Ass3-Module 5- Computer Systems (2024-2025)
Project

UNIVERSITY OF TWENTE.

Requirement Analysis Phase 1

Security by Design Checklist

Project Name: WasteWatch	**Team Members: Alexandr Gidikov, Maarten van Dort, Rein de Boer, Darius Luca, Ivan Mandev, Danil Badarev**
Team ID: 25	**Mentor(s): Dany Shalhoub, Vivan Biju**

Instructions:

A. **All the sections are mandatory.**
B. **Complete the sections in the below table and put a checkmark if you have done.**
C. **Think about your application and work on the sections accordingly.**
D. **Feel free to add extra requirements for reviewing security architecture and their countermeasures for your application, if needed.**

Sr. No.	Review Security Architecture	Put checkmark ✓ if you have completed the Review Security Architecture as suggested in the left column	Additional comments (If required)	Security Controls/Countermeasures	Put checkmark ✓ if you have completed the Security control points as suggested in the left column	Additional comments (if required)
1	**Check Trust Boundaries**, ***for example,*** if you assign a higher privilege level to someone to access a particular resource. In our case, unauthorised access (user) <- employee (maintenance worker) <- admin	✓	Trust boundaries between: • Public users • Employees • Admins	**Check the prevention criteria**, ***for example,*** if your personal information is identified by logging into an application, then you decide to disable the application by removing your personal information and logging in. This is a prevention criterion. strict input validation and sanitization on all user inputs prevents injection attacks like SQL Injection and XSS.	✓	
2	**Identify data flows**, ***for example***, if you read data from an untrusted source for your application. In our case, the untrusted source may be the	✓	Data is read from user input and from several sensors. Additionally, a single compromised trashcan may send wrong status	**Check the mitigation criteria to reduce the impact of the risk/threat for the application.** ***For example*****:** Assume you have a database of users' passwords that are stored as a hash. Two users in the database who have the same password, they'll also have the same hash value. If the attacker identifies the hash value and its associated password, he'll be able to identify all the other passwords that have the	✓	

	misaligned/broken sensor, which sends wrong data to the server.		information to the webserver.	same hash value. This risk can be mitigated by adding a randomly generated string, i.e. salt to each password in the database. Hashing and salting passwords before storing them in the database helps to avoid the risk of people being able to identify user passwords when they have access to the database, even when they know the password-hash relation of specific passwords.		
3	**Entry and Exit points of the system and its components.**	✓	Entry: sensors, website. Exit: servo's, lights, website.	**Make a data flow diagram to visualize and understand the data flow, input, output points, and trust boundary**.	✓	
4	**Write the complete architecture in the SDD template. Review and approve among yourselves and by your assigned mentor(s).**	✓		**Analyze the cost involved in implementing the security controls (if any).**	✓	
	Team members' reviewed:	Alexandr Gidikov, Yes Maarten van Dort, Yes Rein de Boer, Yes Darius Luca, Yes Ivan Mandev, Yes Danil Badarev, Yes				

Software Design Document Template

1. Introduction

The purpose of WasteWatch is to be able to remotely monitor and control trashcans. The system will be able to determine the fullness of the trashcans and automatically control the lid. Furthermore, the location and status of trashcans can be seen on a website. When authorized for the respective actions, users can also unlock the servicing lids of trashcans and modify the system information on users and trashcans.

2. Functional/Non-Functional Requirements

We updated the requirements to:

a) Include authorization: differentiation between public access, employee and admin access, and the requirements concerning their functionalities.
b) Have better priorities:
H = essential for the minimum viable product
M = we definitely want this before the end of the module
L = would be fun/useful if we had time, but not in focus at this point of the project.

The new list of requirements is as follows (underline is new):

Functional

[H] The system should be connected to a website and database to display and control information.
[H] The system should determine for each trashcan how full they are.
[H] The system should lock the servicing lid, unless it is unlocked by an authorized user using the website.
[M] The system should block the lid if the trashcan is registered full.
[M] The system should control the lid of the trashcan, i.e., open when someone is nearby, and close when they leave.
[M] The system should notify employees that a trashcan is full.
[L] The system should be able to group trashcans together and limit authorization to specific groups (Scalability)
[L] The system should have an air freshener installed to remove the unpleasant smell.
[L] The system should turn the lights on when someone opens a trashcan.

