
UNIVERSITAS
INDONESIA
TRASH
TRACKER
Group 5
Annisa Ardelia Setiawan
Mario Matthews Gunawan
Rizqi Zaidan
Recycle

PREFACE

In today's rapidly growing urban world, waste management has become a major challenge. Improper disposal and a lack of understanding about separating waste types continue to harm our environment, making it clear that new and innovative solutions are needed to encourage better habits and sustainable practices. Waste is generated everywhere—from homes to public spaces—but awareness of its impact on the environment remains limited.

This paper introduces Trash Tracker: Smart Waste Monitoring and Management System, a project developed by Group 5 as part of the Rekayasa Perangkat Lunak course. Through this project, we aimed to design a system that not only helps users manage waste more efficiently but also raises awareness about the importance of proper waste sorting. Using IoT technology, the system automates the process of separating waste into wet and dry categories, tracks trash bin levels in real-time, and provides notifications and insights to users via a web-based platform.

The journey of developing Trash Tracker has been both challenging and rewarding. As a team, we delved deep into the principles of IoT, explored practical applications of technology, and collaborated to solve complex problems. Each stage of the project—from brainstorming and designing the architecture to implementing the system and testing its performance—offered valuable learning experiences. It also reinforced the importance of teamwork, attention to detail, and the ability to adapt to challenges.

By addressing waste management issues with a simple yet effective solution, Trash Tracker demonstrates how technology can directly contribute to solving environmental problems. Beyond just a tool, it serves as an initiative to inspire behavioral changes and foster a greater sense of responsibility among users in managing their waste.

As members of Group 5, we are proud to present this project as part of our contribution to advancing sustainable practices. We hope that Trash Tracker inspires others to think creatively about how technology can address pressing global challenges, particularly those related to environmental sustainability. This project has not only strengthened our technical and problem-solving skills but also underscored the role of innovation in shaping a better and cleaner future.

Depok, December 19, 2024

Group 5

Table of Contents

List of Figures

List of Tables

CHAPTER I
INTRODUCTION

1.1 Background

Trash is an integral yet often overlooked byproduct of human life, present in every home, workplace, and public space. Whether through eating, cleaning, or producing goods, waste is generated at every step. The types of waste range from biodegradable materials to non-biodegradable plastics and metals, with wet and dry categories being the most prevalent. Wet waste includes organic materials like food scraps and yard trimmings, while dry waste consists of materials such as paper, glass, and plastic. Despite the clear distinctions, many people remain unaware of the differences between these types of waste, and consequently, improper waste disposal and management become persistent issues.

In addition to the problem of waste classification, another pressing concern is the sheer volume of trash produced on a daily basis. With urban populations increasing, the daily waste output of individuals and communities has skyrocketed. Unfortunately, people often remain unconscious of just how much waste they produce and how this accumulation impacts the environment. Excess waste that is not properly managed leads to significant environmental pollution, contributing to the degradation of natural ecosystems, soil contamination, and water pollution, and even impacting public health.

Given the scale of these issues, there is a growing need for innovative solutions that help manage waste more effectively, while simultaneously educating the public. In today's world, where technology plays a central role in almost every sector, it's possible to incorporate practical solutions for addressing waste management challenges. Technology has improved the efficiency and productivity of many processes, making once time-consuming tasks easier and faster. As a result, society has increasingly embraced technological solutions, particularly those that offer convenience, simplicity, and efficiency.

Trash Tracker seeks to take advantage of these technological trends by providing a smart solution to waste management. The project aims to raise awareness among users about the waste they generate and to make waste sorting easier and more efficient. By automating the process of sorting trash into wet and dry categories and tracking the volume of waste, the system encourages individuals to become more mindful of their waste habits. As a result, it reduces the burden of manual sorting and promotes better waste disposal practices.

The Trash Tracker system not only aims to inform users when the bin is full or when waste needs to be sorted, but also to serve as an educational tool that fosters a greater understanding of environmental impact. By making waste management more practical and accessible, it aims to contribute to a cleaner and more organized community where waste is properly managed, sorted, and reduced. In doing so, it addresses both environmental and societal challenges, and helps promote a more sustainable approach to everyday waste disposal. Ultimately, the goal is to leverage modern technology to inspire a shift toward greater environmental responsibility and public awareness, ensuring that waste, rather than being a problem, becomes a resource that is properly managed.

1.2 Literature Review

Recent studies and innovations in the field of waste management emphasize the increasing role of automation and smart technology in addressing the global waste crisis. As urban populations grow and waste production increases, traditional waste management systems often struggle to keep pace, leading to improper waste disposal, increased landfill use, and environmental degradation. In response, researchers and developers have begun exploring smart waste management systems that leverage automation and Internet of Things (IoT) technologies to make waste collection, sorting, and disposal more efficient and sustainable.

One of the most promising advancements is the development of IoT-based smart bins, which automate many of the processes involved in waste management. These smart bins typically integrate sensors, data processing units, and communication modules to automatically sort waste into appropriate categories, track the volume of waste, and notify users or waste management services when the bin is full or requires attention. The automation of waste sorting helps to reduce human error and increases the efficiency of waste disposal processes, ensuring that recyclable materials are separated from non-recyclable waste and reducing the contamination of waste streams.

A number of sensor technologies have been employed in smart waste management systems to improve waste classification. For instance, moisture sensors have been used to differentiate between wet and dry waste, which is critical for composting and recycling processes. Wet waste, such as food scraps, can be composted, while dry waste, including materials like paper and plastic, is better suited for recycling or incineration. Ultrasonic sensors are often utilized to measure the fullness of trash bins by calculating the distance between the sensor and the trash inside the bin. This helps in determining when the bin needs to be emptied, preventing overflow and ensuring timely waste collection.

Beyond basic sensors, advanced technologies like AI-based image recognition have been researched to further enhance waste classification. AI algorithms can analyze images of the waste to identify specific types of materials (e.g., plastics, metals, or organic waste) and sort them accordingly. These systems hold great promise for industrial-scale waste management, where the variety and volume of waste can be overwhelming for traditional sorting methods. Although AI-powered waste sorting systems are efficient and precise, they are still largely in the research and development phase and often come with significant costs, making them less accessible for smaller-scale applications or everyday household use.

Several studies have emphasized the environmental benefits of smart waste segregation systems, particularly in terms of reducing landfill pollution. When waste is improperly sorted, recyclable materials often end up in landfills, where they contribute to environmental harm. Accurate classification of wet and dry waste not only minimizes contamination in recycling streams but also helps reduce the overall volume of waste that ends up in landfills. Wet waste, in particular, is a major contributor to methane emissions when decomposing in landfills, which has a significant impact on climate change. By automating the process of waste sorting, smart systems can help mitigate these environmental impacts.

Despite these advancements, many current solutions face challenges in terms of cost-effectiveness and accessibility. High-tech systems like AI-based sorting are often too expensive for widespread adoption, particularly in developing regions where waste management issues are most acute. Additionally, many smart waste management solutions lack real-time feedback mechanisms that notify users about the state of their waste bins or provide insights into their waste generation patterns. Without these features, users may not fully engage with the system or develop the waste-conscious behaviors needed for long-term sustainability.

Our project, Trash Tracker, aims to address these gaps by offering a cost-effective, sensor-based solution that is accessible to both households and small businesses. By focusing on essential features such as wet and dry waste separation using moisture sensors and real-time bin fullness detection with ultrasonic sensors, we provide a practical and affordable solution to waste management. In addition, our system includes a user-friendly web interface that allows users to monitor their waste in real-time, receive notifications when their bin is full, and access data on their waste generation trends. This real-time feedback encourages more conscious waste management and helps users stay engaged with the system over time.

1.3 Project's objective

The primary objective of this project is to increase public awareness regarding the waste they generate on a daily basis, specifically highlighting the differences between various types of waste, such as wet and dry waste, and the sheer volume that individuals and households produce. By fostering a deeper understanding of waste generation, the project aims to encourage more responsible waste disposal practices and promote the importance of sorting waste at the source. This will contribute to reducing the overall volume of waste that ends up in landfills, which is a significant factor in environmental pollution.

In addition to raising awareness, the project also seeks to improve waste management practices by introducing technology that simplifies the process of waste tracking and evaluation. The use of practical, user-friendly technology will empower individuals and communities to monitor their waste output more efficiently, making it easier to adopt sustainable habits. By providing real-time data on the types and amounts of waste generated, the project encourages proactive decision-making that supports recycling and composting efforts. Ultimately, this initiative aims to make waste management more accessible and integrated into everyday life, helping to foster a more environmentally conscious society and contributing to the global goal of reducing waste and environmental impact.

1.4 Similar Competitor

For competitors in the UI (University of Indonesia) area, particularly within the scope of waste management, there are currently no direct solutions that focus on monitoring and managing waste. No comprehensive waste monitoring system is deployed around UI, including within the UI engineering faculty, which presents a unique opportunity for the project. Despite the growing concern for sustainable practices and environmental management, there is a gap in technological solutions that address the sorting, tracking, and evaluation of waste in this region.

However, on a broader scale, one notable solution in the market is SmartBin, an IoT-based waste management system. SmartBin is designed to automate the waste collection and sorting process, providing a smart and efficient way to manage waste in urban areas and large facilities. The system utilizes advanced sensors, such as ultrasonic sensors to monitor the capacity of bins, detecting when they are nearly full, and humidity sensors to help classify waste based on moisture levels. SmartBin's integration of these sensors is similar to the technological approach in our project, where we also employ sensors to distinguish between different types of waste and track bin capacity in real-time.

Figure 1.1 Similar Competitor

While SmartBin serves as an advanced commercial solution targeting large-scale waste management, our project focuses on addressing a localized need, particularly within educational institutions such as UI. By filling the gap in the UI area, the project aims to enhance waste management through a similar technological approach but with a focus on public awareness and education. The goal is not just to automate the process but to empower individuals to understand the types and quantities of waste they generate, encouraging long-term behavioral changes towards waste reduction and sustainable practices.

1.5 Key Features/Modules

Below are the key features of our project:

1. Waste Separation

   The system automatically sorts trash into wet and dry categories using an integrated moisture sensor. This feature helps users separate waste more effectively, ensuring that wet waste (such as food scraps) is processed differently from dry waste (like paper and plastics), facilitating more sustainable waste management practices such as composting and recycling.

2. Full Trash Detection

   The system uses ultrasonic sensors to detect when the trash bin is nearing its capacity. By measuring the distance between the sensor and the trash inside the bin, the system can accurately determine when the bin is full, helping to prevent overflow and ensuring timely waste collection or disposal.

3. Smart Notifications

   The system sends real-time notifications to users via mobile apps or web interfaces whenever the bin is full or when specific actions are needed, such as emptying the bin. This ensures users are always aware of the status of their trash, allowing for more proactive and efficient waste management.

4. Data Display

   The website provides users with real-time data on waste type and bin status, offering insights into the types and volumes of waste being generated. This feature helps users become more conscious of their waste production patterns and promotes better waste-sorting habits over time.

5. Control System

   The system is powered by an ESP32 microcontroller, which handles all sensor inputs and actuates the sorting mechanism. The ESP32 manages the waste separation process, monitors the bin's capacity, and sends data to the web interface, ensuring seamless and reliable operation.

6. Web Interface

   The system includes an easy-to-use web interface, allowing users to monitor the status of their trash bin remotely. Through the interface, users can view real-time updates on waste levels, track the separation process, and access historical data to gain insights into their waste habits. The interface is designed to be user-friendly, ensuring a smooth experience for users of all technical skill levels.

1.6 Tools and Technologies

Software

A. Backend and Database

1. Node.js

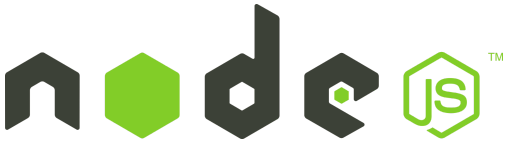

Figure 1.2 Node JS

As the backbone of our server-side application, Node.js provides a fast, asynchronous runtime environment for JavaScript code. By leveraging its event-driven architecture, we can ensure high performance and real-time data processing, which is crucial for handling sensor inputs and communicating with the web interface.

2. Express.js

Figure 1.3 Express JS

This is a powerful web application framework built on Node.js, which simplifies the process of building robust backend services. It handles routing, middleware integration, and API development efficiently, allowing for a scalable and maintainable backend infrastructure for the Trash Tracker system.

3. NeonDB

Figure 1.4 Neon DB

A modern, cloud-native database, NeonDb is designed to handle all data related to smart city infrastructure, sensor readings, and user interactions. Supports structured and unstructured data, providing scalable and efficient storage for diverse data types and real-time analytics.

4. PHPMyAdmin

Figure 1.5 PHP My Admin

A free and open-source administration tool for MySQL and MariaDB. It provides an easy-to-use web interface to manage databases, making tasks like running queries, creating and modifying tables, and managing user permissions much simpler.

