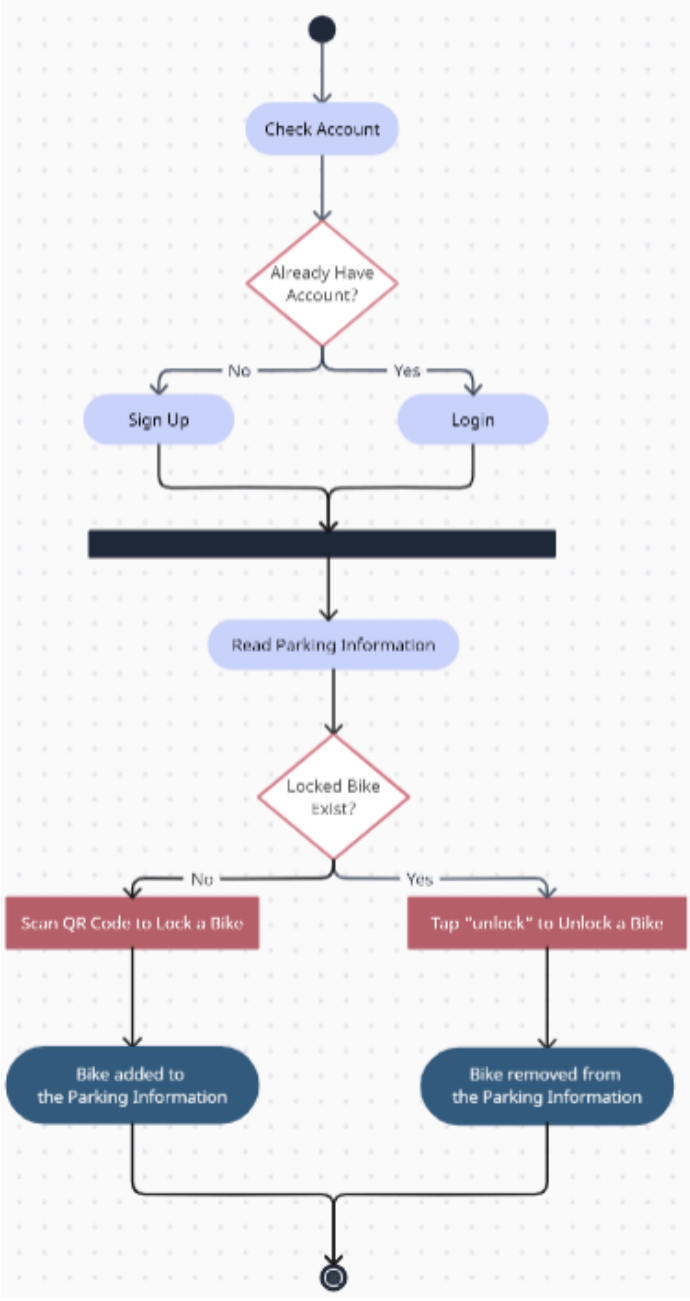
The state diagram for BikeGuard illustrates all the states that a user will experience. Here is an explanation of the state diagram:

1. Login
   The user starts by logging into the system using the web application. Once the user's credentials are verified, the user is given a session to access the application's features.
2. Scan QR Code
   After logging in, the user enters the Scan QR Code state, where they can scan the QR code attached to the parking slot to perform further actions.
3. Lock

If the bike is not locked, the user enters the Lock state, where they can lock the bike in the parking slot through the application.

4. Unlock
   Once the bike is locked, the user can return to the parking slot to unlock their bike. This process takes the user to the Unlock state, where the user will be able to unlock the bike through the application.
5. Logout
   After finishing using the parking feature, the user can choose to log out of the application. The Logout state closes the user's session.

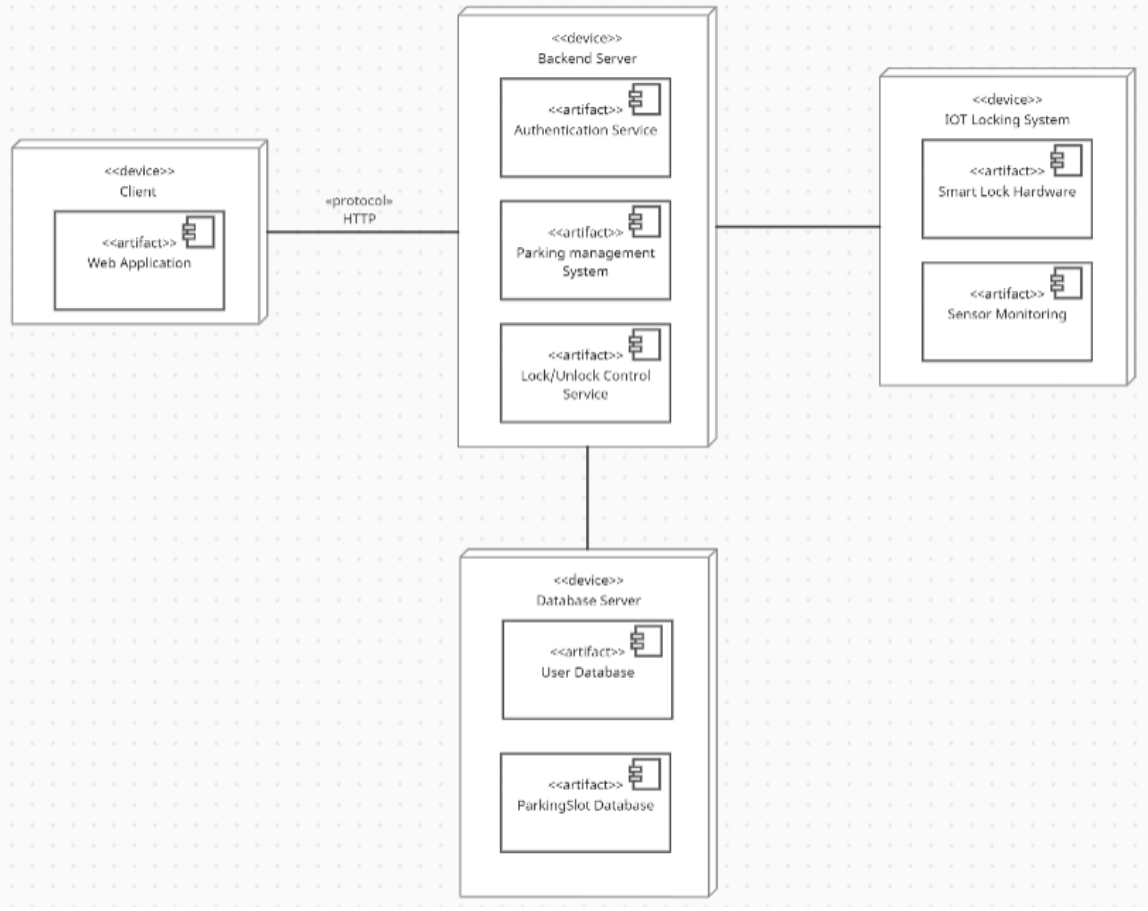
3.4 Activity Diagram



Picture 8 Activity Diagram

Activity Diagram will represent the activities that can be done by the user when running the web application. The application will start with a login page for users who have an account. If the user does not have an account, then the user must create an account to continue. After that, the user will enter the home page of the web application. On the homepage of the application, you will be able to see the status of information from the parking lot. If you have not parked your bike, you can scan the QR Code to lock your bike which will then update the parking lot information. Meanwhile, if you have parked your bike, you can unlock your bike by pressing the unlock button which will delete information from the parking lot.

3.5 Deployment-Component Diagram



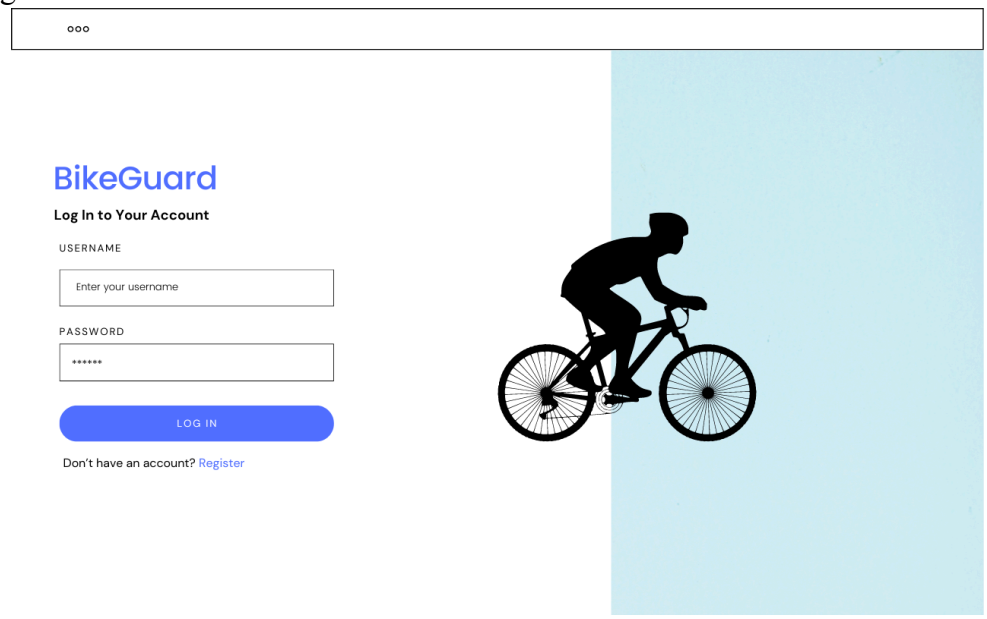
Picture 9 Deployment Component Diagram

The Deployment-Component diagram shows the main interactions between four nodes: Client Node, Server Node, Database Node, and IoT Node, which are interconnected through a network. The Client Node (smartphone or browser) communicates with the Server Node using the HTTP/HTTPS protocol to log in, scan QR codes, and send commands to lock or unlock the

bike. The Server Node accesses the Database Node to retrieve and update user and parking data and communicates with the IoT Node via a wireless network (Wi-Fi or Bluetooth) to automatically control the bike lock and monitor the bike's sensor status in real time.