Non-Functional

[H] The system should open the lid within 2 seconds when a person is detected within 10 centimeters of the sensor.
[H] The system should determine at least 3 trash levels
[M] The system should accurately notify the employees about full trashcans, i.e., at least 90% of the times the trash needs to be taken out.
[M] The system should not open the lid by random events, such as a person passing by.
[M] The system should notify the employees and update the website within 2 minutes after the trashcan is full.
[L] The system should be energy efficient and work on a single charge for at least 2 weeks

User

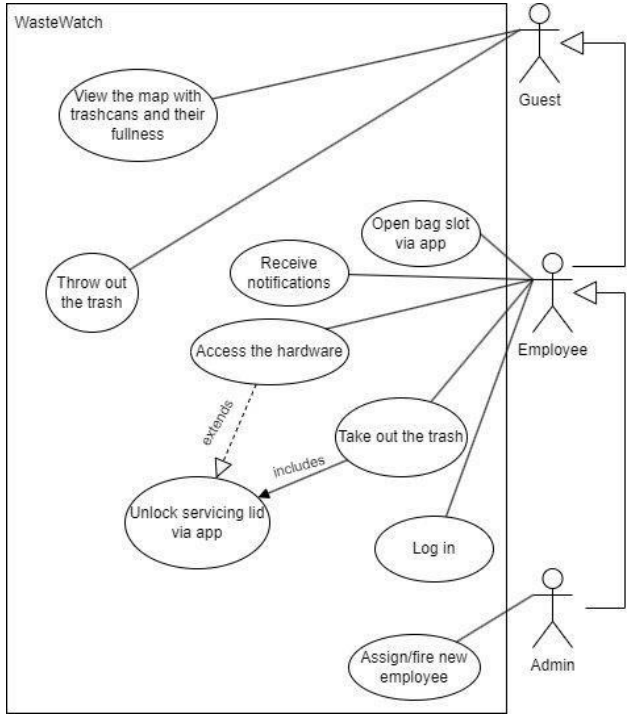
[H] The 'public user' should be able to find trashcans in their area and how full they are.
[H] The 'employee' should also be able to see statistics about the system, and should be able to (un)lock servicing lids.
[M] The 'admin' should also be able to manage authorization, i.e. which users are employees
[L] The 'admin' should also be able to limit employees' access to specific trashcan groups / buildings.

Security

[H] The system should allow user authentication on the website using secure methods.
[H] The system should differentiate in different levels of authorization, i.e., the admin (M-priority) should have more permissions than an employee, and an employee than a 'public user'.
[M] Hardware access and service should only be possible when specifically unlocked in the system.

3. Architectural Design

a) Use Case:



The first thing to notice is that we have implemented so-called permission levels, where Guest is the lowest level and Admin is the highest level. Each next level has functionalities and permissions of previous level.

Guest (implying that Employee and Admin as well) can view the map with all trashcans and their level of fullness.