B. Frontend

1. React

Figure 1.6 React

A widely used JavaScript library, React enables us to build dynamic, interactive user interfaces with ease. It handles the Trash Tracker's front-end components, allowing users to monitor real-time data, view notifications, and interact with the web platform seamlessly. React's component-based architecture ensures the front-end is modular and easy to maintain.

2. TailwindCSS

Figure 1.7 Tailwind CSS

TailwindCSS is a utility-first CSS framework that enables rapid UI development without writing complex custom styles. This framework ensures the front-end is responsive and visually appealing while providing flexibility in designing the user interface for our web application.

3. Vite

Figure 1.8 Vite

A modern build tool designed to enhance the development process, Vite provides a fast development environment, enabling hot module replacement and quick compilation times. It plays a key role in making front-end development faster and more efficient, which is crucial when working with tools like React.

Hardware

1. Wi-Fi Module (ESP32)

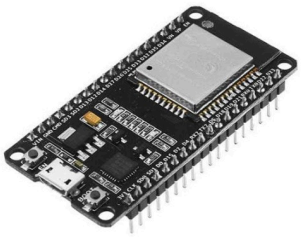

Figure 1.9 Wi-Fi Module Component

The ESP32 microcontroller serves as the primary hardware for connecting the system to the internet. It enables communication between the trash can's sensors and the web server, ensuring real-time data transmission. This module also handles remote control capabilities, allowing users to monitor the bin status from anywhere.

2. Moisture Sensor

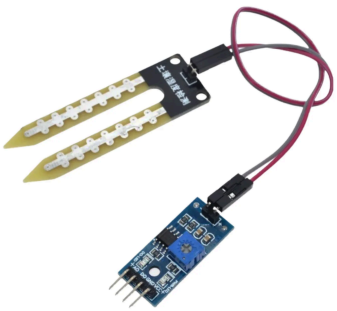

Figure 1.10 Moisture Sensor Component

A crucial component of the system, the moisture sensor is used to differentiate between wet and non-wet trash. By measuring the moisture level of the trash, it ensures accurate sorting into the appropriate categories, improving waste management and enabling proper disposal methods for wet and dry materials.

3. Ultrasonic Sensor (HC-SR04)

Figure 1.11 Ultrasonic Sensor Component

This sensor is responsible for measuring the level of trash inside the bin. By sending ultrasonic waves and calculating the time taken for the waves to return, it determines how full the bin is. This allows the system to notify users when the bin is reaching capacity, preventing overflow.

4. Servo Motor

Figure 1.12 Servo Motor Component

The servo motor controls the movement of the bin flap, which is responsible for physically sorting the trash into wet and dry compartments. The motor responds to input from the moisture sensor, ensuring the correct operation of the waste separation process.

Others

A. Communication

1. Line

A popular messaging platform that enables quick and easy communication among team members, allowing us to discuss project updates, share files, and manage tasks efficiently.

2. Discord

This platform provides voice, video, and text communication. It's especially useful for group discussions, real-time collaboration, and team meetings. There are also channels for specific topics (e.g., design, development, testing) to keep the conversations organized.

3. Trello

A project management tool that helps track progress and manage tasks. Trello allows the team to organize development phases, assign tasks, and set deadlines. It offers a clear visual overview of what needs to be done, by whom, and by when.

Figure 1.13 Communication Tools

B. Design

1. Figma

   A collaborative design tool that helps in creating wireframes, prototypes, and mockups for the website. It allows the design and development teams to collaborate in real-time, ensuring smooth transitions from design to development.

2. Visual Paradigm

   A professional design tool used to create UML diagrams, flowcharts, and system models. It helps in visualizing the system architecture and its components, aiding in clearer communication of system functionality.

3. Canva

   A user-friendly graphic design tool that helps in creating visually appealing presentations, posters, and social media graphics. It provides a wide range of templates and design elements, making it easy for anyone to create professional designs.

Figure 1.14 Design Tools

C. Documentation

1. GitHub

   This platform is used for version control, allowing multiple team members to work on the project simultaneously without conflicts. It stores the entire project's source code, tracks changes, and facilitates collaboration between developers.

2. Google Docs

   Used for collaborative documentation and sharing of project-related information, such as meeting notes, project outlines, and

technical specifications. Its real-time editing capabilities make it easy for teams to update and review documents together.

Figure 1.15 Documentation Tools

D. Coding

1. Visual Studio Code
   A free, open-source code editor developed by Microsoft. It provides powerful features like debugging, syntax highlighting, and version control integration. It supports a wide range of programming languages, making it a versatile tool for coding.

2. Arduino IDE
   An open-source platform used for writing and uploading code to Arduino boards and also other boards such as ESP32. It supports C and C++ programming languages and provides a simple, user-friendly interface for developing hardware projects.

Figure 1.16 Coding Tools

E. Environment

1. Xampp
   A free, open-source cross-platform web server solution stack package. It consists of Apache HTTP Server, MariaDB database, and interpreters for scripts written in PHP and Perl. It's widely used for web development and testing purposes.

2. MySQL Shell
   A widely-used open-source relational database management system (RDBMS). It allows users to manage and organize data, perform queries, and ensure data integrity. MySQL is known for its reliability, performance, and ease of use.

Figure 1.17 Environment Tools

F. Testing

1. Postman
   An API testing tool that simplifies the process of developing and testing APIs. It provides a user-friendly interface for sending requests, viewing responses, and organizing APIs. It supports automated testing and collaboration among team members.

2. Google Form
   A web-based tool that allows users to create surveys and questionnaires. It is easy to use and integrates with other Google services like Google Sheets for data analysis. It is useful for collecting and organizing information from respondents.

Figure 1.18 Testing Tools

1.7 Resources

The development of the Trash Tracker system relies on several key documentation resources to guide the hardware and software implementation:

1. ESP32 Datasheets
   Offering detailed technical specifications, the ESP32 datasheets are crucial for configuring Wi-Fi connectivity, managing power consumption, and ensuring smooth communication between the ESP32 and the web interface. They also provide insights into pin mappings and voltage requirements for correct hardware assembly.

2. API Documentation
   These resources include guidelines for integrating backend services, setting up the communication protocols between hardware and the web server, and building user-friendly interfaces for mobile and web applications. APIs are essential for real-time data transfer, notifications, and remote monitoring of the system.

1.8 Risk Analysis

Potential Risk	Likelihood	Potential Impact	Contingency
Sensor Accuracy	Moderate	Inaccurate waste separation, resulting in mixed trash categories.	Regular calibration and testing of sensors. Implement manual override for mixed materials or ambiguous readings.
Hardware Failures	High	Failure in sorting mechanism or missed notifications.	Design redundancy with backup sensors, regular hardware checks, and include manual controls for override if needed.
Network Connectivity Issues	Moderate	Loss of real-time data transmission and delayed notifications.	Implement offline data storage and use redundant network options; introduce automated reconnection protocols.
Security Breach	High	Unauthorized access, data breaches, or privacy violations.	Use encryption, conduct regular security audits, and ensure strong authentication methods are in place.
Server Downtime	Moderate	Inability to access the web interface or process new data.	Implement scheduled maintenance during off-peak hours and use backup servers; have rapid response protocols for downtimes.
User Interface Issues	Moderate	Confusion or misinterpretation of trash bin status and data.	Conduct frequent user testing, gather feedback, and follow an iterative design process to ensure ease of use.
Insufficient Power Supply	Moderate	System failure due to low power or incomplete trash sorting.	Incorporate backup power solutions, monitor battery levels, and optimize the system for low-power operation.
User Engagement Issues	High	Low user interaction with the web interface and disengagement.	Provide relevant notifications, offer insights on waste management, and integrate features that enhance user engagement.

Cost vs. Efficiency	Moderate	High costs could make the system less feasible for adoption.	Balance cost-efficient hardware with performance, explore affordable alternatives without compromising functionality.

Table 1.1 Risk Analysis

CHAPTER II
PROJECT MANAGEMENT

2.1 Project Integration Management

In the development of the waste monitoring website for the Faculty of Engineering, UI, project integration management played a crucial role in ensuring the effective combination of software engineering principles and project management methodologies. This approach aimed to apply knowledge from software engineering to the project's management processes. Key activities included identifying, defining, integrating, unifying, and coordinating various project elements to ensure smooth progress towards completion.

Integrated Project Planning

The project began with a comprehensive plan that outlined the project's vision, objectives, requirements, and constraints. This plan served as a guide, aligning the team's efforts with the overall project goals. It included detailed strategies for feature integration, such as monitoring data for wet and dry waste bins, risk management for sensor failures, and stakeholder involvement, such as admin users and the general public. By uniting these elements into a cohesive plan, the team established a clear roadmap for development, testing, and implementation, ensuring a synchronized and integrated approach throughout the project lifecycle.

Integrated Change Control

Throughout the project's lifecycle, any changes were identified and evaluated for their impact on other project components. A systematic approach to integrated change control was implemented, allowing the team to assess, understand the effects, and take appropriate action without disrupting the project timeline or objectives. This change control mechanism also allowed for adjustments based on new requirements, such as feature additions or interface updates, ensuring that the project remained flexible yet controlled while meeting stakeholder expectations.

Integrated Progress Monitoring and Control

Progress was continuously monitored against the established project plan, enabling real-time tracking of tasks, schedules, and milestone achievements. Any deviations were quickly addressed through coordinated corrective actions. This iterative monitoring and control process kept the project on track, allowing the team to proactively identify and resolve potential issues, such as sensor integration delays or data errors, ensuring swift decision-making to maintain project momentum.

2.2 Project Scope Management

Project scope management is essential in ensuring that the objectives and deliverables of the waste monitoring website for the Faculty of Engineering, University of Indonesia, are clearly defined and achieved. The scope covers the design, development, and implementation of the website, as well as the integration of sensors for monitoring waste bins.

2.2.1 Scope Planning

Scope planning involves detailed documentation of the project's planning process, outlining the project's goals, objectives, and the specific plans described in earlier sections. For the Trash Tracker project, this includes the planning required for developing the IoT-based trash bin system and its associated web interface. The project's success depends on meeting client expectations, ensuring proper functionality, and achieving completion of both hardware and software components.

2.2.2 Scope Definition

The project's scope is defined by the development of a website that allows users to monitor waste bins across the Faculty of Engineering. Each waste bin will be equipped with fullness and moisture sensors, providing real-time data on the condition of the bins. The website will display this information to users, including monthly waste trends in graphical form. While general users can view the data without logging in, admin users will have the capability to log in and manage or update the data as needed. The scope thus includes both the hardware integration (sensors) and the software components (website and database) necessary to meet these objectives.

2.2.3 Scope Verification

Scope verification ensures that all project deliverables meet stakeholder expectations and requirements. The final system will be verified through the following documentation and validation processes:

a. Software Project Plan: Outlines the development methodology, timelines, and milestones for the website.
b. System Design Diagram: Visualizes the architecture of the website, including sensor integration, user interaction, and database structure.
c. Test Plan: Details the testing procedures to ensure sensor accuracy, data synchronization, and system reliability for both users and administrators.

d. Stakeholder Agreement: Documents the agreement between project stakeholders, including any necessary revisions or updates to the system.
e. User Guide: Provides clear instructions for both admin and general users on how to operate the website and interpret the waste data.

2.3 Software Development Cycle

The software development cycle for the Trash Tracker project outlines the stages involved in creating the software components that power the smart trash bin and its web interface. This project will adhere to industry-standard practices, utilizing an Agile methodology for iterative development and continuous enhancement. The key phases are as follows:

1. Requirements Gathering

   In this initial phase, the project team will gather and document all functional and nonfunctional requirements for the Trash Tracker system. This will include consultations with stakeholders to understand their needs for waste separation, notification systems, and web-based monitoring, as well as any specific technical constraints.

2. Design

   During the design phase, the team will create the system's architectural blueprint and user interface mockups. Focus will be placed on how the sensors, control systems, and web interface will interact with each other. The design will ensure seamless integration of hardware (sensors, actuators) and software (data management, real-time monitoring).

3. Development

   In this phase, the software development team will begin coding the different components of the Trash Tracker system, including the sensor management, notifications, and web interface. The Agile methodology will allow for flexibility and adaptability, making it possible to respond quickly to evolving requirements or new challenges.

4. Testing

   Throughout the development process, ongoing quality assurance testing will be conducted. This will include unit testing of individual components (such as sensors and notification systems), integration testing to ensure all modules work together, user acceptance testing for the web interface, and performance testing to verify the system operates efficiently under load.

5. Deployment

   Once the software has passed all testing phases, it will be deployed. The deployment will ensure the system functions properly in real-world environments, including waste separation and notification processes. Deployment may be phased to minimize disruption and allow for real-time feedback.

6. Maintenance and Iteration

   After deployment, the software will require regular updates and improvements based on user feedback and technological advancements. Using the Agile approach, the team will conduct regular iterations to enhance system functionality and address any bugs or performance issues, ensuring the Trash Tracker system remains responsive and up-to-date.