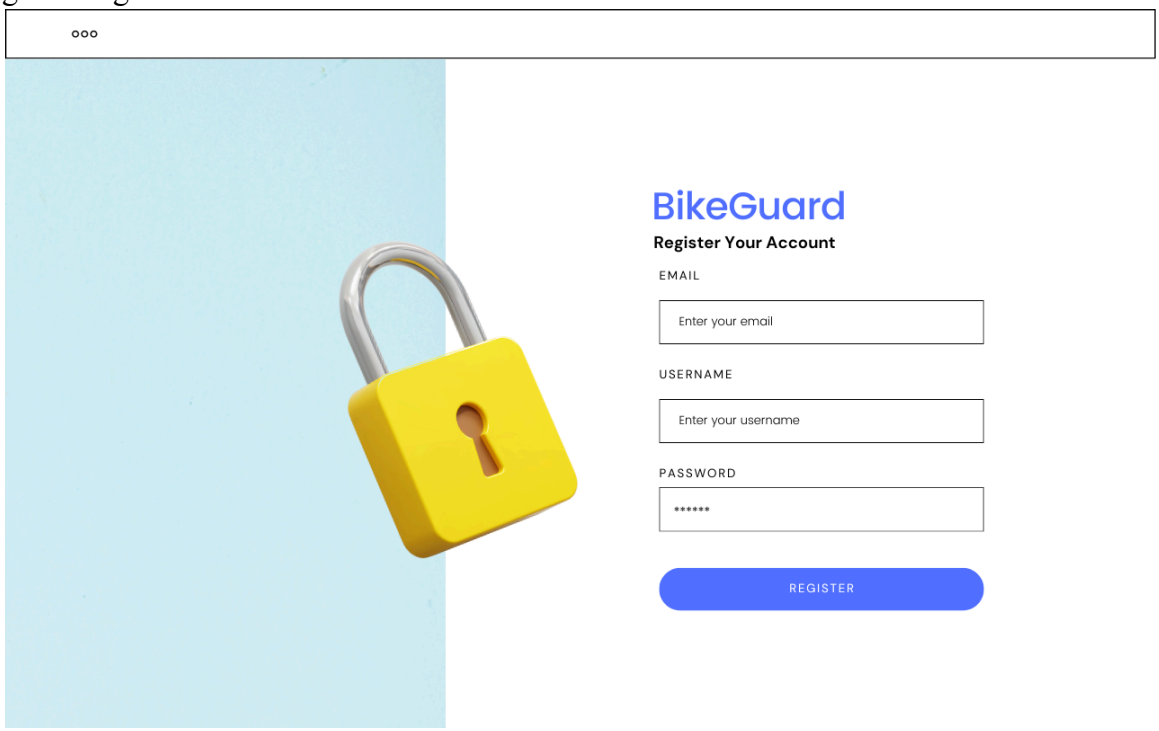
3.6 Website Design

Login Page



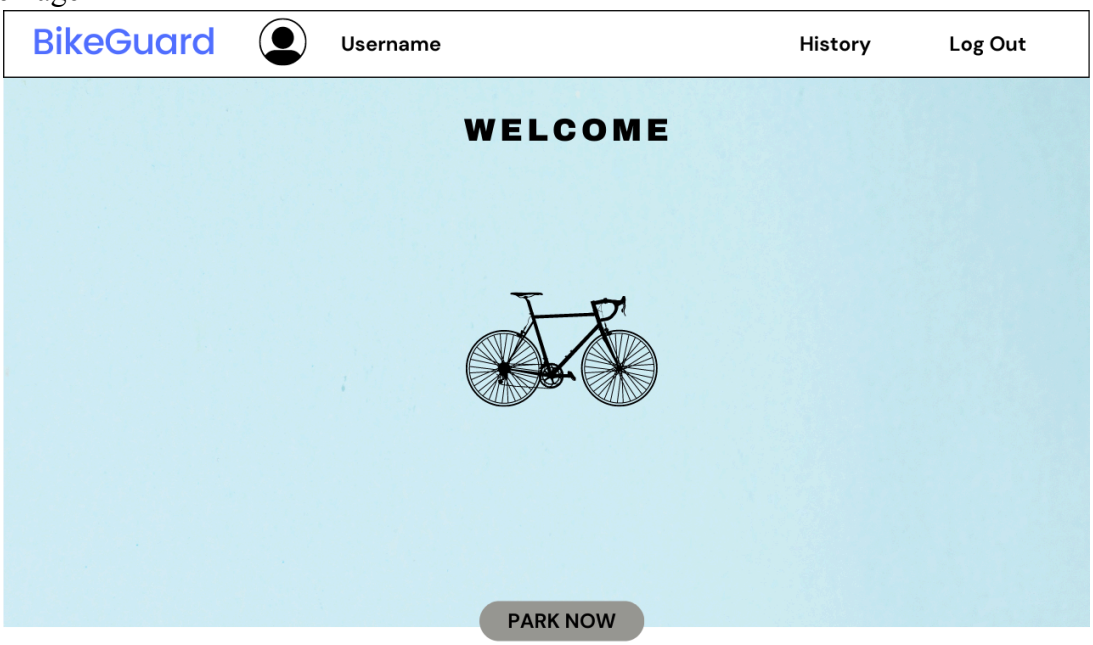
Picture 10 Login Page Design

Register Page



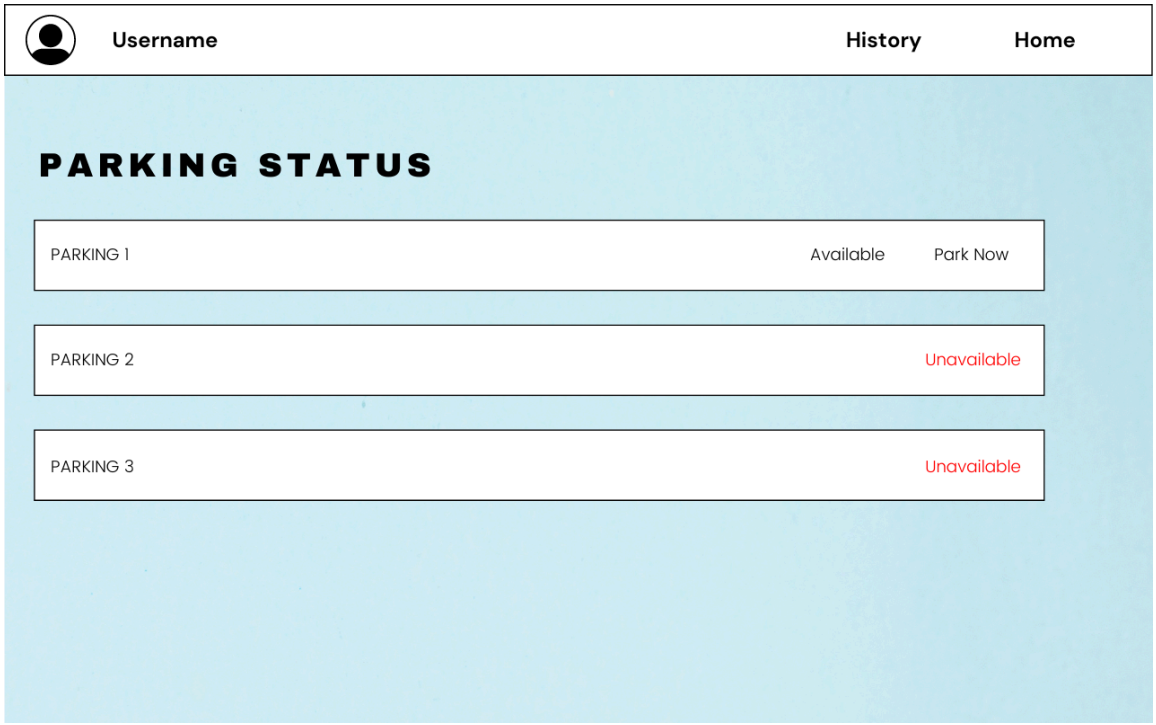
Picture 11 Register Page Design

Home Page



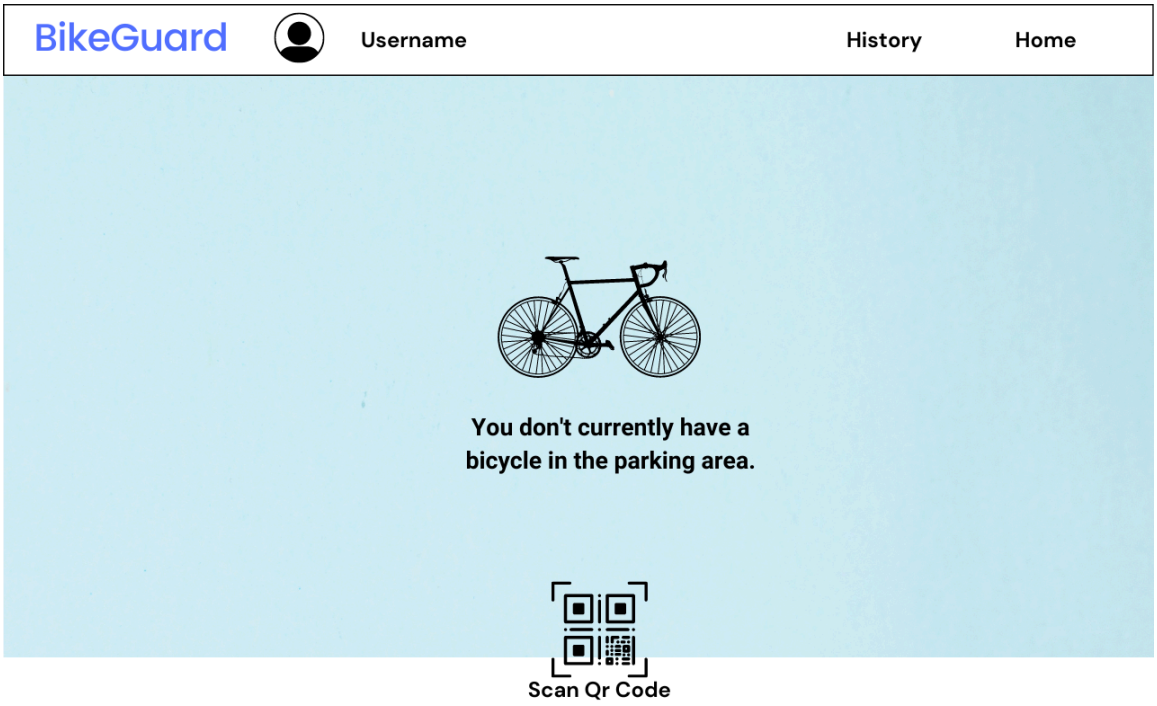
Picture 12 Home Page Design

Parking Status



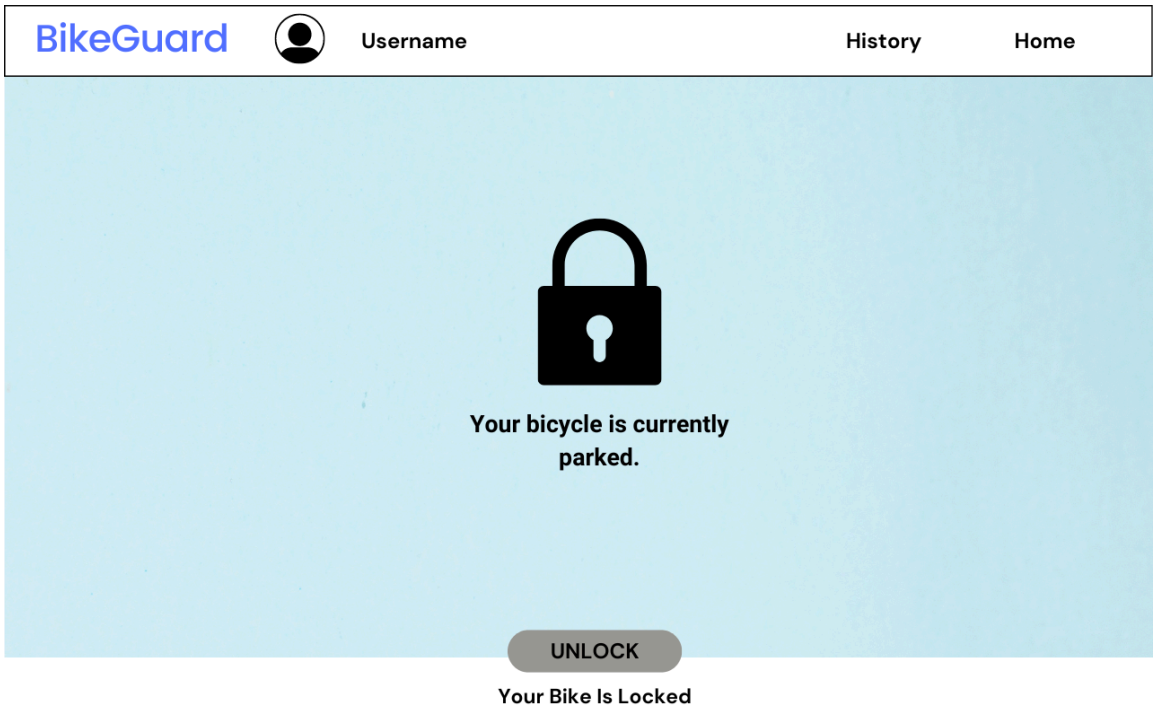
Picture 13 Parking Status Design

Parking Page (Lock)



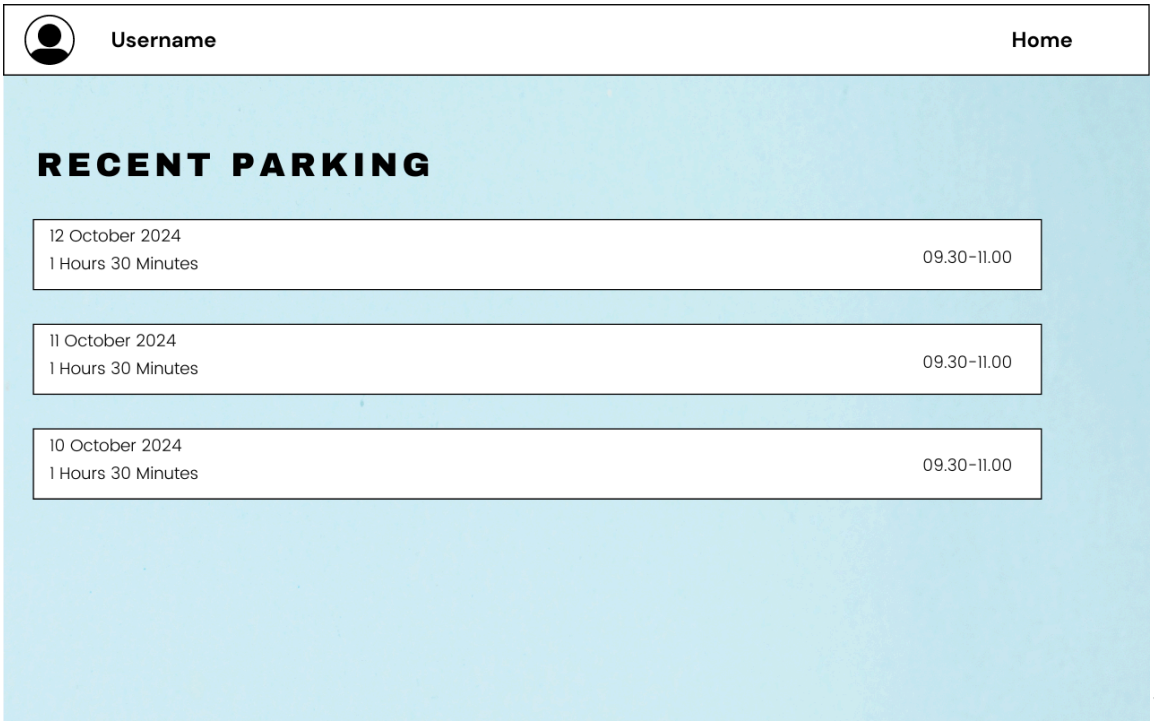
Picture 14 Parking Page Lock Design

Parking Page (Unlock)