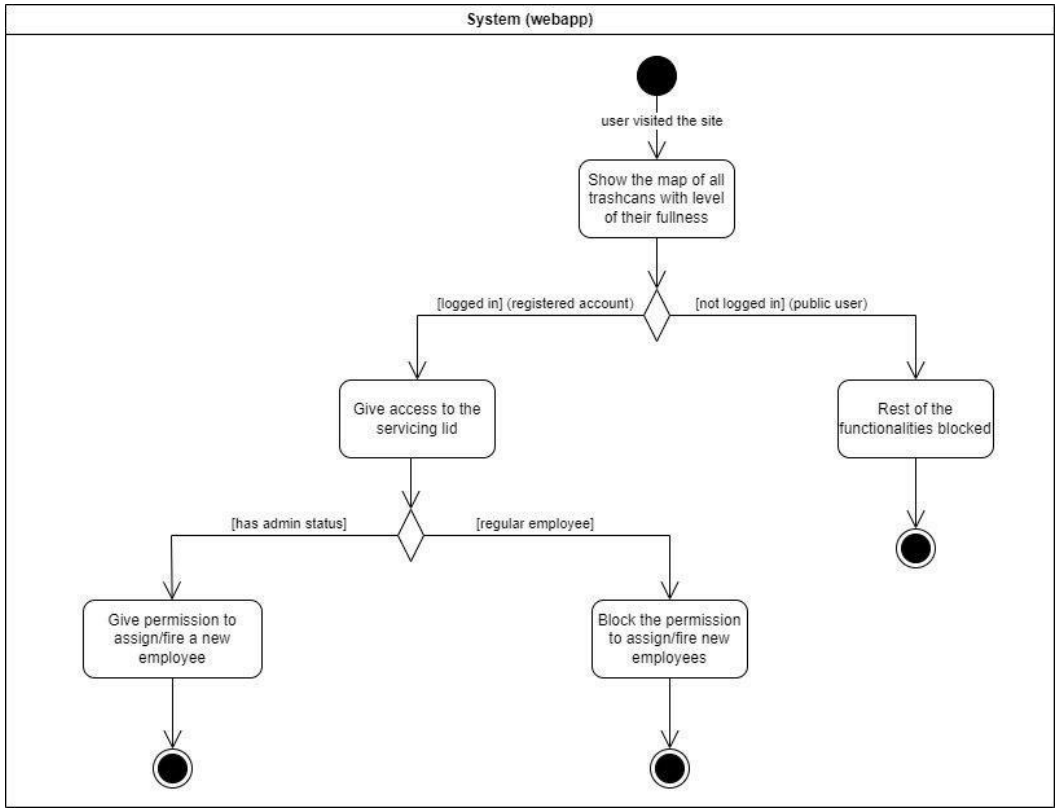
Employee's role is mainly to take out the trash. This obviously includes unlocking the servicing lid, and sometimes might include accessing the hardware, for example, if the trashcan needs technical maintenance. From this level there is also authorization needed, in order to, for instance, receive notifications (shown in detail in the activity diagram with website interaction), and do other website-related actions such as opening the servicing lid.

Admin is different from Employee only with one functionality - assign/fire employees. There is also a low-level requirement when the admin can limit the area of access for an employee, but it is very unlikely that we will have time to implement that.

b) Activity (Website interaction):

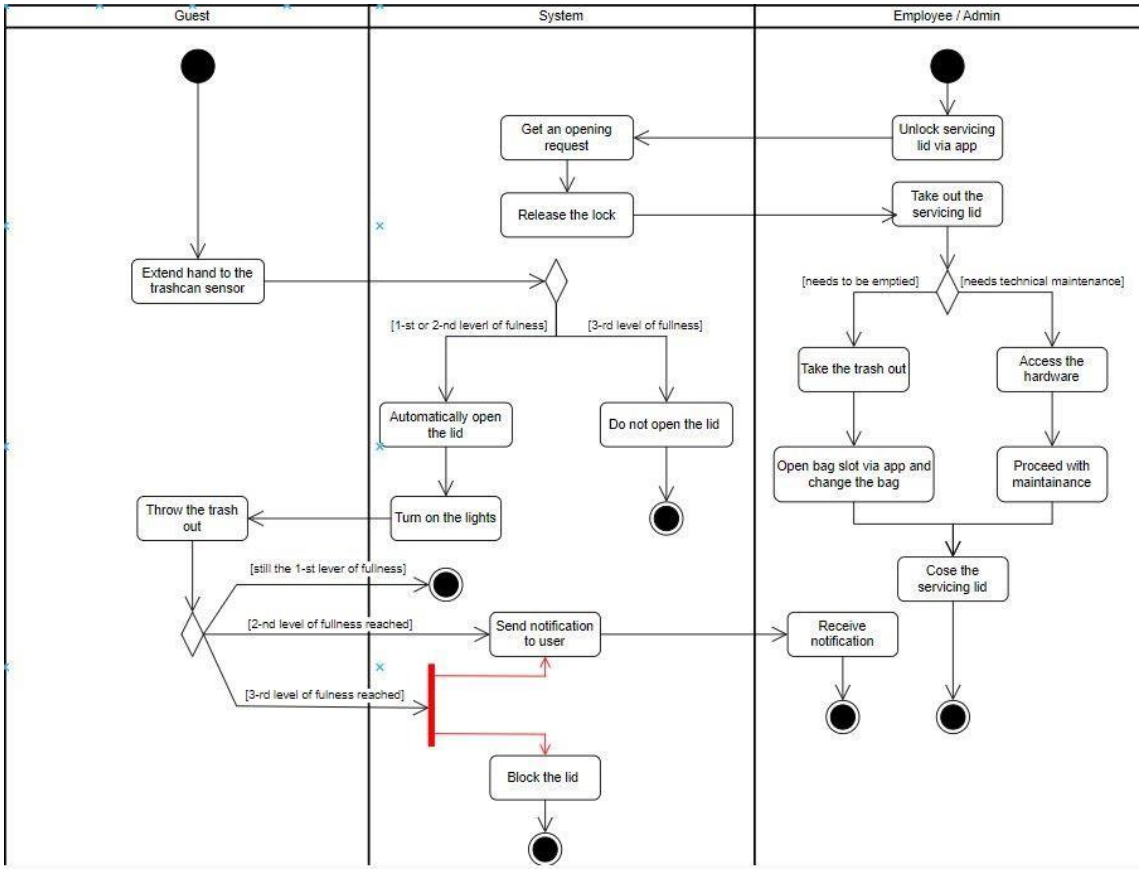
Basically, this diagram shows us the permission levels where every branch gives you more functionality based on your role.