2.4 Timeline

	September				Oktober				November				Desember			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Project Initialization																
Project Scope and Objectives																
Defining project idea and objectives																
Consultation with Lecturer																
Creation of Project Plan and Timeline																
Team Information and Setup																
Assemble the Development Team																
Finalizing the Idea																
Assign Roles and Responsibilities																
Design and Prototyping																
Database Design																
Create Database Scheme																
Define Database Tables and Relationships																
Create API for Integration with Frontend																
Frontend Development																
Create UI/UX Design																
Create Wireframe and App Pages Prototype																
Integrate the Frontend Design with Database																
IoT Device Design																
Gather the Components																
Make Program																
Development																
Authentication and User Management																
Integration of IoT Device and Website Application																
Real-Time Device Monitoring from Website Application																
Deployment																
Final Testing with Client Included																
Deploy the Website and Device																

Table 2.1 Project Timeline

2.5 Responsibilities

Name	Roles
Mario Matthews Gunawan	Brainstorming Session, Diagramming, Hardware Development, Update Documentation
Annisa Ardelia Setiawan	Brainstorming Session, Back-End Development, Database Handling, Update Documentation
Rizqi Zaidan	Brainstorming Session, Frontend Development, UX/UI Designer, Update Documentation

Table 2.2 Members Responsibilities

2.6 Project Cost

Component	Estimation Cost	Description
Hardware	IDR 150.000,00	The hardware cost estimation for building one Trash Tracker device includes essential components such as ESP32, moisture sensor, ultrasonic sensor, and servo motor. -ESP32: IDR 50.000,00 -3 Moisture Sensor: IDR 45.000,00 -Ultrasonic Sensor: IDR 15.000,00 -Servo Motor: IDR 20.000,00 -Jumper: IDR 20.000,00
Software and Database	IDR 0,00	The software and databases used for the Trash Tracker project are obtained at no direct cost, as the project utilizes free development tools such as: - Visual Studio Code (Free) - MongoDB (Free) - GitHub (Free)
Design	IDR 0,00	No direct cost for design and modeling, as the project uses free software tools like: - Figma (Free) - Canva (Free)

Table 2.3 Project Cost Estimation

CHAPTER III

PROJECT ARCHITECTURE & DESIGN

3.1 UML Diagram

3.1.1 Use Case Diagram

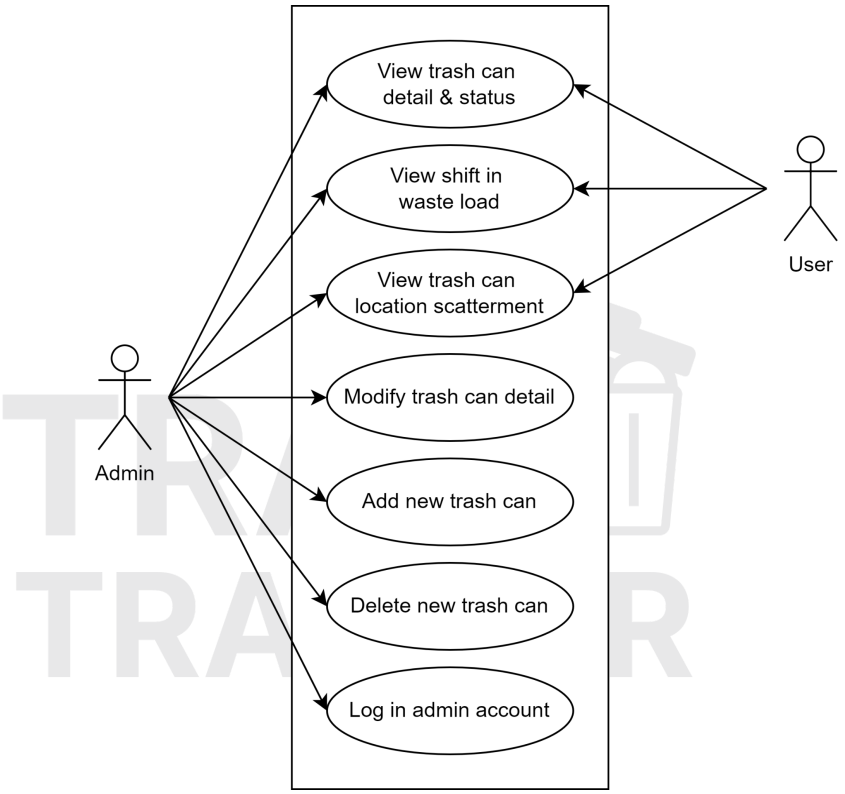

Figure 3.1 Use Case Diagram

This diagram visualizes the use of the project by two parties, the admin and the user.

- User: Represents the general user who interacts with the system. The user can view the details and status of each trash bin, check the waste load shifts, and see the location of each bin without needing to log in.
- Admin: Represents an administrator who has higher privileges and can perform administrative tasks. The admin can do everything the user does, plus additional actions such as modifying website details, adding & deleting bins, and, of course, must log in first.

The diagram clearly illustrates the separate roles and responsibilities of both users and admins within the project, emphasizing the different functionalities accessible to each. This visualization provides a useful reference for grasping the system's structure and the interactions between its various elements.

3.1.2 State Diagram

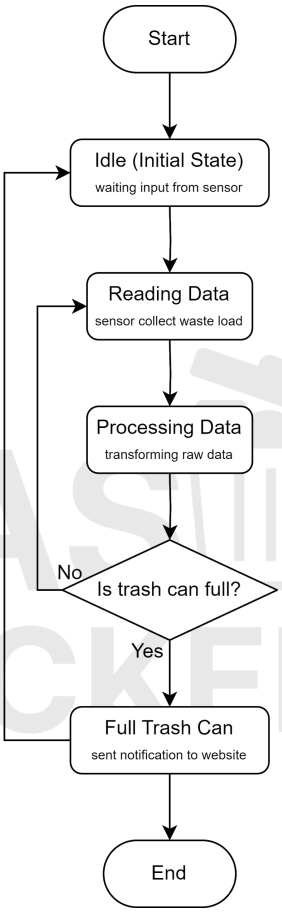

Figure 3.2 State Diagram

This state diagram visually represents the flow of operations within a system that monitors waste levels in a trash bin and sends notifications when it's full.

Flow of diagram:

- Start: The process begins.
- Idle (Initial State): The system starts in an idle state, awaiting input from a sensor.
- Reading Data: The system gathers data from a sensor that measures the waste level in the trash bin.

- Processing Data: The collected sensor data is processed and converted into a useful format.
- Is the trash bin full?: A decision point based on the processed data determines if the trash bin is full or not.
    - No: If the bin isn't full, the system returns to the "Idle (Initial State)" and waits for further sensor input.
    - Yes: If the bin is full, a notification is sent to a website, signaling that the trash bin needs to be emptied.
- End: The process completes.

The state diagram demonstrates the logical flow and operations of the system, showing how it constantly monitors the trash bin's waste level and triggers a notification when needed.

3.1.3 Activity Diagram

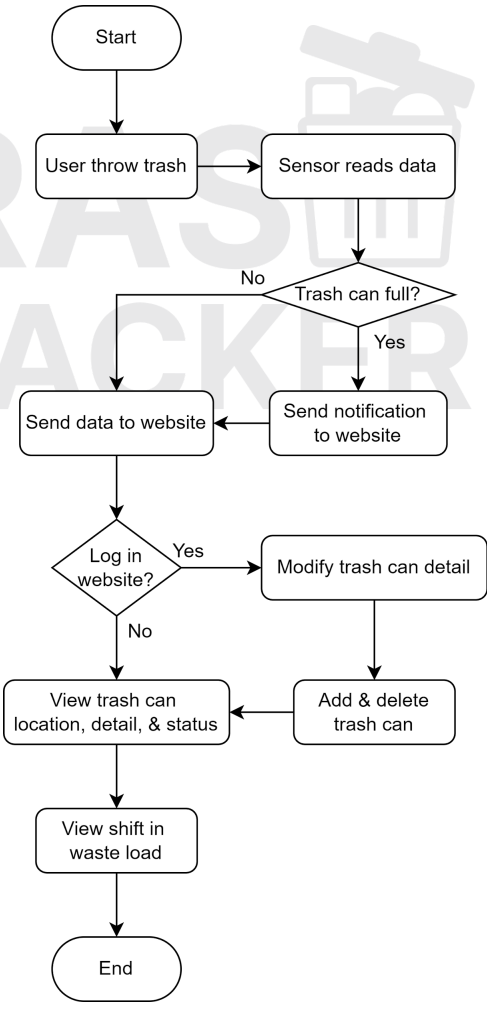

Figure 3.3 Activity Diagram

This diagram shows the flow of activities in a process. In this case, it outlines the workflow of a trash can monitoring system.

Flow of diagram:

- Start: The process begins.
- User throws trash: A user disposes of trash in the bin.
- Sensor reads data: A sensor inside the trash can detects the waste and gathers data on the trash level.
- Is the trash can full?: The system checks whether the trash can is full based on the sensor's data.
    - No: If the can isn’t full, the system returns to the "Sensor reads data" step to continue monitoring.
    - Yes: If the trash can is full, a notification is sent to a website.
- Log in to website?: The system checks if a user is logged in.
    - No: If no user is logged in, it proceeds to "Send data to website."
    - Yes: If a user is logged in, they can modify trash can details.
- Modify trash can details: The logged-in user can update information about the trash can, such as location or status.
- Add & delete trash cans: The user can add or remove trash cans from the system.
- View trash can location, details, & status: The user can check details about the trash can, including its location, status, and more.
- View waste load shifts: The user can also track how the trash load has changed over time.
- End: The process concludes.

The activity diagram illustrates how the system monitors trash levels, sends notifications when the bin is full, and allows users to manage and view trash can information.

3.1.4 Class Diagram

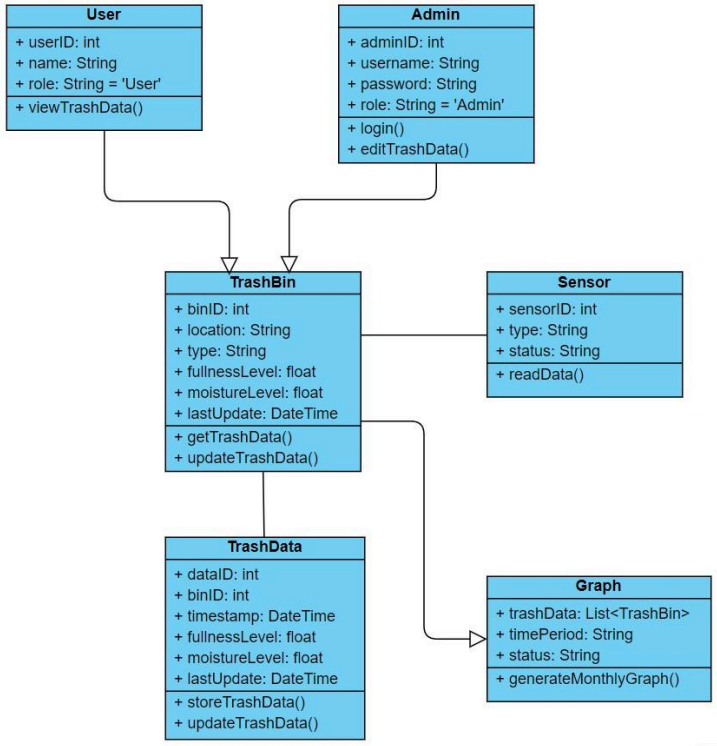

Figure 3.4 Class Diagram

This class diagram shows the structure and relationships between classes in a trash tracking application.

Classes:

- User: Represents a user with attributes like userID, name, role, and the method viewTrashData().
- Admin: Inherits from User, with additional attributes like adminID, username, password, and methods like login() and editTrashData().
- TrashBin: Represents a trash bin, with attributes such as binID, location, type, fullnessLevel, and methods like getTrashData() and updateTrashData().
- Sensor: Represents a sensor attached to a bin, with attributes like sensorID, type, and status, and a method readData().
- TrashData: Stores sensor data with attributes like dataID, binID, timestamp, and methods storeTrashData() and updateTrashData().
- Graph: Generates a graph of trash data, with attributes like trashData, timePeriod, and status, and a method generateMonthlyGraph().

Relationships:

- Association: Connects classes like User to TrashBin for viewing data, and Sensor to TrashBin.
- Inheritance: Admin inherits from User.
- Aggregation: TrashBin aggregates TrashData, meaning TrashData can exist independently.

This class diagram outlines the application's structure and relationships, helping to visualize its design and functionality.