Picture 15 Parking Page unlock Design

History Page



\

Picture 16 History Page unlock Design

CHAPTER 4
Coding Implementation

4.1. Back-End

4.1.1. Admin Register

```
exports.registerAdmin = async (req, res) => {
    const { name, email, password, role } = req.body;

    if (role !== 'admin') {
        return res.status(403).json({ message: 'Access denied:
Invalid role for admin registration' });
    }

    try {
        const result = await pool.query(
            `INSERT INTO users (name, email, password, role)
             VALUES ($1, $2, $3, 'admin') RETURNING *`,
            [name, email, password]
        );

        res.status(200).json(BaseApiResponse('Successfully
created admin', result.rows[0]));
    } catch (error) {
        console.error(error);
        res.status(500).json(BaseApiResponse(error.message,
null));
    }
};
```

4.1.2. Admin Login

```
exports.loginAdmin = async (req, res) => {
    const { email, password, role } = req.body;

    try {
        const result = await pool.query(`SELECT * FROM users WHERE email
= $1 AND password = $2 AND role = $3`, [email, password, role]);

        if(result.rows.length == 0)
            return res.status(404).json(BaseApiResponse('User Not Found',
```

```
null));

        return res.status(200).json(BaseApiResponse("Login Succesful",
result.rows[0])); // Mengembalikan data user yang baru ditambahkan
    }

    catch (error) {
        console.log(error);
        return res.status(500).json(BaseApiResponse(error.message,
null));
    }
}
```

4.1.3. User Register

```
exports.register = async (req, res) => {
    const { name, email, password, role } = req.body;

    if (role && role !== 'user') {
        return res.status(403).json({ message: 'Access denied: Invalid
role for user registration' });
    }

    try {
        const result = await pool.query(
            `INSERT INTO users (name, email, password, role)
             VALUES ($1, $2, $3, 'user') RETURNING *`,
            [name, email, password]
        );

        res.status(200).json(BaseApiResponse('Successfully created user',
result.rows[0]));
    } catch (error) {
        console.error(error);
        res.status(500).json(BaseApiResponse(error.message, null));
    }
};
```

4.1.4. User Login

```
exports.login = async (req, res) => {
    const { email, password, role } = req.body; // Menambahkan role ke
body request
```

```
    try {
        // Query untuk memeriksa email, password, dan role
        const result = await pool.query(
            `SELECT * FROM users WHERE email = $1 AND password = $2 AND
role = $3`,
            [email, password, role]
        );

        // Jika tidak ditemukan user yang sesuai
        if (result.rows.length === 0)
            return res.status(404).json(BaseApiResponse('User Not Found
or Role Mismatch', null));

        // Jika ditemukan, kembalikan data user
        return res.status(200).json(BaseApiResponse("Login Successful",
result.rows[0]));
    }
    catch (error) {
        console.log(error);
        return res.status(500).json(BaseApiResponse(error.message,
null));
    }
};
```

4.1.5. Admin Create QR Code

```
exports.createQRCode = async (req, res) => {
    const { code, expiration_date, associated_parking_slot, user_id } =
req.body;  // Ambil user_id dari request body

    if (!user_id) {
        return res.status(400).json(BaseApiResponse("User ID harus
disertakan", null));
    }

    try {
        // Cek role user berdasarkan user_id yang dikirimkan
        const userCheck = await pool.query('SELECT role FROM users WHERE
id = $1', [user_id]);

        if (userCheck.rows.length === 0 || userCheck.rows[0].role !==
'admin') {
```

```
            return res.status(403).json(BaseApiResponse("Akses ditolak.
Hanya admin yang dapat membuat QR code.", null));
        }

        // Jika user adalah admin, lanjutkan membuat QR Code
        const result = await pool.query(
            `INSERT INTO qrcodes (code, expiration_date,
associated_parking_slot) VALUES ($1, $2, $3) RETURNING *`,
            [code, expiration_date, associated_parking_slot]
        );
        const qrData = result.rows[0];
        res.status(201).json(BaseApiResponse("QR Code berhasil dibuat",
QRCodeResponse(qrData)));
    } catch (error) {
        console.error(error);
        res.status(500).json(BaseApiResponse("Gagal membuat QR Code",
null));
    }
};
```

4.1.6. Verify QR Code

```
exports.verifyQRCode = async (req, res) => {
    const { code } = req.body; // QR code is sent in the request body

    try {
        // Query to find the QR Code by code value
        const result = await pool.query(
            `SELECT * FROM qrcodes WHERE code = $1`,
            [code]
        );

        if (result.rows.length === 0) {
            return res.status(404).json(BaseApiResponse("QR Code not
found", null));
        }

        const qrData = result.rows[0];

        // Check if the QR code is valid and not expired
        const currentDate = new Date();
        const expirationDate = new Date(qrData.expiration_date);
```

```
        if (!qrData.is_valid) {
            return res.status(400).json(BaseApiResponse("QR Code is
invalid", null));
        }

        if (expirationDate < currentDate) {
            return res.status(400).json(BaseApiResponse("QR Code has
expired", null));
        }

        // If valid and not expired, return success response
        res.status(200).json(BaseApiResponse("QR Code is valid",
QRCodeResponse(qrData)));
    } catch (error) {
        console.error(error);
        res.status(500).json(BaseApiResponse("Failed to verify QR Code",
null));
    }
};
```

4.1.7. Admin Create Parking Slot

```
exports.createParkingSlot = async (req, res) => {
    const { location, user_id } = req.body;  // Ambil user_id dari
request body (misalnya dari form login)

    if (!user_id) {
        return res.status(400).json(BaseApiResponse("User ID harus
disertakan", null));
    }

    try {
        // Cek role user berdasarkan user_id yang dikirimkan
        const userCheck = await pool.query('SELECT role FROM users WHERE
id = $1', [user_id]);

        if (userCheck.rows.length === 0 || userCheck.rows[0].role !==
'admin') {
            return res.status(403).json(BaseApiResponse("Akses ditolak.
Hanya admin yang dapat membuat slot parkir.", null));
        }
```

```
        // Jika user adalah admin, lanjutkan membuat slot parkir
        const result = await pool.query(
            `INSERT INTO parking_slots (location) VALUES ($1) RETURNING
*`,
            [location]
        );
        const slotData = result.rows[0];
        res.status(201).json(BaseApiResponse("Slot parkir berhasil
dibuat", ParkingSlotResponse(slotData)));
    } catch (error) {
        console.error(error);
        res.status(500).json(BaseApiResponse("Gagal membuat slot parkir",
null));
    }
};
```

4.1.8. Availability Parking

```
exports.getAllParkingSlots = async (req, res) => {
    try {
        const result = await pool.query(
            `SELECT parking_slots.*,
                    users.name AS reservedBy,
                    CASE
                        WHEN parking_slots.reserved_by IS NOT NULL THEN
'Unavailable'
                        ELSE 'Available'
                    END AS status
            FROM parking_slots
            LEFT JOIN users ON parking_slots.reserved_by = users.id`
        );

        const parkingSlots = result.rows.map(row => ({
            id: row.id,
            location: row.location,
            isOccupied: row.is_occupied,
            reservedBy: row.reservedBy,
            status: row.status
        }));

        res.status(200).json(BaseApiResponse("Semua slot parkir berhasil
diambil", parkingSlots));
    } catch (error) {
```

```
        console.error(error);
        res.status(500).json(BaseApiResponse("Gagal mengambil semua slot
parkir", null));
    }
};
```

4.2. Front-End

4.2.1. QR Code Scanning

```
  const handleScan = (data) => {
    if (data && data !== qrData) {
      setQrData(data);
      verifyQRCode(data);
    }
  };

  const verifyQRCode = (scannedCode) => {
    setError(null);
    axios
      .post("http://localhost:3000/qr-code/verify", { code: scannedCode
})
      .then((response) => {
        const { message, data } = response.data;

        if (message === "QR Code is valid" && data.associatedParkingSlot)
{
          localStorage.setItem("parkingSlotId",
data.associatedParkingSlot);
          navigate("/bike-in-use");
        } else {
          setError("QR Code valid tetapi tidak terkait dengan slot
parkir.");
        }
      })
      .catch((error) => {
        console.error("Error during QR code verification:", error);
        setError("Gagal memverifikasi QR Code. Silakan coba lagi.");
      });
  };
```

4.3. IoT

4.3.1. Detect Bike

```
void sensorTask(void *pvParameters) {
  for (;;) {
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);

    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.034 / 2;

    xSemaphoreTake(mutex, portMAX_DELAY);
    if (distance < 200) {
      isBikePresent = true;
    } else {
      isBikePresent = false;
    }



    unsigned long currentMillis = millis();
    if (abs(distance - lastDistance) > 1 ||
        currentMillis - lastDistancePrintTime >=
distancePrintInterval) {
      Serial.print("Distance: ");
      Serial.println(distance);
      lastDistance = distance;
      lastDistancePrintTime = currentMillis;
    }
    xSemaphoreGive(mutex);

    vTaskDelay(500 / portTICK_PERIOD_MS);
  }
}
```

```
xSemaphoreTake(mutex, portMAX_DELAY);
 if (isBikePresent) {
```

```
  Serial.println("Sepeda terdeteksi di lokasi parkir.");
 } else {
  Serial.println("Lokasi parkir kosong.");
 }
 xSemaphoreGive(mutex);
```

4.3.2. Locking System

```
void lockControlTask(void *pvParameters) {
  for (;;) {
    xSemaphoreTake(mutex, portMAX_DELAY);
    if (isLocked) {
      myservo.write(0);
      digitalWrite(buzzer, HIGH);

      if (lastLockState != isLocked) {
        Serial.println("Sepeda sedang terkunci.");
        lastLockState = isLocked;
      }
    } else {
      myservo.write(90);
      digitalWrite(buzzer, LOW);

      if (lastLockState != isLocked) {
        Serial.println("Sepeda terbuka.");
        lastLockState = isLocked;
      }
    }
    xSemaphoreGive(mutex);

    vTaskDelay(100 / portTICK_PERIOD_MS);
  }
}
```

CHAPTER 5
Unit Testing

5.1. Introduction

Testing is a critical phase in the software development life cycle of our Smart Parking BikeGuard System, Our group implements software testing procedures and documentation regarding product testing to validate our system. BikeGuard integrates technologies such as NodeJS, ReactJS, ExpressJS, and IoT Real-time operating systems. This combination of technologies aims to create an attractive and useful website for the Universitas Indonesia community to park their bicycles with a guaranteed security system

The primary goal of this testing phase is to ensure the usability, reliability, and efficiency of BikeGuard, This Document contains the approach of BikeGuard users to thoroughly test BikeGuard's features. The strategy will include the scenario to handle common and specific cases, ensuring that BikeGuard remains a good and efficient web application for all its users

5.2. Testing Objectives

The primary objectives of testing for BikeGuard are as follows:

- Main Functionality:
  Ensure the key features such as Account Validation, QR Scanning, and Locking System are working properly according to system specifications.
- User Experience (UX) Validation:
  Ensure the web application provides a smooth and intuitive UX, both in terms of navigation and interaction with IoT features.
- System Security:
  Ensure the system security mechanisms such as user authentication and access control to the locking system are working properly to protect user data and prevent unauthorized access.
- System Performance:
  Ensure the system can handle real-time data interactions without significant performance degradation, especially for the features that require interaction with IoT devices.
- Data Consistency and Synchronization:
  Ensure that data sent between the web interface and IoT devices is consistent and updated in real-time, without data loss or conflict.