The guest leaves the diagram after the first branch, because rest of the functionality for him is blocked, and admin gets access to the whole functionality, including: see the map of trashcans, their level of fulness, access the servicing lid and assign/fire employees.



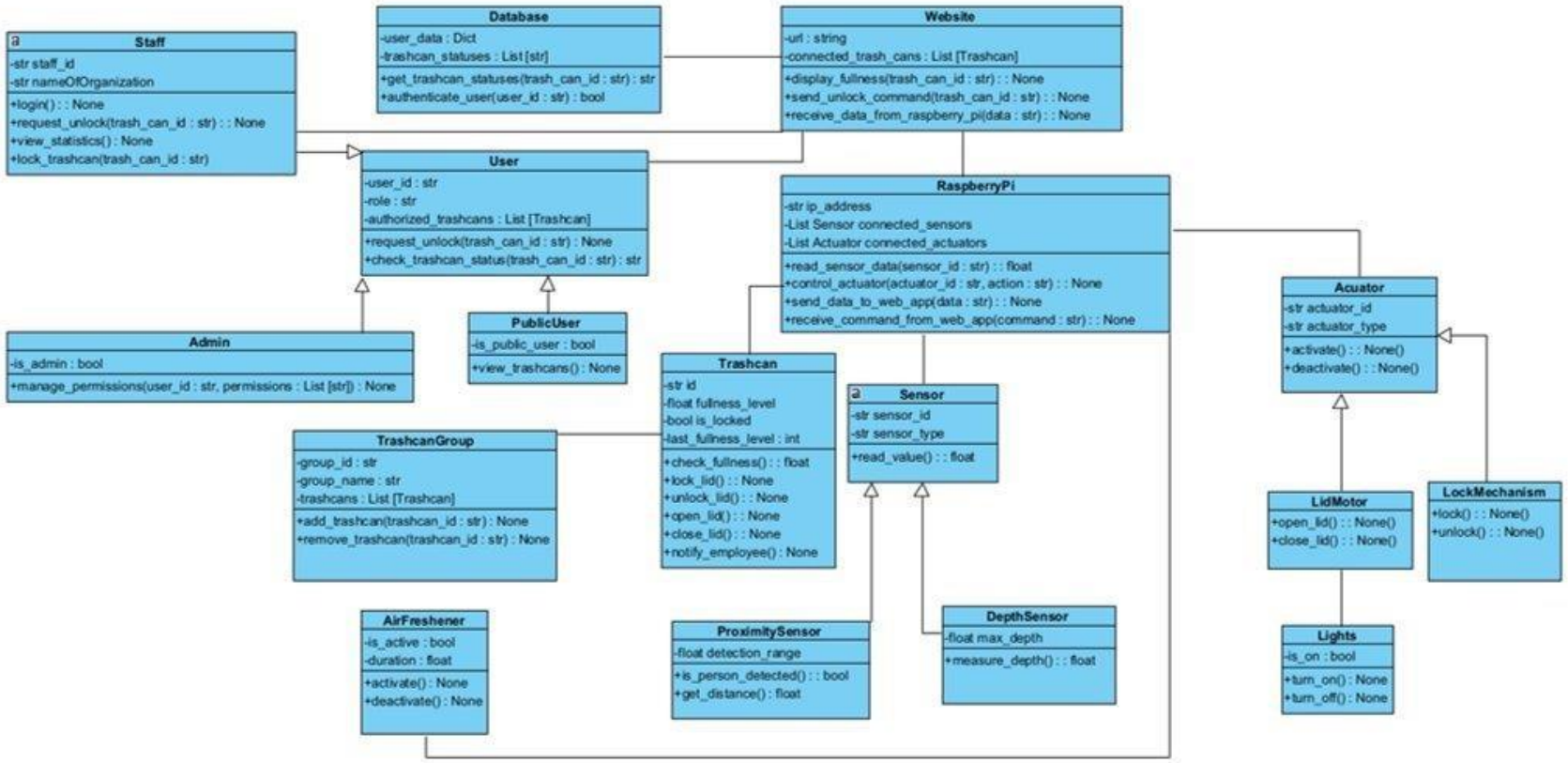
c) Activity (Real life interaction):