3.2 Design

3.2.1 Website Design

- Home page

Figure 3.5 Home Page Part 1

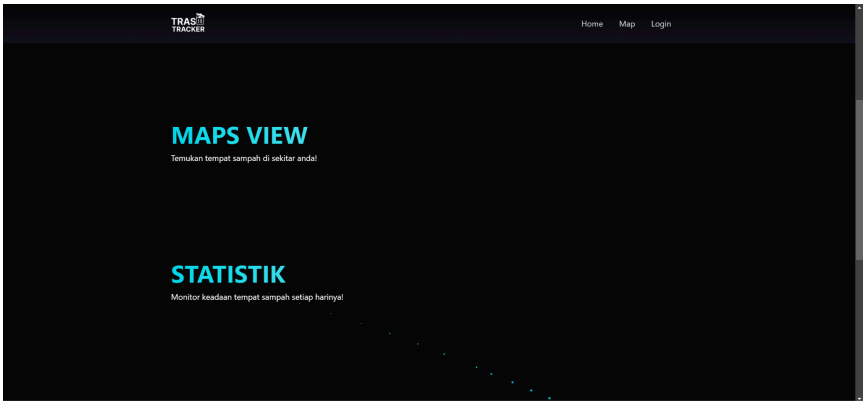

Figure 3.6 Home Page Part 2

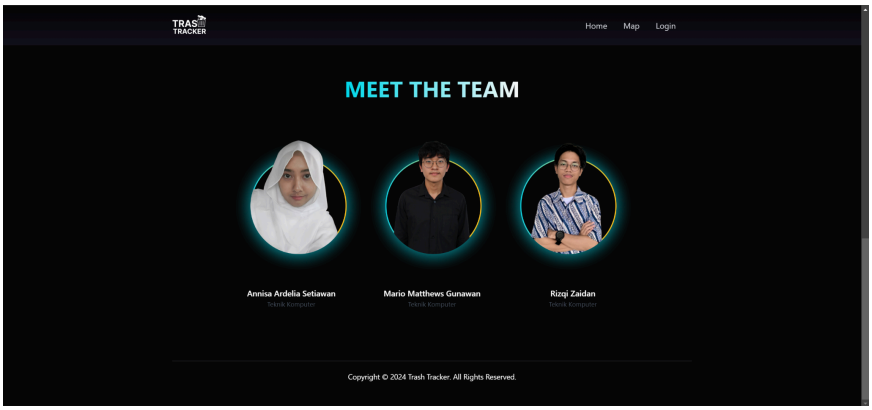

Figure 3.7 Home Page Part 3

- Map page

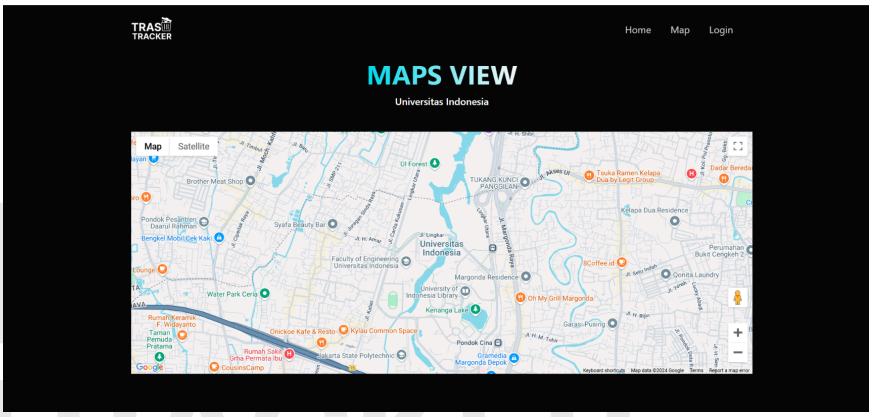

Figure 3.8 Map Page Part 1

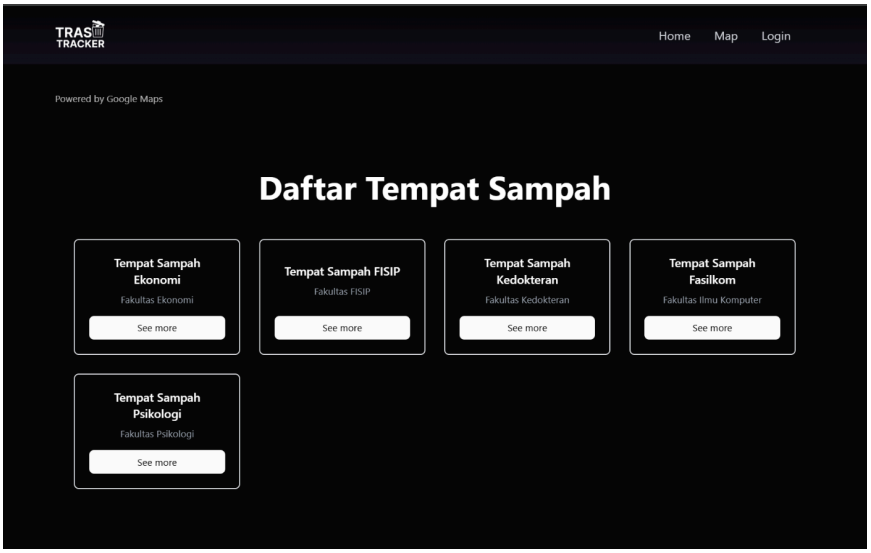

Figure 3.9 Map Page Part 2

- Trash detail page

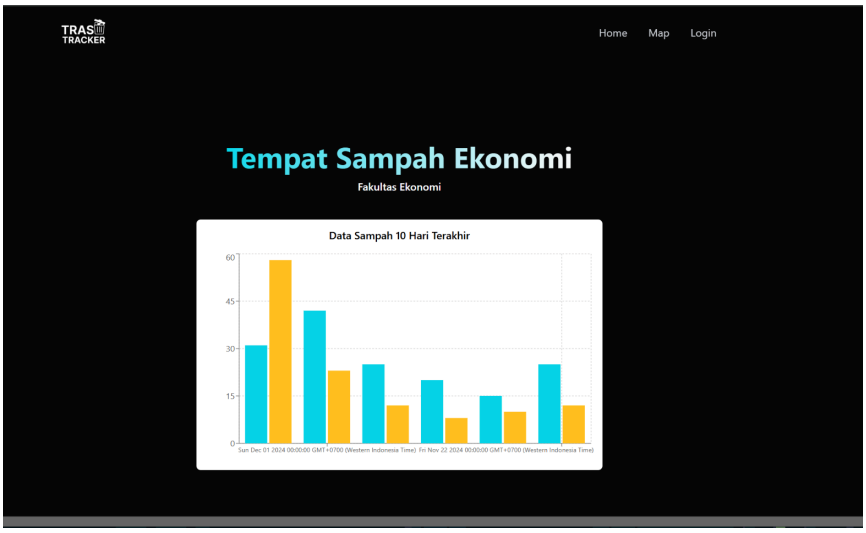

Figure 3.10 Trash Detail Page

- Data management page

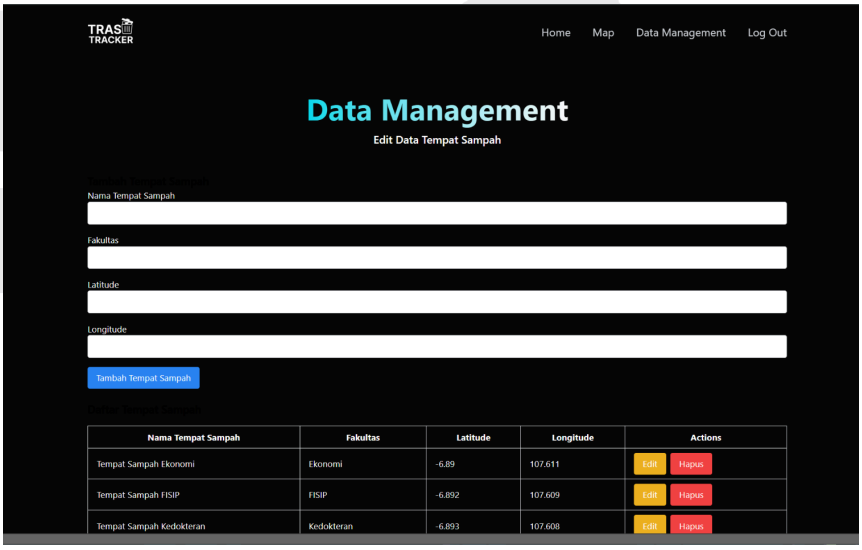

Figure 3.11 Data Management Page

- Login page

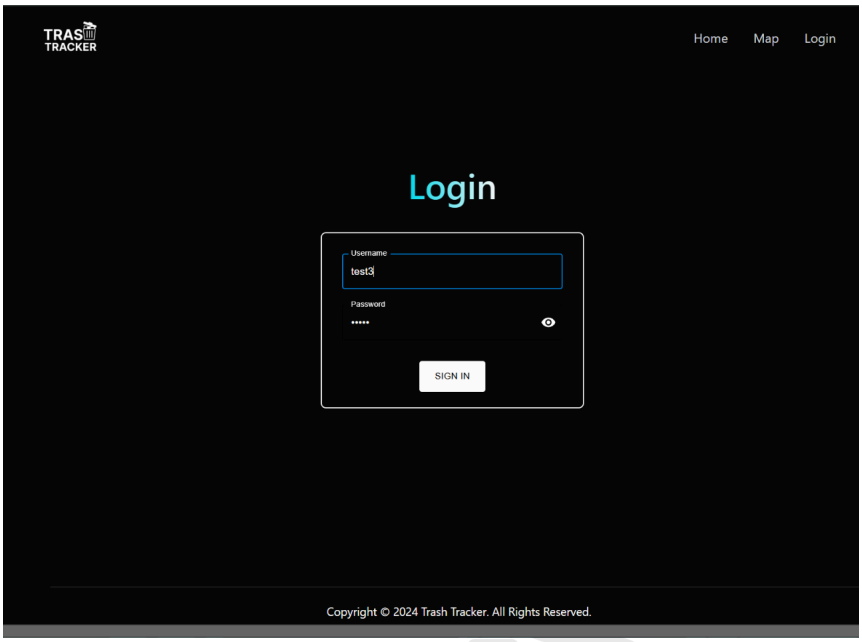

Figure 3.12 Data Management Page

3.2.2 Device Design

Figure 3.13 Device Design Part 1

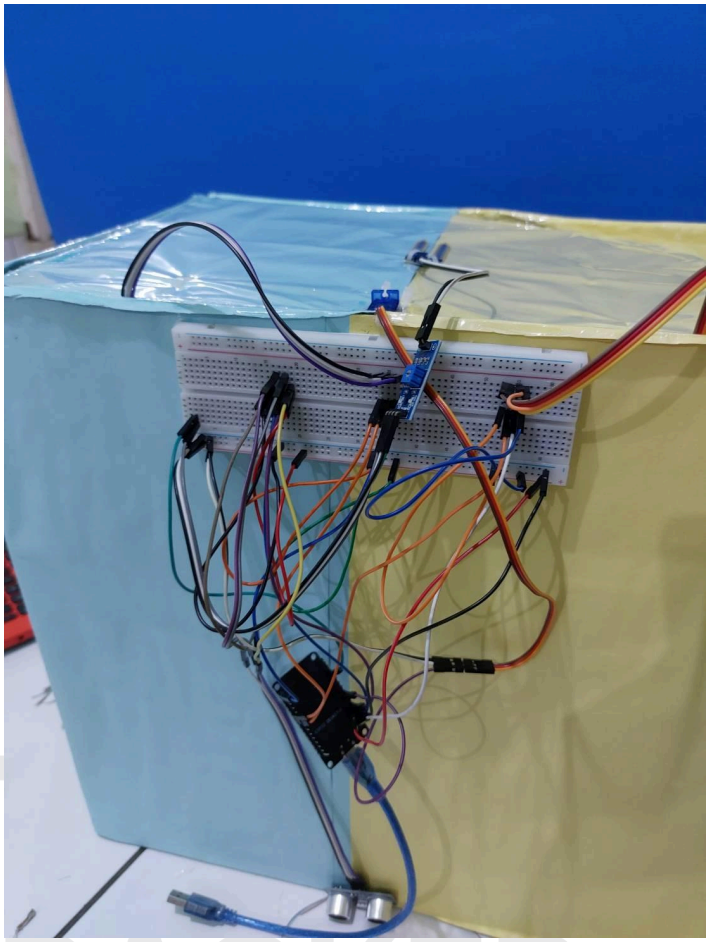

Figure 3.14 Device Design Part 2

CHAPTER IV

CODE IMPLEMENTATION

4.1 Back-End

4.1.1 Admin Register

```
//Register Admin
const Register = async (req, res) => {
    const { username, pass } = req.body;

    try {
      // Check if the username already exists in the database
      const query = "SELECT * FROM admin WHERE username = $1";
      const result = await db.query(query, [username]);

      if (result.rows.length > 0) {
           return res.status(400).json({ error: 'Username already
exists' });
      }

      // Hash the password using bcrypt
      console.log(pass);
      const hashedPassword = await bcrypt.hash(pass, 10);

      // Insert the admin information into the database
        const insertQuery = 'INSERT INTO admin (username, pass)
VALUES ($1, $2)';
      await db.query(insertQuery, [username, hashedPassword]);

            res.status(200).json({ message: 'Admin registered
successfully' });
    } catch (error) {
      console.error('Error registering admin:', error);
      res.status(500).json({ error: 'Failed to register admin' });
    }
};
```