5.3. Passing Criteria

The system is successful in the testing phase if the following passing criteria are fulfilled from the test

- During the testing phase, all critical features in the BikeGuard system can pass the specified test cases
- During the testing phase, the metrics of performance meet or exceed pre-established benchmarks
- During the testing phase, the usability testing indicates a positive user experience and feedback without any functionality issues
- During the testing phase, security tests identify all vulnerabilities and resolve them
- During the testing phase, Integration tests ensure seamless interaction validation between all system components including from the Backend, Frontend, and IoT devices
- No high-severity bugs are present during the deployment time

5.4. Quality Risks

BikeGuard potential issues that could affect the quality of the system being tested are:

- QR Code Scanning Failures
  The system may fail to read QR codes correctly, especially in low-light conditions or with damaged QR codes.
- Sensor Misreads:
  The IoT sensors may give inaccurate readings that confuse users while checking the availability status.
- System Downtime or Latency
  The system may experience delay due to server issues or IoT communication failure that can affect real-time interactions.
- Security Vulnerabilities
  Security risks such as unauthorized access or vulnerabilities in the authentication system could compromise user data, including sensitive activities like locking/unlocking bikes.
- Inconsistent Data Synchronization
  There could be issues with syncing data between the frontend, backend, and IoT devices, leading to discrepancies in the parking status or user actions (failed locks or incorrect availability status).
- User Interface (UI) Issues
  The UI may not provide a smooth user experience across different devices, screen sizes, or operating systems, leading to lower user engagement.

5.5. Test Approach

- Test Planning
  Develop a complete test plan including detailed objectives, scope, timeline, and resources for BikeGuard
- Unit Testing
  Focusing on BikeGuard Features such as account creation, login validation, QR code Scanning, and other BikeGuard Features with Test data such as Mock data sets like dummy user accounts,

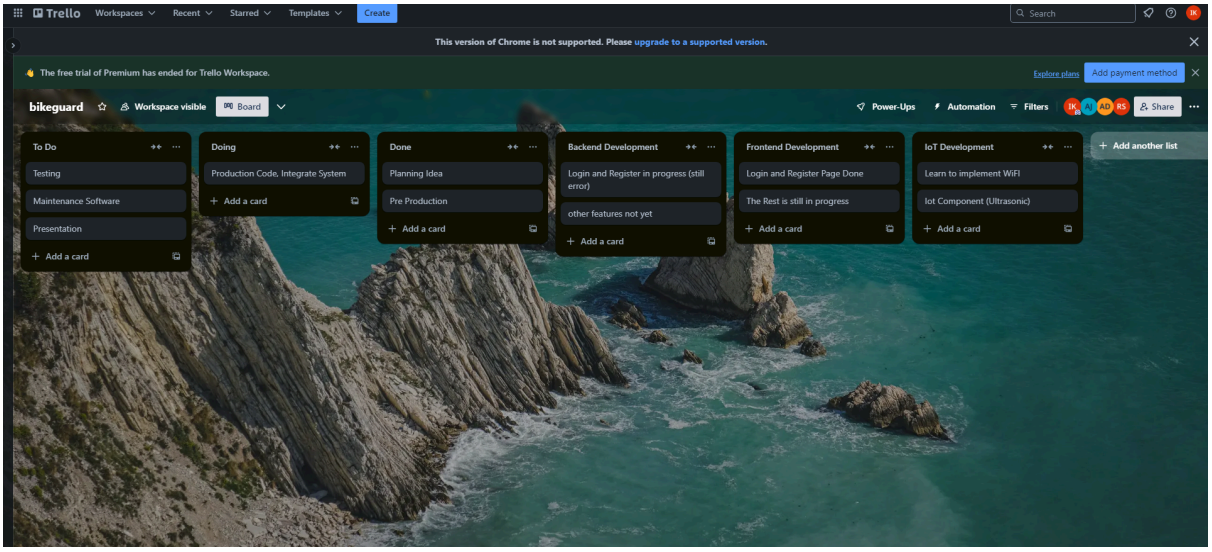
QR codes, and parking slot statuses

- Integration Testing
  Focusing on integrating between backend or BikeGuard features with IoT devices including the successful communication between IoT sensor and backend, Data consistency for website interface, and real-time status updates
- System Testing
  Focusing on testing end-to-end the BikeGuard System including Scenario tests for all environments from the backend, frontend, and IoT
- Defect Management
  Tracking and Resolve identified defects on BikeGuard Including From the Backend, Frontend, IoT, and Database Environment during the testing process
- User Acceptance Testing
  Engaging end-users in UAT such as Students and Staff Universitas Indonesia to validate the system usability, the process is to test common workflow such as parking bikes, and get feedback from end-users regarding the performance and responsiveness
- Performance Tuning
  Ensuring BikeGuard can operate efficiently under load conditions including the Asses system when handling multiple users at the same time and determining system in an extreme condition such as high data traffic in IoT and user interaction
- Documentation
  Maintain detailed records for all testing activities, including testing plan, testing case and scenario, testing defect logs and resolution report, testing process, and testing result

5.6. Resource & Environment Needs

Testing Tools

A. Trello
   Trello is a versatile project management tool designed to help the team organize and prioritize tasks using boards, lists, and cards. Trello has an intuitive and flexible visual, making it user-friendly for new users to adapt to various project management styles. Trello facilitates tracking testing progress, structuring test cases, and collaborating with the team to identify and resolve issues in the project.



Picture 17 Trello

B. Postman

Postman is a collaborative API development platform that streamlines the creation, testing, and documentation of APIs. Postman has a user-friendly interface that enables the sending of HTTP requests and the analysis of server responses, Postman is used to test NodeJS backend API endpoints for our BikeGuard backend systems. It allows the tester to craft and execute the API request, verify responses, and ensure seamless integration between application components.

Picture 18 Postman

C. Wokwi

Wokwi is an online simulation platform designed for building and testing IoT projects. It has a virtual environment that allows the team to develop virtual hardware components such as microcontrollers, sensors, and other components, Wokwi supports many programming languages such as C, C++, Assembly, and Python and it has all common microcontrollers including ESP32 and Arduino.

Wowki focuses on the developers to create prototypes, debugging, and testing the IoT real-time system. In BikeGuard we use Wokwi test and make an IoT system before we deploy the system in real life, so it would be easier to find problems or bug during the implementation coding life cycle

Picture 19 Wokwi

CHAPTER 6
Test Result

6.1. Testing Result

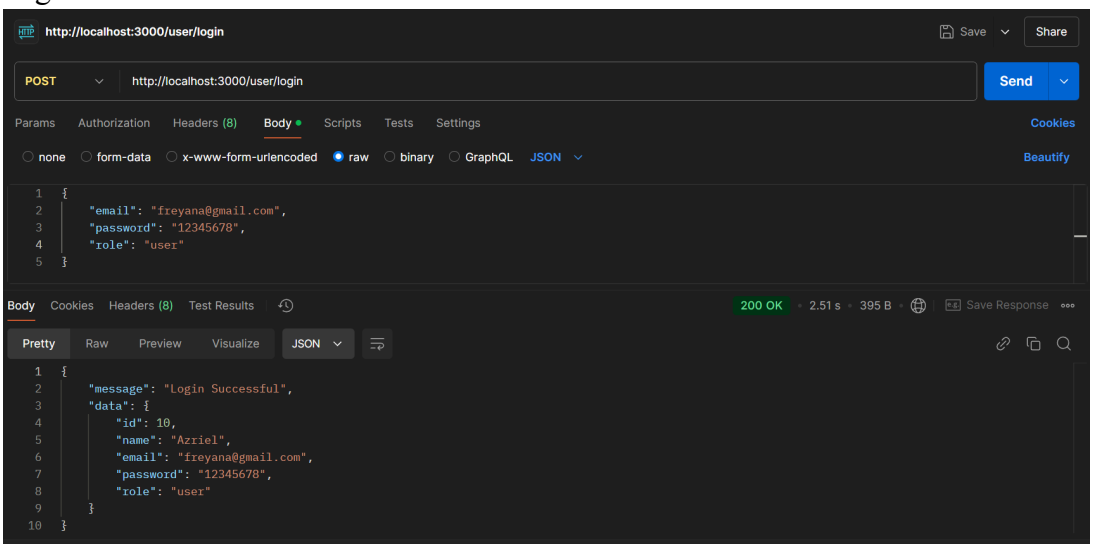
No	Feature	Test Case	Input Case	Expected Output Case	PASS/FAIL
1	Register	Check If the system success in creating an account	Username, Email, Password, and Role	Users successfully Create an account based on their input	PASS
2	Login	Check If the system success in logging in	Username, Password, and Role	User successfully login into their account if the password and username correct	PASS
3	QR Scan	Check if the system success in reading QRCode	QR Code	The system can read the QR code	PASS
4	Check Parking	Check if the system is successful in checking the parking status of the user account	Account Status	The system can find out the parking status of the user's account	PASS

5	Make Parking Slot	Check if the system is successful in making a slot parking	Admin Activity	The system can make a new parking slot for the user	PASS
6	Detection Parking System	Check if the IoT system is successfully detected any bikes around the parking space	IoT Sensor	The system can determine if the parking slot is available based on data read by an ultrasonic sensor	PASS
7	Locking System	Check if the system is successful in locking and unlocking the system	User Activity on the website	The system can lock and unlock the bike locker based on the interaction between the user and the website	PASS
8	Integrating System	Check If the IoT component can send the data to the server correctly	Check Data Availability	The website can get the data from IoT devices and sensor	PASS

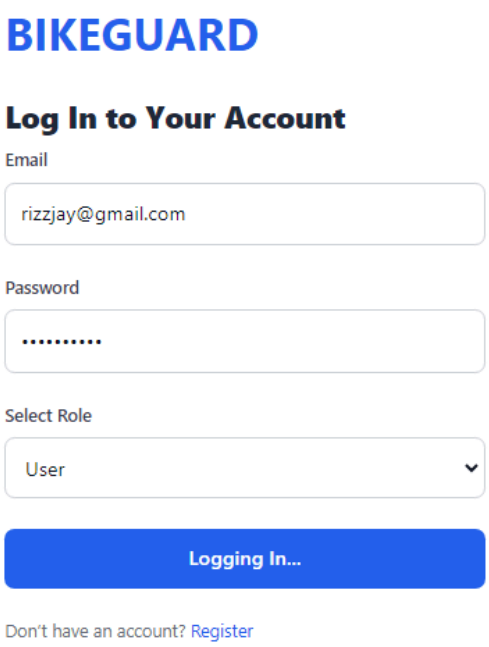
Table 2 Testing Result

6.2. Website Testing

- Login User = Success

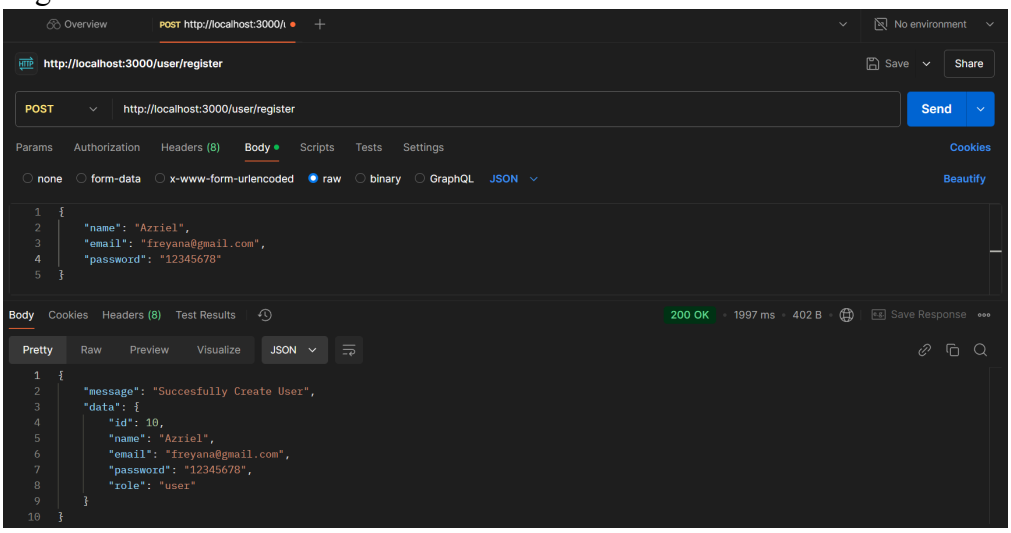


Picture 20 Postman Login User

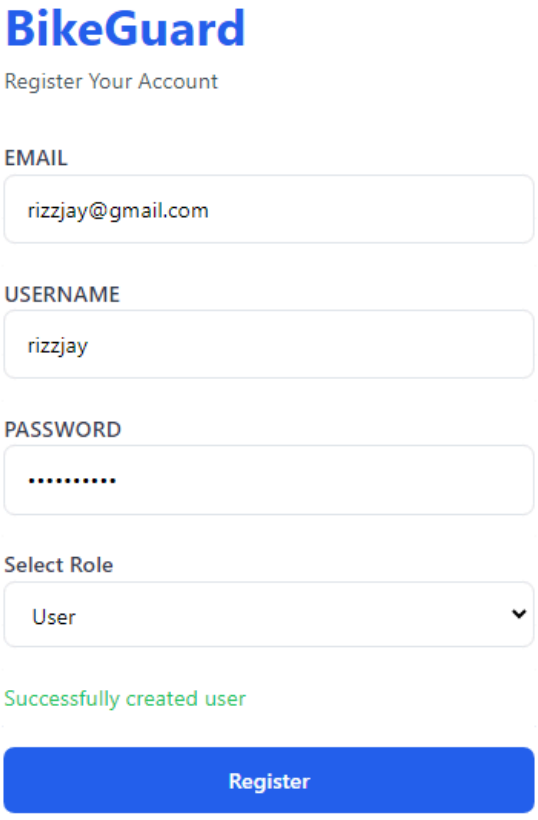


Picture 21 Website Login User

- Register User = Success

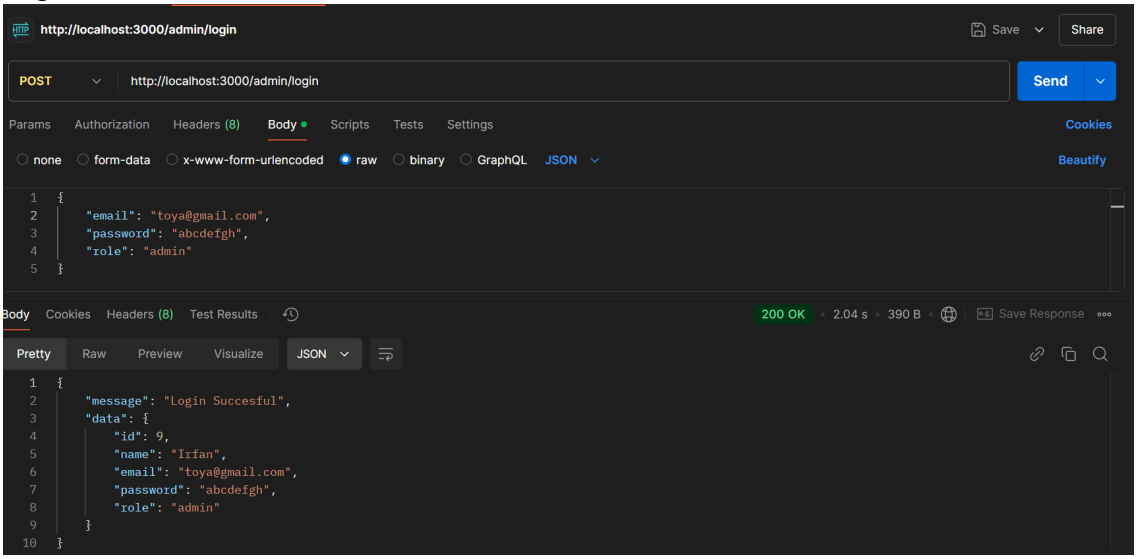


Picture 22 Postman Register User

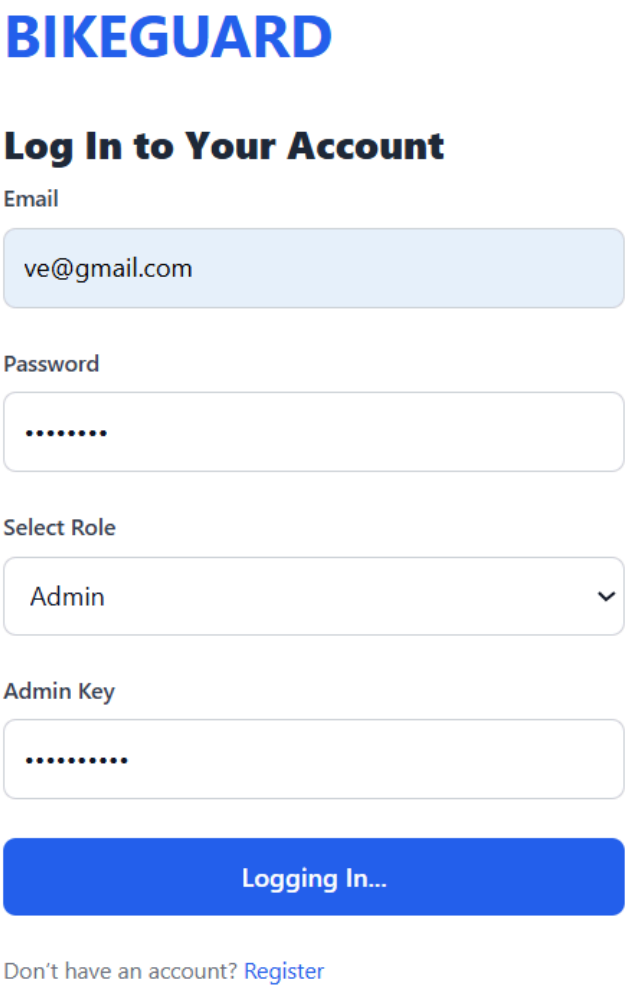


Picture 23 Website Register User

- Login Admin : Success

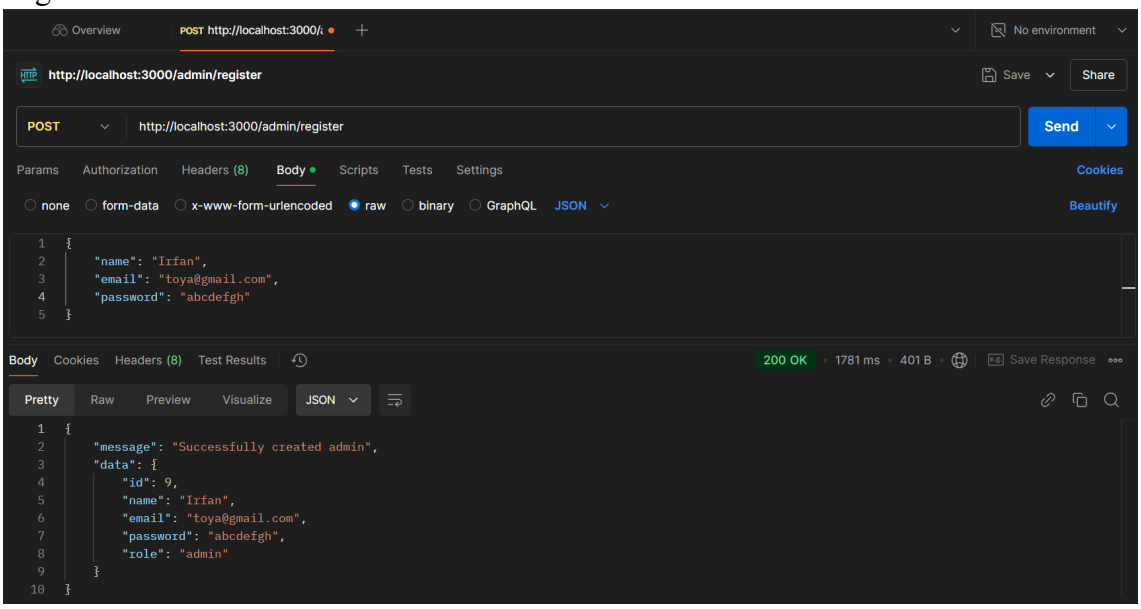


Picture 24 Postman Admin Login

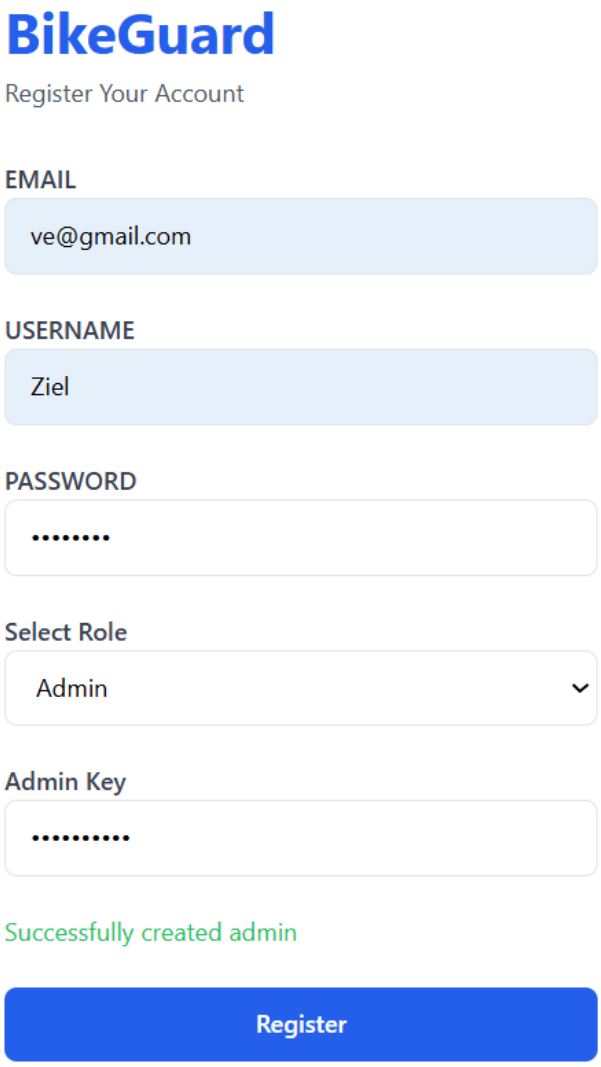