4.1.2 Admin Login

```
const Login = async (req, res) => {
    const { username, pass } = req.body;

    db.query(
      `SELECT * FROM admin WHERE username = '${username}'`,
      async (err, result) => {
        if (result.rows.length === 0) {
            return res.status(401).json({ error: "Invalid username
or password" });
        }
        if (err) {
          console.error("Error executing query", err);
          return;
        }
        const storedData = result.rows[0];
              const passwordMatch = await bcrypt.compare(pass,
storedData.pass.trim());

        if (passwordMatch) {
```

```
                    return res.status(200).json({ message: 'Login
successful' });
        } else {
          // Password does not match
          console.log(storedData);
            return res.status(401).json({ error: 'Invalid username
or password' });
        }
      }
    );
};
```

4.1.3 Add Trash Bin

```
// Fungsi untuk menambahkan tempat sampah
const addTempatSampah = async (req, res) => {
    const { nama, fakultas, latitude, longitude } = req.body;

    // Validasi input
    if (!nama || !fakultas || !latitude || !longitude) {
        return res.status(400).json({
                message: 'All fields (nama, fakultas, latitude,
longitude) are required.',
        });
    }

    try {
        // Query untuk menambahkan tempat sampah baru
        const result = await db.query(
              `INSERT INTO tempat_sampah (nama, fakultas, latitude,
longitude)
             VALUES ($1, $2, $3, $4) RETURNING *`,
            [nama, fakultas, latitude, longitude]
        );

        // Mengembalikan data tempat sampah yang ditambahkan
        res.status(201).json({
            message: 'Tempat sampah berhasil ditambahkan.',
            tempatSampah: result.rows[0],
        });
    } catch (error) {
              console.error('Error adding new trash location:',
error.message);
        res.status(500).json({
                message: 'An error occurred while adding the trash
location.',
        });
    }
};
```

4.1.4 Delete Trash Bin

```
// Fungsi untuk menghapus tempat sampah
const deleteTempatSampah = async (req, res) => {
    const { id } = req.params;

    try {
        // Query untuk menghapus tempat sampah berdasarkan id
          const result = await db.query('DELETE FROM tempat_sampah
WHERE id = $1', [id]);
```

```
        // Jika tidak ada tempat sampah dengan id tersebut
        if (result.rowCount === 0) {
              return res.status(404).json({ message: 'Tempat sampah
tidak ditemukan.' });
        }

          res.status(200).json({ message: 'Tempat sampah berhasil
dihapus.' });
    } catch (error) {
                  console.error('Error deleting trash location:',
error.message);
        res.status(500).json({
              message: 'An error occurred while deleting the trash
location.',
        });
    }
};
```

4.1.5 Edit Trash Bin

```
// Fungsi untuk mengedit tempat sampah
const editTempatSampah = async (req, res) => {
    const { id } = req.params; // Mengambil ID dari parameter URL
     const { nama, fakultas, latitude, longitude } = req.body; //
Data yang ingin diperbarui

    // Validasi input
    if (!id) {
        return res.status(400).json({
            message: 'Tempat sampah ID is required.',
        });
    }

    try {
        // Query untuk memperbarui tempat sampah berdasarkan ID
        const result = await db.query(
            `UPDATE tempat_sampah
            SET nama = COALESCE($1, nama),
                fakultas = COALESCE($2, fakultas),
                latitude = COALESCE($3, latitude),
                longitude = COALESCE($4, longitude)
            WHERE id = $5
             RETURNING *`,
            [nama, fakultas, latitude, longitude, id]
        );

         // Jika tidak ada baris yang diperbarui, berarti ID tidak
ditemukan
        if (result.rowCount === 0) {
            return res.status(404).json({
                message: `No trash location found with ID ${id}.`,
            });
        }

        // Mengembalikan data tempat sampah yang diperbarui
        res.status(200).json({
            message: 'Tempat sampah berhasil diperbarui.',
            tempatSampah: result.rows[0],
        });
    } catch (error) {
```

```
                console.error('Error updating trash location:',
error.message);
        res.status(500).json({
              message: 'An error occurred while updating the trash
location.',
        });
    }
};
```

4.1.6 Get Trash Bin Data

```
// Fungsi untuk mengambil data tempat sampah
const getTempatSampah = async (req, res) => {
    try {
         const result = await db.query('SELECT * FROM tempat_sampah
ORDER BY id ASC');
        const tempatSampah = result.rows;

        res.status(200).json(tempatSampah);
    } catch (error) {
        console.error('Error fetching tempat sampah:', error);
        res.status(500).json({ error: 'Internal Server Error' });
    }
};
```

4.1.7 Get Today Trash Data

```
// Fungsi untuk mengambil jumlah sampah basah dan kering pada hari
ini untuk tempat sampah berdasarkan id
const getTrashData = async (req, res) => {
    try {
        // Mendapatkan id tempat sampah dari parameter URL
        const { id } = req.params;

        // Mengambil tanggal hari ini
          const today = new Date().toISOString().split('T')[0]; //
Format YYYY-MM-DD

         // Query untuk mendapatkan jumlah sampah basah dan kering
hari ini berdasarkan tempat sampah
        const query = `
                    SELECT t.id AS tempat_sampah_id, t.nama AS
tempat_sampah, s.jenis_sampah, SUM(s.jumlah_sampah) AS total
            FROM sampah s
            JOIN tempat_sampah t ON s.tempat_sampah_id = t.id
             WHERE t.id = $1 AND s.tanggal = $2 AND (s.jenis_sampah
= 'basah' OR s.jenis_sampah = 'kering')
            GROUP BY t.id, t.nama, s.jenis_sampah
            ORDER BY s.jenis_sampah;
        `;
        const result = await db.query(query, [id, today]);

        // Jika tidak ada data
        if (result.rows.length === 0) {
                return res.status(404).json({ message: `Belum ada
sampah hari ini` });
        }

        // Menyusun data untuk mengembalikan response
        const data = result.rows.reduce((acc, row) => {
```

```
            if (row.jenis_sampah === 'basah') {
                acc.basah = row.total;
            } else if (row.jenis_sampah === 'kering') {
                acc.kering = row.total;
            }
            return acc;
        }, { basah: 0, kering: 0 });

        res.status(200).json(data);
    } catch (error) {
        console.error('Error fetching data:', error);
        res.status(500).json({ error: 'Error fetching data' });
    }
};
```

4.1.8 Get Trash Data History

```
// Fungsi untuk mengambil jumlah sampah basah dan kering selama 10
hari terakhir berdasarkan tempat sampah
const getDailyTrashData = async (req, res) => {
    try {
        // Mendapatkan id tempat sampah dari parameter URL
        const { id } = req.params;

        // Mengambil tanggal hari ini dan 10 hari sebelumnya
        const today = new Date();
        const last10Days = new Date();
        last10Days.setDate(today.getDate() - 10);

        // Format tanggal menjadi YYYY-MM-DD
        const todayFormatted = today.toISOString().split('T')[0];
                                    const    last10DaysFormatted    =
last10Days.toISOString().split('T')[0];

         // Query untuk mendapatkan jumlah sampah basah dan kering
dalam 10 hari terakhir berdasarkan tempat sampah
        const query = `
                    SELECT t.id AS tempat_sampah_id, t.nama AS
tempat_sampah, s.tanggal, s.jenis_sampah, SUM(s.jumlah_sampah) AS
total
            FROM sampah s
            JOIN tempat_sampah t ON s.tempat_sampah_id = t.id
              WHERE t.id = $1 AND s.tanggal BETWEEN $2 AND $3 AND
(s.jenis_sampah = 'basah' OR s.jenis_sampah = 'kering')
            GROUP BY t.id, t.nama, s.tanggal, s.jenis_sampah
            ORDER BY s.tanggal DESC, s.jenis_sampah;
        `;
                    const   result   =   await   db.query(query,   [id,
last10DaysFormatted, todayFormatted]);

        // Jika tidak ada data
        if (result.rows.length === 0) {
                return res.status(404).json({ message: `Belum ada
sampah 10 hari terakhir` });
        }

        // Menyusun data untuk mengembalikan response
        const data = result.rows.reduce((acc, row) => {
            const date = row.tanggal;
            if (!acc[date]) {
                acc[date] = { basah: 0, kering: 0 };
```

```
            }
            if (row.jenis_sampah === 'basah') {
                acc[date].basah = row.total;
            } else if (row.jenis_sampah === 'kering') {
                acc[date].kering = row.total;
            }
            return acc;
        }, {});

          // Menyusun data agar urutan tanggalnya berurutan dari
yang terbaru
        const sortedData = Object.keys(data)
            .sort((a, b) => new Date(b) - new Date(a))
            .map(date => ({
                date,
                basah: data[date].basah,
                kering: data[date].kering
            }));

        res.status(200).json(sortedData);
    } catch (error) {
        console.error('Error fetching data:', error);
        res.status(500).json({ error: 'Error fetching data' });
    }
};
```

4.1.9 Add Trash Data

```
const addDataSampah = async (req, res) => {
    const { tanggal, jenis_sampah, jumlah_sampah, tempat_sampah_id
} = req.body;

    // Validasi input
     if (!tempat_sampah_id || !jenis_sampah || !jumlah_sampah ||
!tanggal) {
        return res.status(400).json({
                    message: 'All fields (tanggal, jenis_sampah,
jumlah_sampah, tempat_sampah_id) are required.',
        });
    }

    try {
        // Query untuk menambahkan data sampah baru
        const result = await db.query(
                    `INSERT INTO sampah (tanggal, jenis_sampah,
jumlah_sampah, tempat_sampah_id)
            VALUES ($1, $2, $3, $4) RETURNING *`,
                        [tanggal, jenis_sampah, jumlah_sampah,
tempat_sampah_id]
        );

        // Mengembalikan data sampah yang ditambahkan
        res.status(201).json({
            message: 'Data sampah berhasil ditambahkan.',
            sampah: result.rows[0],
        });
    } catch (error) {
                console.error('Error adding new trash data:',
error.message);
        res.status(500).json({
```

```
                message: 'An error occurred while adding the trash
data.',
            });
    }
};
```

4.1.10 Get All Sensor Data

```
// Get All Sensor Data
const getAllSensors = async (req, res) => {
     const query = 'SELECT * FROM sensor_dataa ORDER BY timestamp
ASC LIMIT 5';

    db.query(query, (err, results) => {
        if (err) {
                console.error('Error fetching all data from local
database:', err);
            return res.status(500).json({ error: 'Failed to fetch
all sensor data' });
        }
        res.status(200).json({ data: results });
    });
};
```

4.1.11 Get Latest Sensor Data

```
// Get Latest Sensor Data
const getLatestSensor = async (req, res) => {
     const query = 'SELECT * FROM sensor_dataa ORDER BY timestamp
DESC LIMIT 1';

    db.query(query, (err, results) => {
        if (err) {
              console.error('Error fetching the latest data from
local database:', err);
            return res.status(500).json({ error: 'Failed to fetch
the latest sensor data' });
        }
        if (results.length === 0) {
             return res.status(404).json({ message: 'No sensor data
available' });
        }
        res.status(200).json({ data: results[0] });
    });
};
```

4.2 Front-End

4.2.1 Bar Chart

```
import React, { useEffect, useState } from "react";
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip,
ResponsiveContainer } from "recharts";

const SampahBarChart = ({ tempatSampahId, nama, fakultas }) => {
  const [data, setData] = useState([]);

  useEffect(() => {
    const fetchData = async () => {
      try {
```

```
                                const response = await
fetch(`http://localhost:3001/DailySampah/${tempatSampahId}`);
        const sampahData = await response.json();

        const chartData = sampahData.map(item => ({
          date: item.date,
          basah: item.basah,
          kering: item.kering,
        }));

        setData(chartData);
      } catch (error) {
        console.error("Error fetching data:", error);
      }
    };

    if (tempatSampahId) {
      fetchData();
    }
  }, [tempatSampahId]);

  return (
       <div  className="flex  flex-col  justify-center  items-center
p-8">
            <h1  className="font-poppins  text-center  text-[55px]
justify-center mb-12 pt-16">
                        <span   className="text-gradient   font-bold
text-[55px]">{nama}</span><br />
      </h1>
            <h2  className="font-poppins  text-center  text-[20px]
justify-center -mt-12 mb-12">
                         <span   className="text-white   text-[20px]
font-semibold">Fakultas {fakultas}</span>
      </h2>

           <div  className="w-full  max-w-3xl  bg-white  rounded-lg
shadow-md p-4">
            <h2 className="text-center text-xl font-semibold mb-4
text-black">
          Data Sampah 10 Hari Terakhir
        </h2>

        {data.length > 0 ? (
          <ResponsiveContainer width="100%" height={400}>
            <BarChart data={data} >
              <CartesianGrid strokeDasharray="3 3" />
              <XAxis dataKey="date" tick={{ fontSize: 12 }} />
              <YAxis />
              <Tooltip />
                  <Bar dataKey="basah" fill="#35D5E7" name="Sampah
Basah" /> {/* Ganti warna */}
                <Bar dataKey="kering" fill="#ffc300" name="Sampah
Kering" /> {/* Ganti warna */}
            </BarChart>
          </ResponsiveContainer>
        ) : (
                <p  className="text-center  text-gray-500">Loading
data...</p>
        )}
      </div>
    </div>
  );
};
```