Picture 25 Website Admin Login

- Register Admin: Success

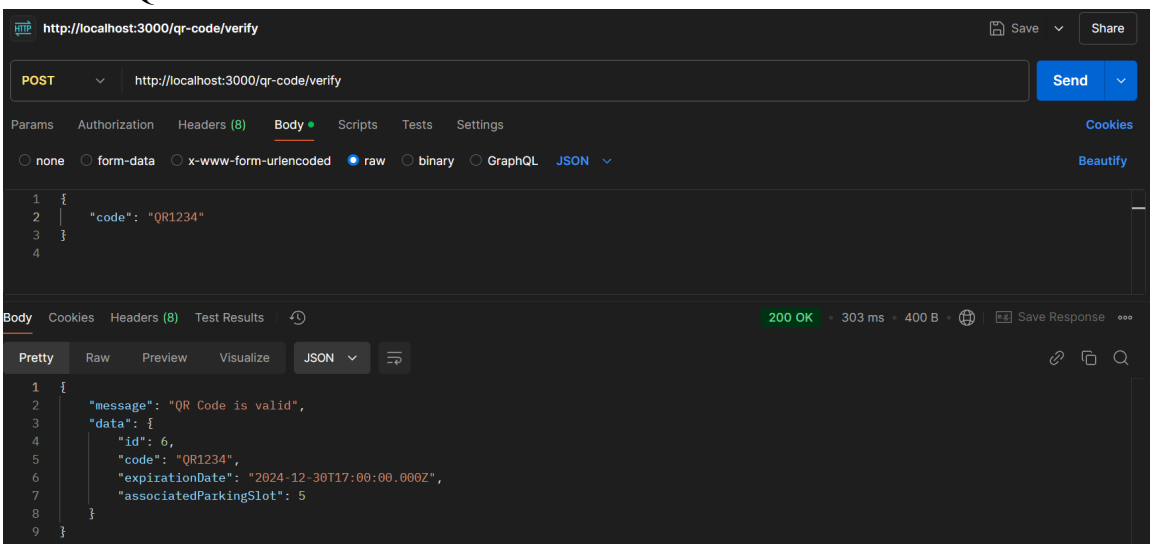


Picture 26 Postman Admin Register

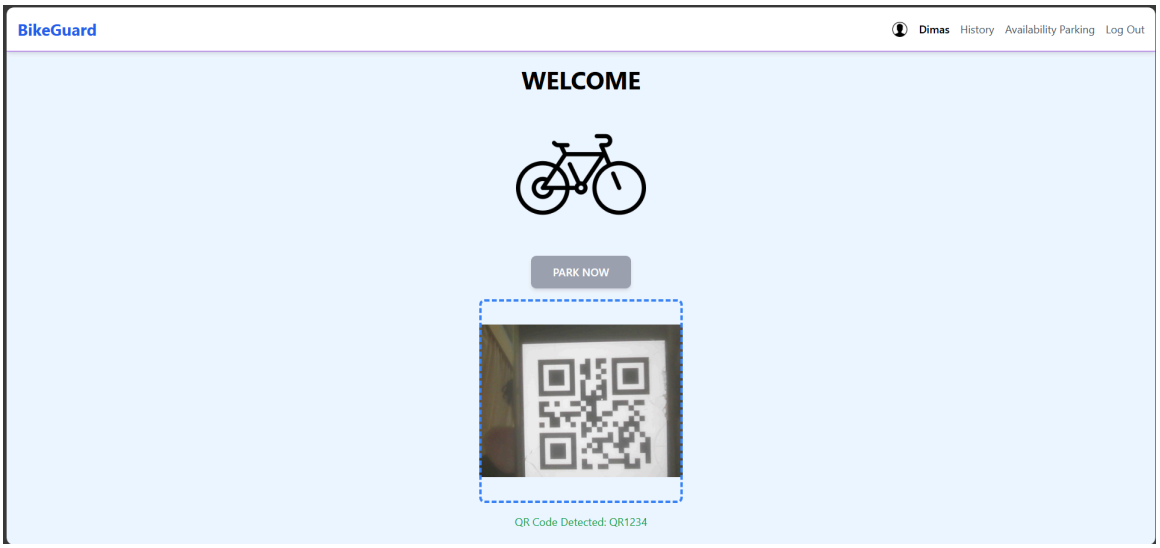


Picture 27 Website Admin Register

- Verifikasi QR Code: Success

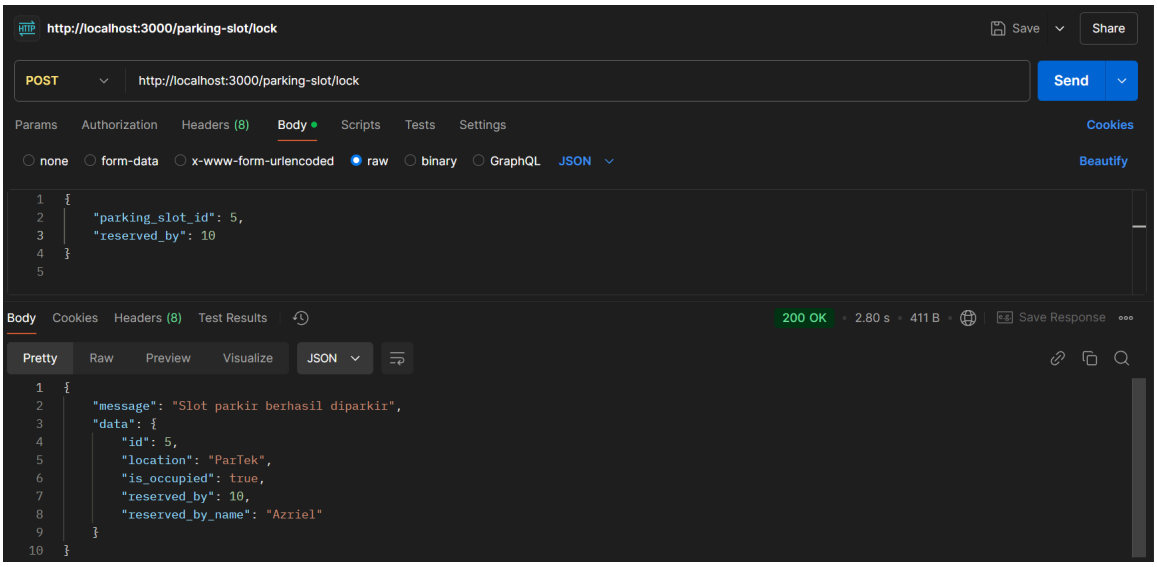


Picture 28 Postman QR Code

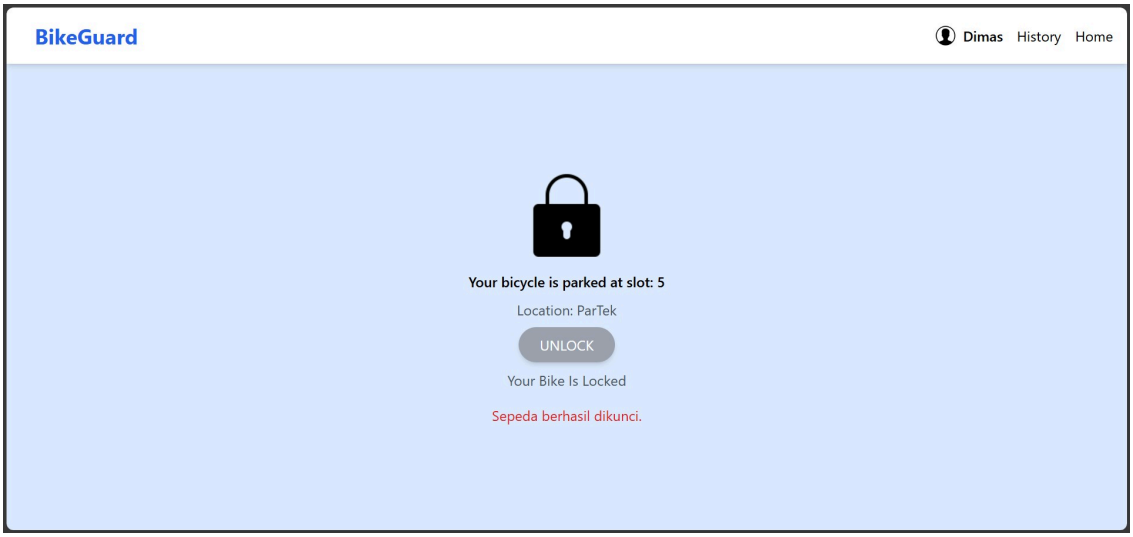


Picture 29 Website QR Code

- Lock Bike: Success

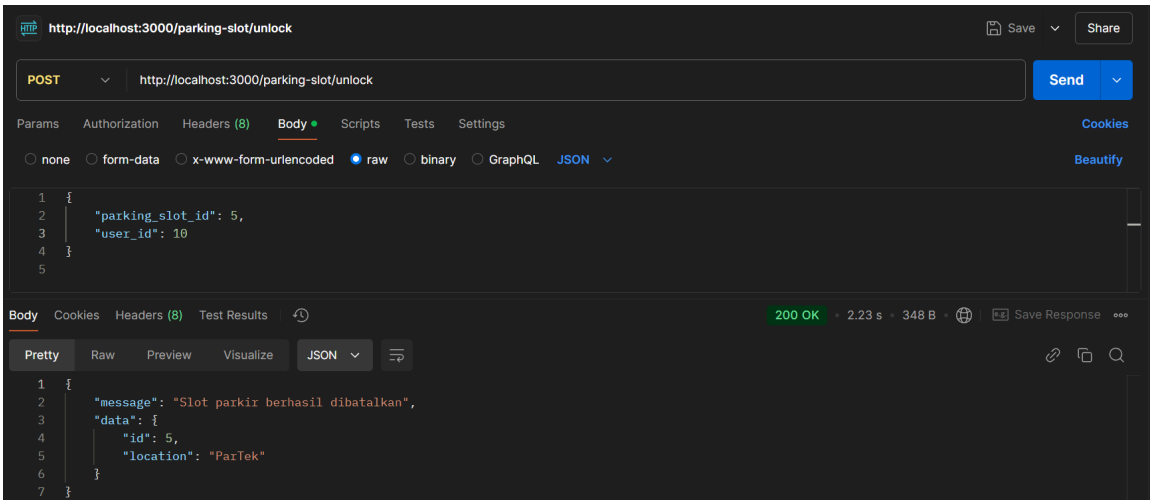


Picture 30 Postman Lock Bike

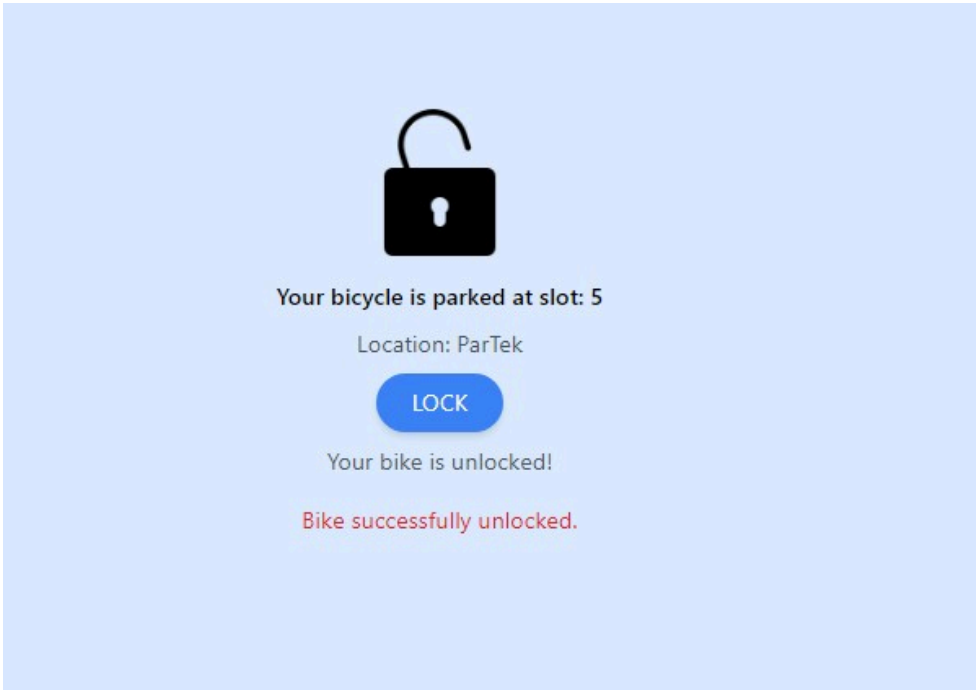


Picture 31 Website Lock Bike

- Unlock Bike:

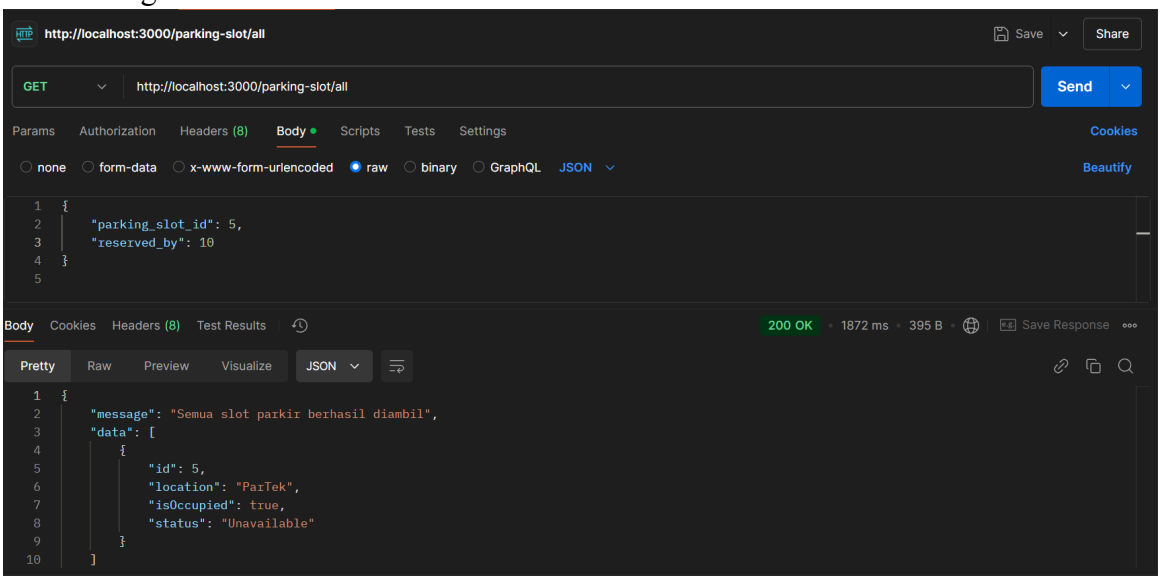


Picture 32 Postman Unlock Bike

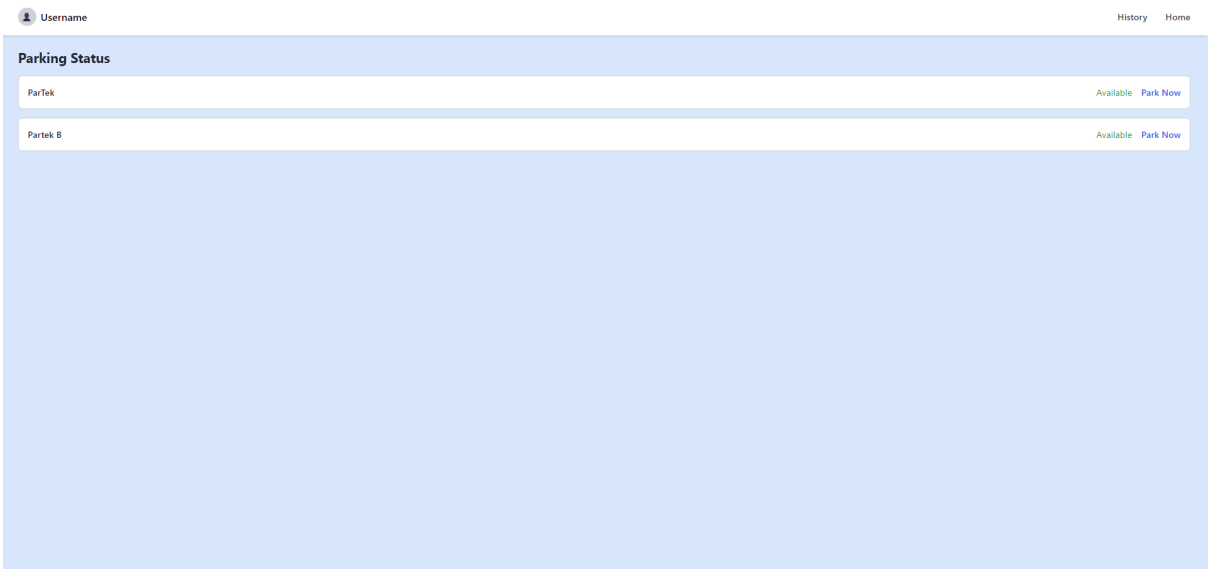


Picture 33 Website Unlock Bike

- List Parking Slot:



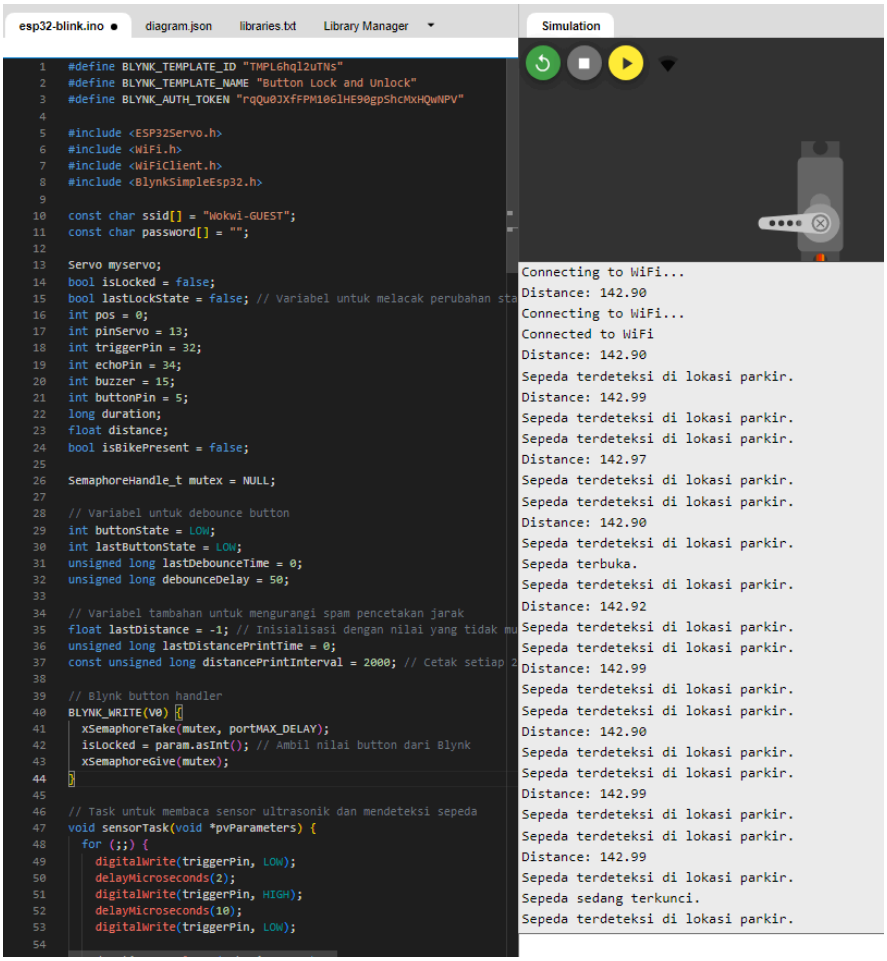
Picture 34 Postman List Parking



Picture 35 Website List Parking

6.3. IoT Testing (Virtual Testing)

- Ultrasonic Sensor = Success
- Button Locking and Unlocking = Success
- Integrating with Backend and Frontend = Success



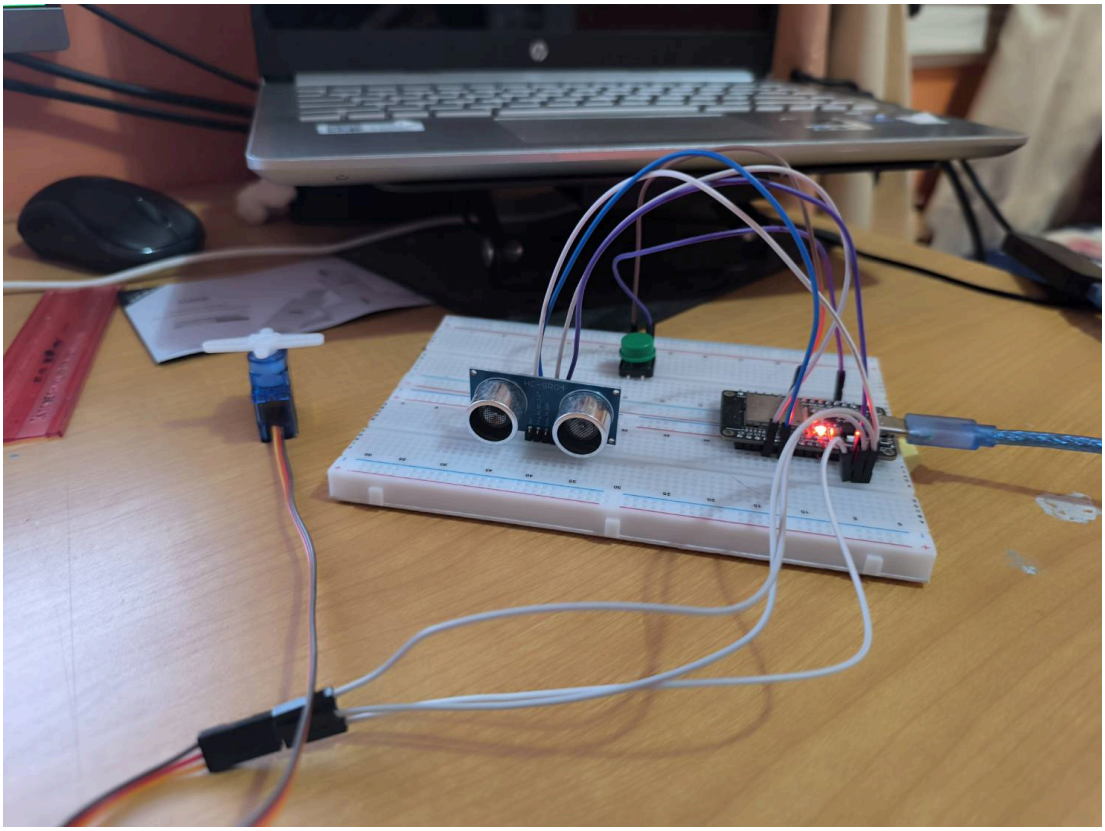
esp32-blink.ino ● | diagram.json | libraries.txt | Library Manager

```
#define BLYNK_TEMPLATE_ID "TMPL6hql2uTNs"
#define BLYNK_TEMPLATE_NAME "Button Lock and Unlock"
#define BLYNK_AUTH_TOKEN "rqQu0JXfFPM106lHE90gpShcMxHQwNPV"

#include <ESP32Servo.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

const char ssid[] = "Wokwi-GUEST";
const char password[] = "";

Servo myservo;
bool isLocked = false;
bool lastLockState = false; // Variabel untuk melacak perubahan sta
int pos = 0;
int pinServo = 13;
int triggerPin = 32;
int echoPin = 34;
int buzzer = 15;
int buttonPin = 5;
long duration;
float distance;
bool isBikePresent = false;

SemaphoreHandle_t mutex = NULL;

// Variabel untuk debounce button
int buttonState = LOW;
int lastButtonState = LOW;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

// Variabel tambahan untuk mengurangi spam pencetakan jarak
float lastDistance = -1; // Inisialisasi dengan nilai yang tidak mu
unsigned long lastDistancePrintTime = 0;
const unsigned long distancePrintInterval = 2000; // Cetak setiap 2

// Blynk button handler
BLYNK_WRITE(V0) {
  xSemaphoreTake(mutex, portMAX_DELAY);
  isLocked = param.asInt(); // Ambil nilai button dari Blynk
  xSemaphoreGive(mutex);
}

// Task untuk membaca sensor ultrasonik dan mendeteksi sepeda
void sensorTask(void *pvParameters) {
  for (;;) {
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);
```

Simulation

```
Connecting to WiFi...
Distance: 142.90
Connecting to WiFi...
Connected to WiFi
Distance: 142.90
Sepeda terdeteksi di lokasi parkir.
Distance: 142.99
Sepeda terdeteksi di lokasi parkir.
Sepeda terdeteksi di lokasi parkir.
Distance: 142.97
Sepeda terdeteksi di lokasi parkir.
Sepeda terdeteksi di lokasi parkir.
Distance: 142.90
Sepeda terdeteksi di lokasi parkir.
Sepeda terbuka.
Sepeda terdeteksi di lokasi parkir.
Distance: 142.92
Sepeda terdeteksi di lokasi parkir.
Sepeda terdeteksi di lokasi parkir.
Distance: 142.99
Sepeda terdeteksi di lokasi parkir.
Sepeda terdeteksi di lokasi parkir.
Distance: 142.90
Sepeda terdeteksi di lokasi parkir.
Sepeda terdeteksi di lokasi parkir.
Distance: 142.99
Sepeda terdeteksi di lokasi parkir.
Sepeda terdeteksi di lokasi parkir.
Distance: 142.99
Sepeda terdeteksi di lokasi parkir.
Sepeda sedang terkunci.
Sepeda terdeteksi di lokasi parkir.
```

Picture 36 IoT Wokwi Testing



Picture 37 IoT Hardware Testing

6.4. User Acceptance Testing

User acceptance testing is a critical phase where the early version of the BikeGuard System is provided to users for evaluation. A few chosen users will have the opportunity to explore the BikeGuard system without any supervision from the developer and provide valuable feedback that can help the developer improve the BikeGuard system. The feedback will be collected through a few questions in a survey.

Survey Questions

Common Question

1. How easy was the web application for you to understand and use BikeGuard?
    a. Very Easy
    b. Easy
    c. Neutral
    d. Difficult

    e. Very Difficult
2. How would you rate the system's overall experience?
    a. Very Good
    b. Good
    c. Neutral
    d. Bad
    e. Very Bad

Feature Question

3. Does the QR Scanning work properly and quickly?
    a. Works well and fast
    b. Works well but slow
    c. Works but inconsistently
    d. Works but often fails
    e. Doesn't work at all
4. Does the Locking system fulfill and satisfy your expectations?
    a. Very Satisfied
    b. Satisfied
    c. Neutral
    d. Dissatisfied
    e. Very Dissatisfied
5. How often do you experience feature failures when using the BikeGuard system?
    a. Never
    b. Rarely
    c. Occasionally
    d. Frequently
    e. Very Frequently

Performance Question

6. Is the system response time adequate for operations like Login, Register, QR scanning, and other activities?
    a. Very Fast
    b. Fast
    c. Neutral
    d. Slow
    e. Very Slow
7. Does the system remain stable even when multiple users are accessing BikeGuard at the same time?
    a. Very Stable
    b. Stable
    c. Neutral

d. Unstable
e. Very Unstable

Additional Feedback

8. Please provide any comments or suggestions for improvement:

   [Free-text response]

Tester	Overall Score	Feedback
Aulia Anugrah Aziz	4.28/5	The app is user-friendly, QR code feature is efficient.
Christopher Satya Fredella Balakosa	4.42/5	Great design with user-friendly interface
Christopher Sutandar	4/5	Simple and intuitive, but the app occasionally lags when using locking system.
Daniel Niko Mardjaja	4/5	Good usability, but the availability of parking slots could be displayed more clearly.
Darren Nathanael Boentara	4/5	Smooth experience overall, but navigating page could be faster and more reliable.
Edgrant Henderson Suryajaya	4.28/5	Efficient features, but the app's loading time should be improved.
Farhan Nuzul Noufendri	4.14/5	Good functionality, but response times of QR code scan need to be more consistent.
Louis Benedict Archie	4.28/5	Excellent QR code system, but navigation could be more fluid.
Mario Matthews Gunawan	4/5	Convenient to use, but needs better visualization for parking slots.
Rizqi Zaidan	4.28/5	Well-designed app with helpful features, but minor

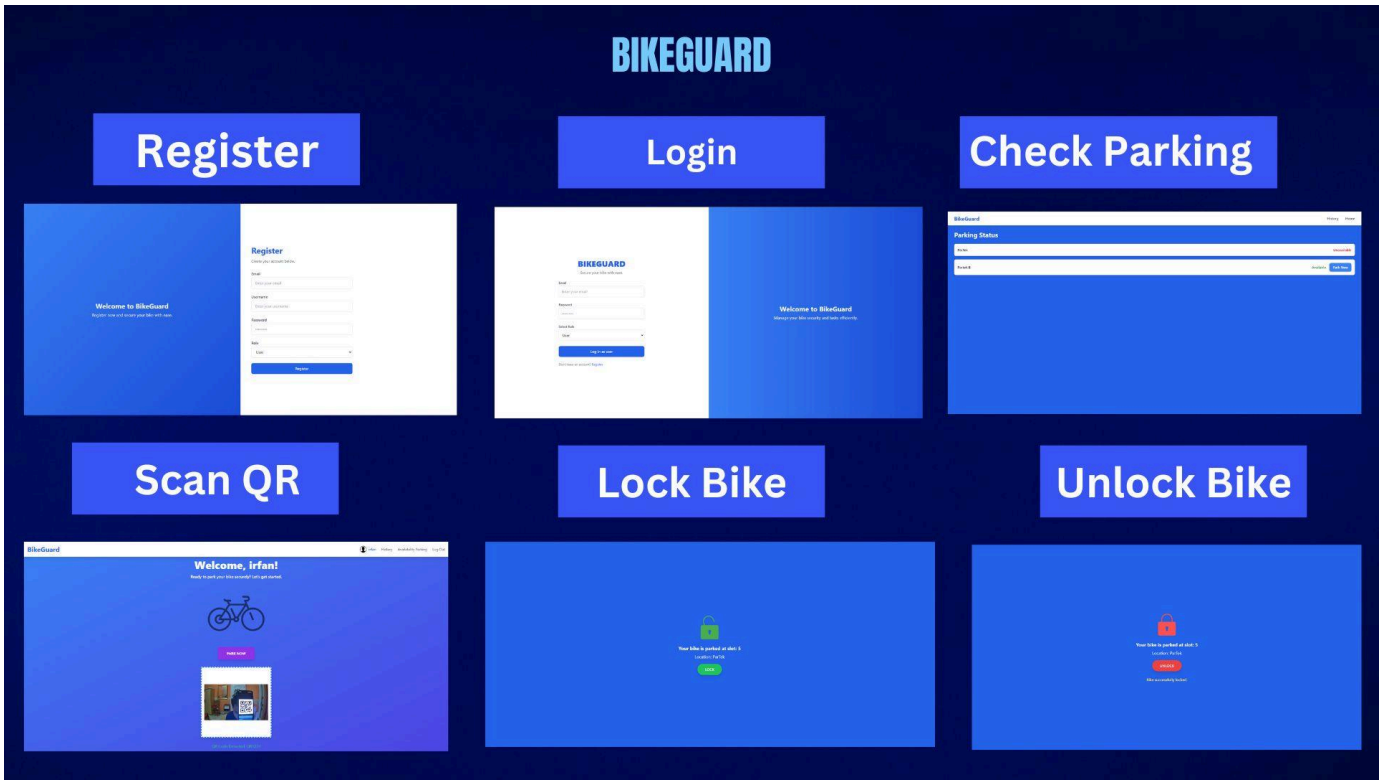
		performance issues need attention.

Table 3 System Score and Feedback

6.5. Client Testing

Picture 38 Testing Product By Client

CHAPTER 7
User Manual



Picture 39 User Manual

CHAPTER 8
MoU

Surat Perjanjian Kerjasama Pembuatan dan Pengembangan "BikeGuard" yang selanjutnya disebut "Perjanjian" dibuat dan ditandatangani pada hari Minggu, 6 Oktober 2024 yang bertempat di mana oleh dan antara:

Nama Anggota : Andrew Kristofer Jian

Tempat, Tanggal Lahir: Cimahi, 9 Juli 2004

Alamat : Jalan R. Moch. Saleh No. 25 Sadamalun I RT 03/RW 07, Kel. Nagasari, Kec. Karawang Barat, Karawang 41312

No. Telp : +62 878-8182-0317

Nama Anggota : Azriel Dimas Ash-Shidiqi

Tempat, Tanggal Lahir: Bandung, 5 Februari 2003

Alamat : Jl. Dr. GSSJ Ratulangi, No. 30. Kec. Menteng, Kota Jakarta Pusat

No. Telp : +62 821-1668-5115

Nama Anggota : Irfan Yusuf Khaerullah

Tempat, Tanggal Lahir: Depok, 25 Juni 2004

Alamat : Perumahan Legenda Wisata Zona Acropolis JL Acropolis Utara Blok C3 NO 3

No. Telp : +62 821-1465-1694

Dalam hal ini, bertindak sebagai *Developer Team* dari perangkat lunak "BikeGuard" dan selanjutnya disebut dengan "Pihak Pertama".

Nama *Client* : Rizqi Zaidan

Tempat, Tanggal Lahir: Jakarta, 29 Desember 2003

Alamat : Rumah Kost Amora Kukusan, Kecamatan Beji, Kota Depok, Jawa Barat 16425

No. Telp : +62 817-7927-8925

Dalam hal ini, bertindak sebagai *Client* dan selanjutnya disebut dengan “Pihak Kedua”.

Dengan ini, Pihak Pertama dan Pihak Kedua telah mencapai kesepakatan untuk mengadakan kerja sama dengan ketentuan ketentuan berikut.

PASAL 1
RUANG LINGKUP KERJA SAMA

1. Pihak Pertama setuju untuk mengembangkan perangkat lunak “BikeGuard” yang akan digunakan sebagai sistem keamanan parkir sepeda di Universitas Indonesia.

2. Pihak Pertama menunjuk Pihak kedua untuk melakukan pengujian terhadap perangkat lunak “BikeGuard”.

3. Pihak Kedua setuju untuk melakukan pengujian sesuai dengan prosedur yang disepakati oleh kedua belah pihak.

4. Pihak Kedua setuju untuk memberikan laporan evaluasi perkembangan “BikeGuard” kepada pihak pertama.

PASAL 2
JANGKA WAKTU

1. Kerja sama ini akan dimulai pada tanggal 6 oktober 2024 dan berakhir pada tanggal 9 Desember 2024.

2. Pihak Pertama dan Pihak Kedua sepakat setiap perubahan jadwal harus disepakati secara bersama secara tertulis antara Pihak Pertama dan Pihak Kedua.

PASAL 3
TANGGUNG JAWAB

1. Pihak Pertama bertanggung jawab untuk menyelesaikan perangkat lunak “BikeGuard” sesuai dengan spesifikasi dan waktu yang telah disepakati.

2. Pihak Kedua bertanggung jawab untuk menyediakan sumber daya yang dibutuhkan dalam pengembangan perangkat lunak “BikeGuard”.

PASAL 4
PEMBAYARAN DAN BIAYA

1. Pihak Kedua setuju untuk membayar pihak pertama sebesar Rp 300.000,- untuk pengembangan perangkat lunak “BikeGuard”.

PASAL 5
KOMPENSASI

1. Pihak Pertama Setuju untuk membayar Pihak kedua sejumlah Rp 100.000,- sebagai imbalan untuk melakukan pengujian terhadap perangkat lunak “BikeGuard” di akhir kerja sama.

2. Apabila Pihak Kedua gala menyelesaikan pengujian atau tidak memberikan laporan evaluasi dalam waktu yang ditentukan, Pihak Pertama berhak untuk membatalkan atau menunda kompensasi.

PASAL 6
KEPEMILIKAN

1. Hak atas kekayaan intelektual perangkat lunak “BikeGuard” sepenuhnya akan menjadi milik Pihak Pertama, termasuk *source code*, desain dan hal hal yang berkaitan dengan pengembangan perangkat lunak.

2. Pihak Kedua tidak memiliki hak untuk menjual atau mendistribusikan perangkat lunak “BikeGuard” tanpa persetujuan dari Pihak Pertama.

PASAL 7
KERAHASIAAN

1. Pihak Kedua setuju untuk menjaga kerahasiaan informasi dari perkembangan perangkat lunak “BikeGuard” selama kerjasama dan setelah berakhirnya kerja sama.

2. Pihak Kedua dilarang menyebarkan informasi rahasia untuk kepentingan pribadi tanpa persetujuan dari pihak pertama.

PASAL 8
PEMUTUSAN PERJANJIAN

1. Kedua belah pihak setuju bahwa perjanjian dapat diakhiri oleh salah satu pihak apabila selama proses perkembangan perangkat lunak “BikeGuard” terjadi pelanggaran dalam perjanjian kerjasama ini.

PASAL 9
PENYELESAIAN SENGKETA

1. Apabila terjadi perselisihan antara kedua belah pihak terkait dengan pelaksanaan perjanjian ini, maka akan diselesaikan secara musyawarah untuk mufakat.

PASAL 10
PEMELIHARAAN DAN DUKUNGAN

1. Pihak Pertama sepakat untuk memberikan dukungan teknis dan pemeliharaan perangkat lunak “BikeGuard” selama 90 hari setelah penyerahan akhir perangkat lunak kepada Pihak Kedua.

2. Dukungan teknis meliputi perbaikan bug, pembaruan kecil, serta bantuan untuk penggunaan perangkat lunak.

3. Jika ada pengembangan fitur tambahan atau pembaruan besar, Pihak Kedua setuju bahwa layanan tersebut akan dikenakan biaya terpisah yang akan dibicarakan kemudian.

PASAL 11
LAIN-LAIN

1. Perjanjian ini mengikat kedua belah pihak sejak ditandatangani dan tidak dapat diubah kecuali dengan persetujuan tertulis dari kedua belah pihak.

2. Segala sesuatu yang belum diatur dalam perjanjian ini akan disepakati dan ditetapkan kemudian oleh kedua belah pihak secara tertulis sebagai bagian yang tidak terpisahkan dari perjanjian ini.

3. Perjanjian ini dibuat dalam rangkap 2 (dua), masing-masing memiliki kekuatan yang sama, dan diberikan kepada Pihak Pertama dan Pihak Kedua.

LEMBAR PENGESAHAN

Surat Perjanjian Kerjasama telah dibaca, dipahami dan disepakati oleh kedua belah pihak pada hari ini dan tanggal tersebut pada surat perjanjian kerja sama ini.

Pihak Pertama

Andrew Kristofer Jian

Azriel Dimas Ash-Shidiqi

Irfan Yusuf Khaerullah

Pihak Kedua

Rizqi Zaidan