4.2.2 Pie Chart

```
import React, { useEffect, useState } from "react";
import { PieChart, Pie, Cell, ResponsiveContainer } from
"recharts";

const SampahPieChart = ({ tempatSampahId }) => {
  const [data, setData] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
   const [totalSampah, setTotalSampah] = useState(0); // Untuk
menyimpan total sampah
   const [statusSampah, setStatusSampah] = useState(""); // Untuk
menyimpan status sampah

  useEffect(() => {
    const fetchData = async () => {
      try {
        if (!tempatSampahId) throw new Error("Trash location ID is
required.");

                                    const response = await
fetch(`http://localhost:3001/JumlahSampah/${tempatSampahId}`);

        if (!response.ok) {
          if (response.status === 404) {
            const errorData = await response.json();
                  throw new Error(errorData.message || "No data
available.");
          }
          throw new Error("Failed to fetch trash data.");
        }

        const sampahData = await response.json();

        // Menyusun data untuk PieChart
        const basah = parseInt(sampahData.basah) || 0;
        const kering = parseInt(sampahData.kering) || 0;

        // Menentukan total sampah
        const total = basah + kering;
        setTotalSampah(total);

        // Menentukan status berdasarkan total sampah
        if (total <= 25) {
          setStatusSampah("Low");
        } else if (total <= 50) {
          setStatusSampah("Medium");
        } else {
          setStatusSampah("High");
        }

        const chartData = [
          { id: "basah", value: basah, label: "Sampah Basah" },
          { id: "kering", value: kering, label: "Sampah Kering" },
        ];

        setData(chartData);
        setLoading(false);
      } catch (err) {
        setError(err.message);
        setLoading(false);
      }
    };
```

```
    fetchData();
  }, [tempatSampahId]);

  // Menentukan warna untuk pie chart
  const COLORS = ["#35D5E7", "#ffc300"]; // Warna untuk pie chart

  if (loading) {
        return  <p  className="text-center  text-gray-500">Loading
data...</p>;
  }

  if (error) {
    return <p className="text-center text-red-500">{error}</p>;
  }

  return (
        <div  className="flex  justify-center  items-center  -mt-20
mb-12">
        <div  className="flex  flex-row  w-full  max-w-4xl  bg-primary
rounded-lg shadow-md p-4">

        {/* Menampilkan total sampah di sebelah kiri */}
         <div  className="flex  flex-col  justify-center  items-center
mr-8">
          <p  className="text-white  text-lg">
            Total Sampah: {totalSampah} items
          </p>
           <p  className={`text-lg  font-semibold  ${statusSampah  ===
"Low"   ?   "text-green-500"   :   statusSampah   ===   "Medium"   ?
"text-yellow-500" : "text-red-500"}`}>
            {statusSampah}
          </p>
        </div>

        {/* Pie Chart */}
        <div  className="flex-1">
             <h1  className="font-poppins  font-semibold  text-white
text-center  text-[30px]  justify-center  mb-12  pt-8">
            Komposisi sampah hari ini
          </h1>

          {data.length  >  0  ?  (
            <ResponsiveContainer  width="100%"  height={300}>
              <PieChart  className="bg-primary">
                <Pie
                  data={data}
                  dataKey="value"
                  nameKey="label"
                  innerRadius={60}
                  outerRadius={80}
                  label={(entry)  =>  entry.label}
                >
                  {data.map((entry,  index)  =>  (
                    <Cell
                      key={`cell-${entry.id}`}
                           fill={COLORS[index  %  COLORS.length]}  //
Mengubah warna pie chart
                    />
                  ))}
                </Pie>
              </PieChart>
            </ResponsiveContainer>
```

```
          ) : (
                  <p  className="text-center  text-gray-500">No  data
available  for  this  trash  location.</p>
          )}
        </div>
      </div>
    </div>
  );
};
```

4.3 IoT

```
#include <ESP32Servo.h> // Pustaka untuk ESP32
#include <WiFi.h>
#include <HTTPClient.h>

// Servo motor setup
Servo servo1;

const int trigPin = 12;
const int echoPin = 13; // Ganti dengan pin yang sesuai
long duration;
int distance = 0;
int potPin = 34; // ESP32 menggunakan pin ADC seperti 34 atau 35
int soil = 0;
int fsoil;
int  maxDryValue  =  1;          //  Nilai  kelembapan  untuk  menentukan  jenis
sampah
int Ultra_Distance = 25;  // Jarak sensor ultrasonik ke sensor kelembapan
dalam cm

// Additional ultrasonic sensors
#define TRIG_PIN1 5   // Trigger Pin for Sensor 1
#define ECHO_PIN1 18  // Echo Pin for Sensor 1
#define TRIG_PIN2 23  // Trigger Pin for Sensor 2
#define ECHO_PIN2 19  // Echo Pin for Sensor 2

// WiFi Configuration
#define WIFI_SSID "Alia"
#define WIFI_PASSWORD "12345678"

// Server Configuration
const char* serverUrl = "http://192.168.66.76/makersgroup/sensor.php";

String sendval1, sendval2, sendval3, sendval4, postData;

void checkWiFiConnection() {
  while (WiFi.status() != WL_CONNECTED) {
    Serial.println("Connecting to Wi-Fi...");
    delay(1000);
  }
  Serial.println("Connected to Wi-Fi");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}
```

```
float measureDistance(uint8_t trigPin, uint8_t echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  float distance = (duration * 0.034) / 2; // Convert time to distance in
cm

  // Handle invalid readings
  if (distance > 35) {
    distance = 0; // Cap the distance to 0 cm if it exceeds
  } else {
    distance = 35 - distance; // Adjust distance based on the new formula
  }

  return distance;
}

void setup() {
  Serial.begin(115200);

  // Servo motor and soil sensor setup
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  servo1.attach(26); // Ganti 26 dengan pin PWM yang sesuai untuk ESP32
  Serial.println("Soil Sensor     Ultrasonic          Servo");
  Serial.println(".........");

  // WiFi setup
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  checkWiFiConnection();

  // Additional ultrasonic sensors setup
  pinMode(TRIG_PIN1, OUTPUT);
  pinMode(ECHO_PIN1, INPUT);
  pinMode(TRIG_PIN2, OUTPUT);
  pinMode(ECHO_PIN2, INPUT);
}

void loop() {
  Serial.println("Dry Wet Waste Segregator");

  // Measure soil sensor and ultrasonic distance for segregator
  int soil = 0;
  for (int i = 0; i < 2; i++) {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(7);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    delayMicroseconds(10);
```

```
    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.034 / 2 + distance;
    delay(10);
  }
  distance = distance / 2;

  if (distance < Ultra_Distance && distance > 1) {
    delay(1000);
    for (int i = 0; i < 3; i++) {
      soil = analogRead(potPin);
      soil = constrain(soil, 485, 1023);
      fsoil = (map(soil, 485, 1023, 100, 0)) + fsoil;
      delay(75);
    }
    fsoil = fsoil / 3;

    Serial.print("Humidity: ");
    Serial.print(fsoil);
    Serial.print("%    Distance: ");
    Serial.print(distance);
    Serial.print(" cm");
    if (fsoil > maxDryValue) {
      delay(1000);
      Serial.println("\nGarbage Detected! WET");
      servo1.write(180);
      delay(3000);
    } else {
      delay(1000);
      Serial.println("\nGarbage Detected! DRY");
      servo1.write(0);
      delay(3000);
    }
    servo1.write(90);
  }
  distance = 0;
  fsoil = 0;

  // Measure additional ultrasonic distances
  float distance1 = measureDistance(TRIG_PIN1, ECHO_PIN1);
  float distance2 = measureDistance(TRIG_PIN2, ECHO_PIN2);

  // Calculate percentages
  float percentage3 = (distance1 * 100) / 35
  ;
  float percentage4 = (distance2 * 100) / 35;

  sendval1 = String(distance1); // Convert distance1 to a string
  sendval2 = String(distance2); // Convert distance2 to a string
   sendval3 = String(percentage3, 2); // Convert percentage3 to a string
with 2 decimal places
   sendval4 = String(percentage4, 2); // Convert percentage4 to a string
with 2 decimal places

  // Display measurements
```

```
   Serial.printf("Wet Sensor - Measured Distance: %.2f cm, Percentage:
%.2f%%\n", distance1, percentage3);
   Serial.printf("Dry Sensor - Measured Distance: %.2f cm, Percentage:
%.2f%%\n", distance2, percentage4);

  // Prepare and send POST data
    postData = "sensor1=" + sendval1 + "&sensor2=" + sendval2 +
"&percentage3=" + sendval3 + "&percentage4=" + sendval4;
  Serial.print("Sending data: ");
  Serial.println(postData);

  WiFiClient client;
  HTTPClient http;
  http.begin(client, serverUrl);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  int httpCode = http.POST(postData);

  // Handle server response
  if (httpCode == HTTP_CODE_OK) {
    String response = http.getString();
    response.replace("<br>", "\n");
    Serial.println("Server response: ");
    Serial.println(response);
  } else {
    Serial.print("HTTP POST request failed with error code: ");
    Serial.println(httpCode);
  }
  http.end();

  delay(5000); // Wait 10 seconds before the next iteration
}
```

CHAPTER V

UNIT TESTING

5.1 Testing

5.1.1 Purpose and Scope

a. Purpose: Ensure that the Automated Trash Can system functions as intended by verifying all hardware and software components meet the functional and non-functional requirements.
b. Scope: Testing will cover both hardware and software functionalities, focusing on:
    - Separation of wet and dry trash.
    - Full-bin detection.
    - Real-time notifications sent to the web interface.
    - Accurate and reliable data collection.
c. Target Audience: Development team, stakeholders, and QA testers.

5.1.2 Objectives

a. Verify that the system operates as per the design specifications.
b. Identify and resolve bugs or malfunctions during testing phases.
c. Enhance overall system reliability through iterative testing and feedback.

5.1.3 Features to be Tested

a. Hardware
    - Moisture sensor for detecting wet and dry trash.
    - Ultrasonic sensor for full-bin detection.
    - Servo motor for sorting trash.
b. Software
    - Data processing and integration with hardware.
    - Notification system for web alerts.
    - Real-time data display on the web interface.
c. Performance
    - Latency in notification delivery.
    - Sensor response time.

5.1.4 Test Approach

a. Unit Testing for individual hardware components (sensors and motor).
b. Integration Testing for combined hardware and software modules.
c. System Testing for end-to-end functionality verification.
d. Performance Testing to measure latency response on different conditions.

5.1.5 Test Environment

a. Hardware
   - ESP32
   - Moisture sensor
   - Ultrasonic sensor
   - Servo motor

b. Software
   - NeonDB
   - Visual Code Studio
   - Arduino IDE

c. Testing Tools
   - Serial Monitor for debugging
   - Postman for API testing
   - Browser for UI tests

5.1.6 Entry and Exit Criteria

a. Entry Criteria
   - All development tasks for a specific module are completed.
   - Necessary hardware and software components are set up and configured.

b. Exit Criteria
   - All test cases pass without critical bugs.
   - Bug resolution rate exceeds 95%.
   - Test reports are completed and approved by the QA team.

5.2 Deliverables

5.2.1 Test Case Document

No	Test Case	Steps	Expected Result	Status (PASS/FAIL)
1	Moisture sensor detection	Connect the sensor to ESP32 and input both wet and dry materials. Check the serial output.	Correct identification of trash type (wet/dry).	PASS
2	Full-bin detection	Fill the bin to capacity and check the ultrasonic sensor's output.	Notification "Bin Full" is sent to the web.	PASS

3	Servo motor sorting	Trigger sorting for wet and dry trash.	Notification "Bin Full" is sent to the web.	PASS
4	Web interface monitoring	Access the web interface and verify the displayed data.	Notification "Bin Full" is sent to the web.	PASS
5	Admin Account Registration	Register account using postman	Notification “Admin registered successfully” in postmane	PASS
6	Admin Login	Access the web and login to admin account	Admin is successfully logged in.	PASS
7	Add Trash Bin	Access the web, login into admin account, go to “Data Management” and add new trash bin	New trash bin added to trash bin list	PASS

Table 5.1 Test Case Documentation

5.2.2 Test Proof Documentation

1. Moisture sensor detection

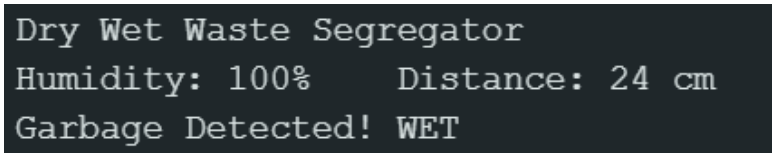

Figure 5.1 Sensor Detecting Wet Trash

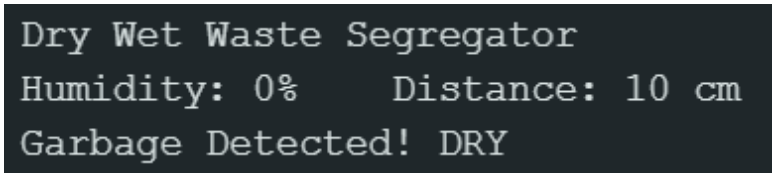

Figure 5.2 Sensor Detecting Dry Trash

2. Full-bin detection

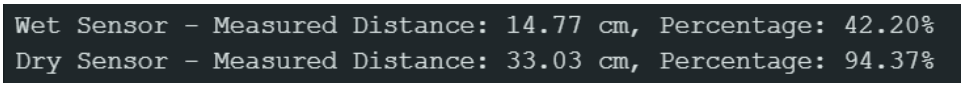

Figure 5.3 Dry Bin is Almost Full

3. Servo motor sorting

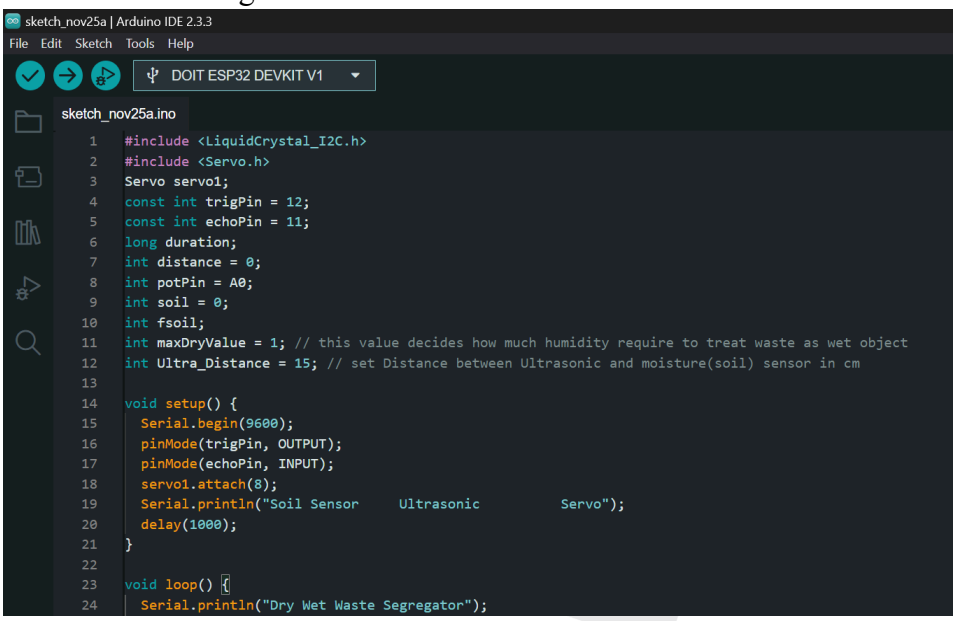

Figure 5.4 Servo Motor Sorting

4. Web interface monitoring

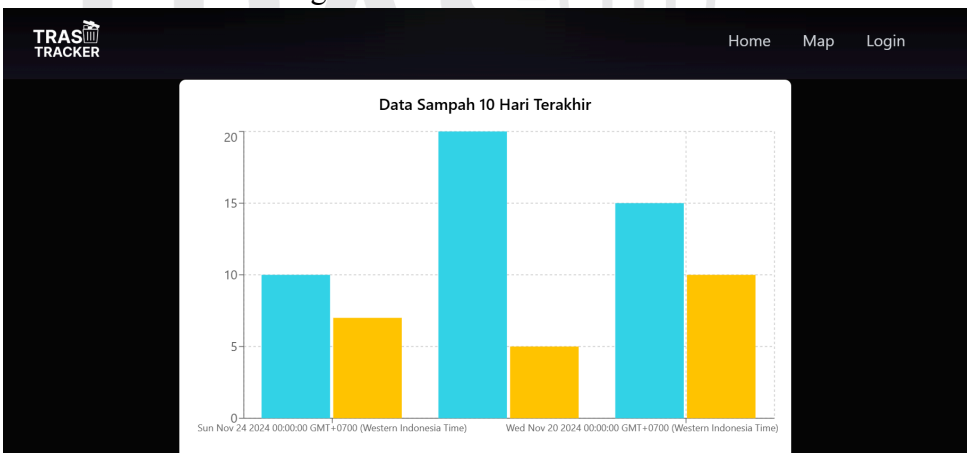

Figure 5.5 Web Interface Monitoring

5. Admin account registration

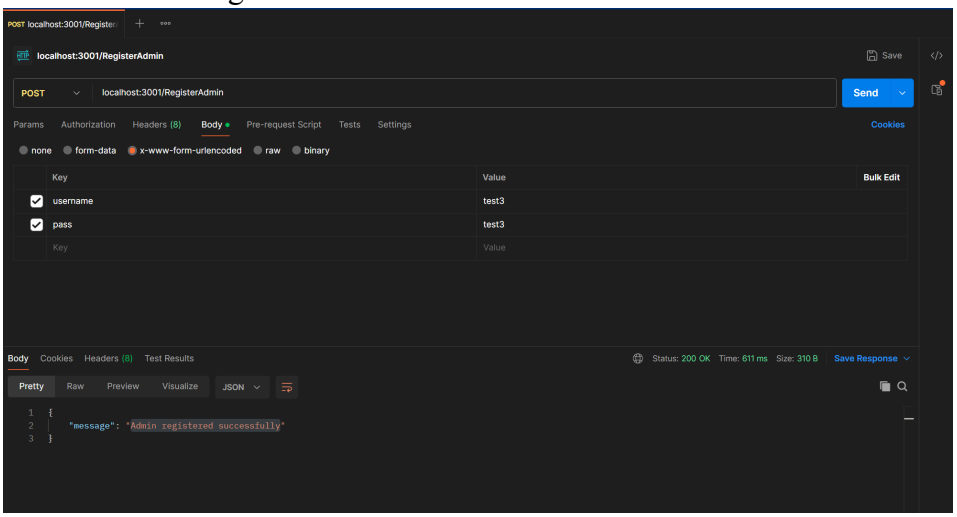

Figure 5.6 Admin Account Registration

6. Admin login

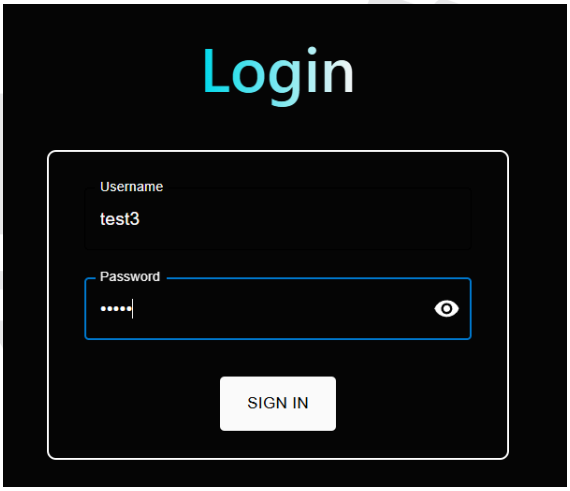

Figure 5.7 Admin Login

Figure 5.8 Admin Successfully Login

7. Add trash bin

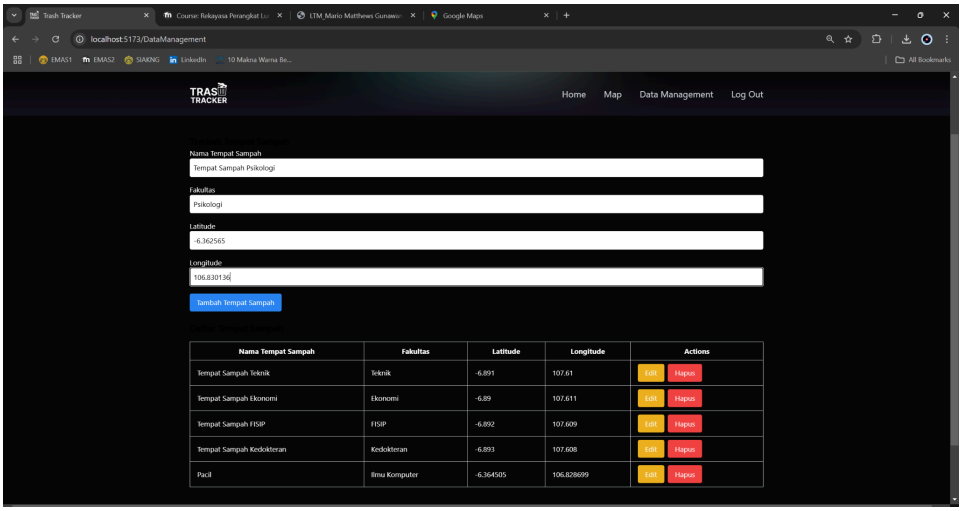

Figure 5.9 Add Trash Bin Part 1

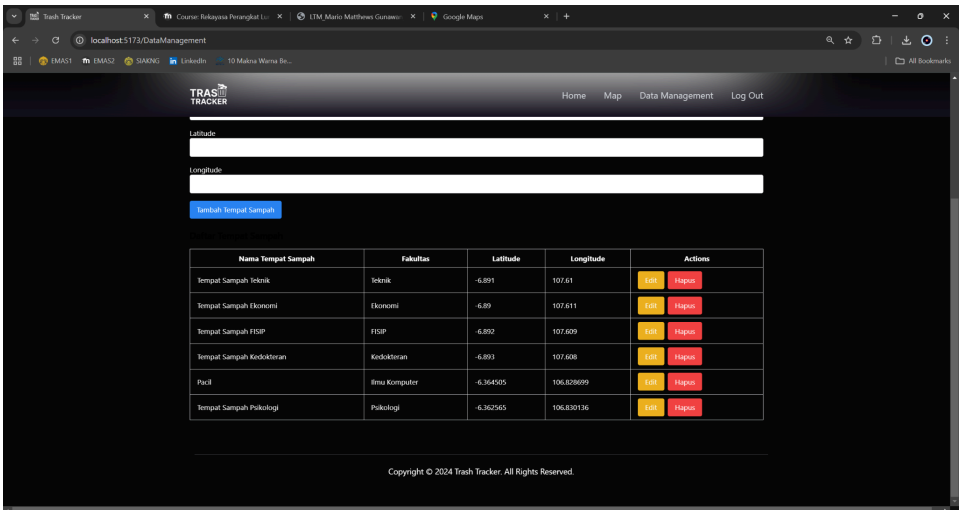

Figure 5.10 Add Trash Bin Part 2

5.2.3 Test Report

- Date: November 24, 2024
- Summary
  - Total test cases: 7
  - Passed: 4
  - Failed: 3
  - Identified bugs: 3

- Observations
    - Calibration of the moisture sensor needs improvement.
    - Notification delays occur intermittently under poor Wi-Fi conditions.
    - Integration between the IoT system and the web platform remains incomplete.

5.2.4 Test Execution Details

Phase	Result	Comments
Unit Testing	Passed (80%)	The hardware has been fully assembled. However, minor issues with the servo motor were resolved during testing.
Integration Testing	Passed (100%)	The website with both front-end and back-end is already completed, and the hardware is ready and can be connected to the hardware database. However, the integration between the website and the hardware database has not yet been successfully achieved.
Performance Testing	Passed (75%)	Latency of 2-5 seconds detected during high network traffic.
System Testing	Passed (100%)	Awaiting results for scenarios involving mixed trash composition.

Table 5.2 Test Execution Details

5.2.5 Testing Form

User Demographic

1. What is your age group?
    - < 18
    - 18 - 24
    - 25 - 34
    - 35 - 44
    - ≥ 45
2. What is your role?
    - Sanitation worker
    - Lecturer
    - Student
    - General public
    - Other: _____

User Experience

3. How easy was it to understand how to use the trash bin?
    - Very easy
    - Easy
    - Difficult
    - Very Difficult
4. How easy was it to understand how to use the application?
    - Very easy
    - Easy
    - Difficult
    - Very Difficult
5. Did you experience any difficulty finding any information about the trash?
    - Not at all
    - Occasionally
    - Frequently
    - Always

Application Features

6. Is the displayed information on the amount of waste (dry and wet) clear enough?
    - Very clear
    - Clear enough
    - Not very clear
    - Not clear at all
7. How do you find the chart comparing dry and wet waste amounts?
    - Very informative
    - Informative enough
    - Not very informative
    - Not informative at all

Application performance

8. How would you rate the application's speed in loading data?
    - Very fast
    - Fast enough
    - Slow
    - Very slow
9. Is the data displayed in the application always accurate?
    - Yes
    - Sometimes inaccurate
    - Often inaccurate

Hardware performance

10. How accurate is the humidity sensor in identifying wet and dry waste?
    - Very accurate
    - Accurate enough
    - Occasionally inaccurate
    - Often inaccurate
11. Does the servo mechanism open the trash bin smoothly and at the right time?
    - Always smooth and timely
    - Sometimes smooth and timely
    - Occasionally problematic
    - Often problematic
12. Does the servo mechanism open the trash bin to the correct position?
    - Always correct
    - Sometimes correct
    - Occasionally problematic
    - Often problematic
13. Have you experienced any issues with the hardware components (humidity sensor, ultrasonic sensor, or servo)? If yes, please describe.
    ____(Open-ended answer)

Design and User Interface

14. Do you feel comfortable with the design of this application?
    - Very comfortable
    - Comfortable enough
    - Neutral
    - Not comfortable
15. Do you find the design of this application attractive?
    - Very attractive
    - Attractive enough
    - Neutral
    - Not attractive

Satisfaction

16. Does this application help you monitor the amount of waste in trash bins?
    - Very helpful
    - Helpful enough
    - Not helpful

5.2.6 Testing Results

Tester	Score	Feedback
Irfan Yusuf Khaerullah	8.5/10	Very helpful and efficient for waste management.
Daniel Niko Madjaja	8.8/10	Easy to use, but the interface could be more visually appealing.
Farhan Nuzul Noufendri	8.7/10	Real-time monitoring is excellent and practical.
Edgrant Henderson Suryajaya	8.9/10	Informative and accurate, but the design could be slightly improved.
Mario Matthews Gunawan	8.8/10	The servo mechanism works great and updates are consistent.
Louis Benedict Archie	9.5/10	Very efficient and easy to understand for all users.
Darren Nathanael Boentara	8.7/10	Great design and functionality, very practical.
Christopher Satya Fredella Balakosa	8.6/10	Accurate data and smooth hardware performance.
Christopher Sutandar	8.8/10	The app is very intuitive and user-friendly.
Aulia Anugrah Aziz	8.9/10	The chart's colors could be improved, but overall it's great.

Table 5.3 Testing Results

5.2.7 Client Test

Figure 5.11 Client Test

CHAPTER VI

USER MANUAL

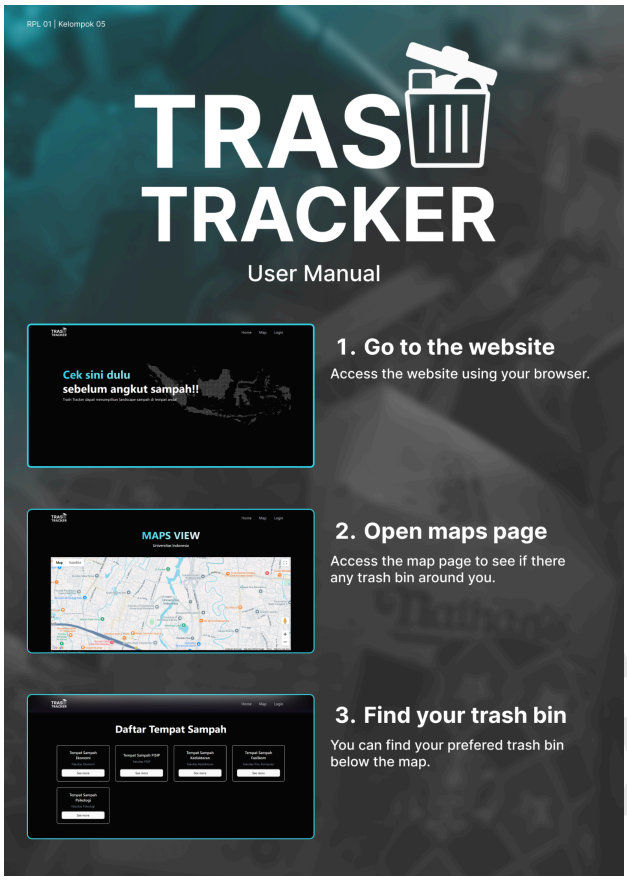

Figure 6.1 User Manual Part 1

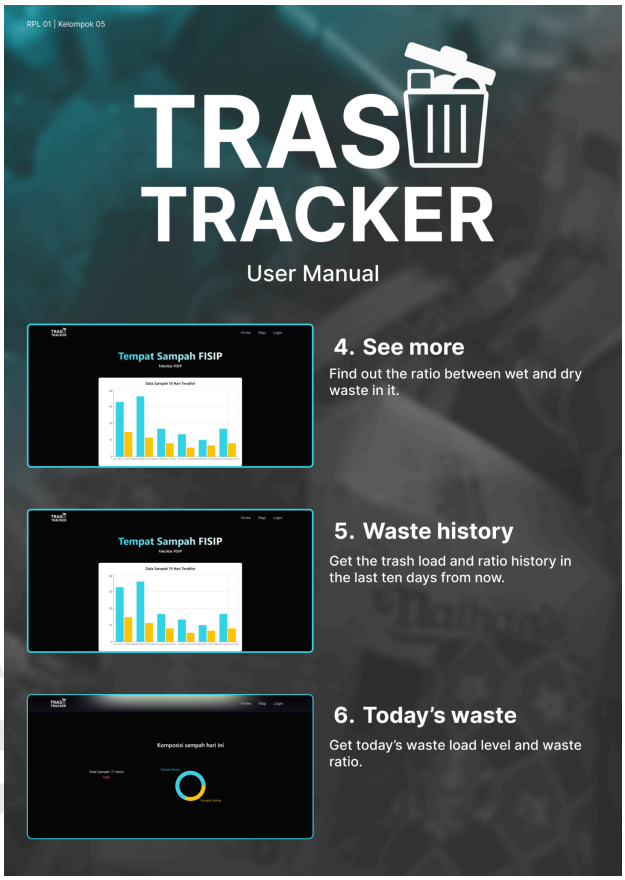

Figure 6.2 User Manual Part 2

REFERENCE

(n.d.) *Codesandbox.io*, 2024, Retrieved October 12, 2024, from https://codesandbox.io/p/sandbox/github/tannerlinsley/react-charts/tree/beta/examples/simple?file=%2Freport.20200813.175012.49872.0.001.json&from-embed

"PHP MySQL Create Database." (n.d.) *Www.w3schools.com*, Retrieved November 5, 2024, from https://www.w3schools.com/php/php_mysql_create.asp

(n.d.) "React Tutorial." *W3schools.com*, 2020, Retrieved November 5, 2024, from https://www.w3schools.com/REACT/DEFAULT.ASP

(2023, December 15). YouTube. Retrieved November 18, 2024, from https://youtu.be/MXFFoh1oZhE?si=LIjx1g_--AtqHDPg

LETTER OF AGREEMENT
SOFTWARE DEVELOPMENT AGREEMENT
FOR THE DEVELOPMENT OF "TRASH TRACKER"

On this day, October 6th, 2024, the undersigned parties:

1. **Name** : Irfan Yusuf Khaerullah
   Student ID (NPM) : 2206813290
   Major studies : Computer Engineering

Hereinafter referred to as the **First Party**.

2. **Name** : Mario Matthews Gunawan
   Student ID (NPM) : 2206810452
   Major studies : Computer Engineering

   Name : Annisa Ardelia Setiawan
   Student ID (NPM) : 2206059471
   Major studies : Computer Engineering

   Name : Rizqi Zaidan
   Student ID (NPM) : 2206059742
   Major studies : Computer Engineering

Hereinafter referred to as the **Second Party**.

Hereby declare that we have agreed to cooperate in the development of the **Trash Tracker** under the following terms:

Section 1

SCOPE OF WORK

The Second Party is responsible for creating a website named **Trash Tracker** with the following features:

- Waste Separation: Automatically sort trash into wet and dry categories using a moisture sensor
- Full Trash Detection: Detects when the trash bin is full using ultrasonic sensors.
- Smart Notifications: Send notifications to users (via mobile/web) when the bin is full or when specific actions are needed.
- Data Display: Real-time data on waste type and bin status will be accessible through a website.
- Control System: An ESP32 microcontroller based to control the sensors and actuators, ensuring proper sorting and monitoring.
- Web Interface: Allows users for easy monitoring via website interface.

Section 2

DURATION

The website development will take place over **3 months** starting from the date this agreement is signed. The Second Party is obliged to complete the project within the agreed timeframe.

Section 3

PROJECT COST

The cost for developing the **Trash Tracker** website is **IDR 100.000,00**, to be paid in two phases:

- The first payment of 50% of the total amount upon signing this agreement.
- The second payment of 50% upon completion of the project, as verified by the First Party.

Section 4

RIGHTS AND OBLIGATIONS OF THE FIRST PARTY

- The First Party must provide all necessary information for website development, such as logos, content, data, and any required technical specifications, in a timely manner to avoid delays.
- The First Party must review and provide feedback on design and development progress within a reasonable timeframe.
- The First Party has the right to request revisions within 1 month after each project milestone is submitted. Revisions must fall within the original scope of work.
- The First Party is responsible for the approval of the final product, ensuring that all agreed specifications and features are delivered before the project is marked complete.
- Any new feature requests outside the original scope of work may require additional payments and should be negotiated separately.

Section 5

RIGHTS AND OBLIGATIONS OF THE SECOND PARTY

- The Second Party is obliged to complete the website development according to the specifications and schedule outlined in this agreement.
- The Second Party must implement revisions requested by the First Party within the agreed timeframe as long as they are within the original scope of the project.
- The Second Party is responsible for testing the website and ensuring it is free of major bugs or issues before delivering the final product.
- The Second Party is required to provide a user guide, along with basic technical training to the First Party on how to use and maintain the website. This includes details on how to manage content, monitor system status, and address basic troubleshooting.
- The Second Party must hand over all relevant project files, including the source code, design files, and any documentation necessary for the future maintenance of the website.
- The Second Party agrees to provide post-launch support for 1 month to resolve any unexpected issues or bugs that arise after the website is delivered.

Section 6

INTELLECTUAL PROPERTY RIGHTS

Upon completion of the project and receipt of the final payment, the intellectual property rights to the **Trash Tracker** website, including its code, design, and content, will be transferred to the First Party. The Second Party reserves the right to reuse non-specific code or features for other projects, excluding any proprietary elements specific to this project.

Section 7

MAINTENANCE AND SUPPORT

The Second Party agrees to provide technical support for a period of **1 month**, which includes bug fixes and basic system updates. Any additional feature requests or extended support after this period will require a new agreement or additional payments.

Section 8

CONFIDENTIALITY

Both parties agree to maintain the confidentiality of any proprietary or sensitive information exchanged during the development of the **Trash Tracker** website. Such information may not be shared with third parties without prior written consent from the other party.

Section 9

TERMINATION CLAUSE

This agreement may be terminated by either party if the other fails to fulfil their obligations as stipulated in this contract. If the agreement is terminated before completion, the Second Party retains ownership of all incomplete work, and no further payment will be required unless specified in a settlement.

Section 10

WARRANTIES AND LIABILITIES

The Second Party warrants that the website will function as intended and will be free of significant bugs or issues for a period of **1 month**. The Second Party is not liable for any indirect, incidental, or consequential damages that may arise from the use of the website, unless due to gross negligence or willful misconduct.

Section 11

FORCE MAJEURE

Neither party can be held liable for delays or failure to fulfil this agreement if such delays or failures are due to circumstances beyond their control (force majeure), such as natural disasters, fire, or unavoidable technical disruptions.

Section 12

DISPUTE RESOLUTION

If a dispute arises during the execution of this agreement, both parties agree to resolve it through mutual consultation. If no agreement is reached, the dispute will be resolved according to applicable legal procedures.

Section 13

CLOSING

This agreement is made in two copies, both of which have equal legal force. Each party receives one copy as proof of this cooperation agreement.

Thus, this agreement is made truthfully and agreed upon by both parties without any coercion from any other party.

First Party

Depok, October 6th, 2024

Irfan Yusuf Khaerullah

Second Party

Depok, October 6th, 2024

Mario Matthews Gunawan

Annisa Ardelia Setiawan

Rizqi Zaidan