The diagram shows two ways to interact with the trash can: as a Guest and as an Employee. Since the difference between Employee and Admin was already shown in the previous Activity diagram, in this diagram they act the same. The guest interaction describes what happens when the person tries to throw their trash out, it shows all the possibilities, in which case the lid opens automatically, when it is blocked, when the message is sent to an employee and which of these actions happen simultaneously. The employee's part describes what happens if the person is responsible for either



d) Class:

- User Roles (Staff, Admin, PublicUser): Defines different access levels. Staff can unlock and lock the trashcan, view stats, while Admin manages permissions for users. PublicUser has limited access, mainly viewing.
- Trashcan: This class handles the core functions like fullness checks, locking/unlocking the lid, and notifying staff when the trashcan is full. Connected sensors (ProximitySensor, DepthSensor) measure distance and fullness.
- Raspberry Pi: Acts as the brain, gathering data from sensors, controlling actuators (LidMotor, LockMechanism), and sending/receiving commands from the web app.
- Website: Allows users to interact with the trashcan (check fullness, unlock lid) and display connected trashcans.
- Sensors/Actuators: The ProximitySensor detects users approaching, triggering the lid motor to open the lid, and Lights turn on. DepthSensor checks fullness, while LockMechanism locks/unlocks the lid.

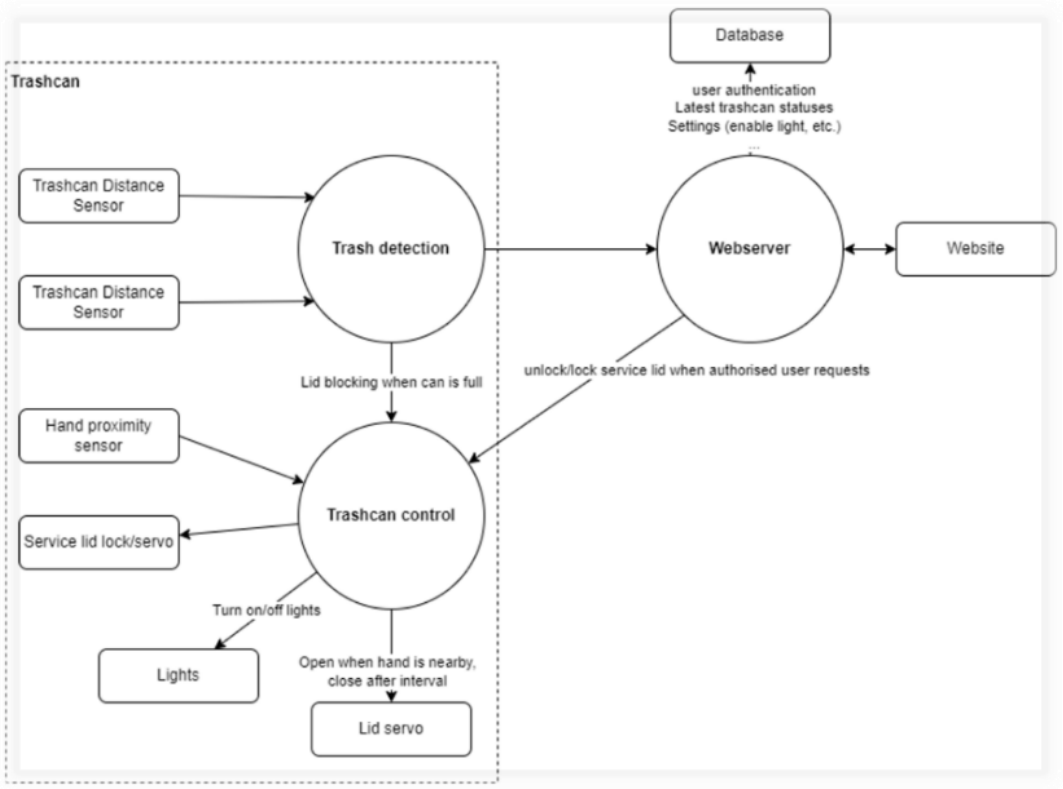


e) Data flow:

- The core of the system is the webserver process on the right. This is used to host the website, giving a remote access point to users, and is connected to the database containing information about all trashcans and users.
- Multiple trashcans can be connected to the webserver. Each trashcan holds two processes: detection and control.
    - The *trash detection* process is responsible for accurately determining the fullness of the trashcan based on the output of multiple distance sensors in the trashcan.
    - The *trashcan control* process is responsible for opening/locking the normal and service lids and lights. It also takes input from the Hand Proximity Sensor.

- All sensors are considered **entry points** that could convey faulty information. To mitigate the risk of incorrectly determining the fullness of the trashcans, multiple sensors are used.
  The website is considered the most vulnerable entry point, as it grants authorized users broad access to the system. Because of this, it is important to make sure that user input is properly sanitized and that passwords conform to minimum requirements and are stored safely.

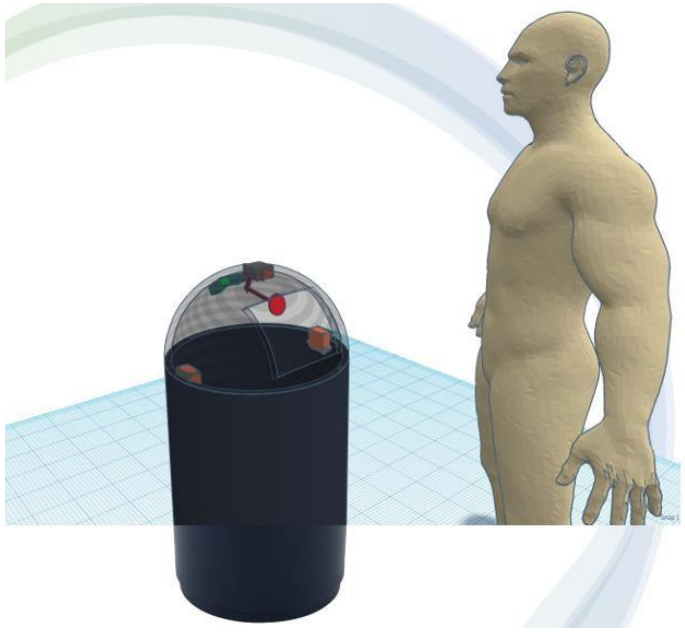
- The servo's, locks, and lights are considered **exit points**, but they do not convey any personal information and don't allow access to the rest of the system. The website is also an exit point, and is again the most vulnerable, requiring careful consideration on what is and is not shared with the users.



4. Product User Interface

To make the visual side of this project solid, we decided to completely design the front-end in Visily. Visily is a powerful front-end design tool, capable of generating associated CSS code. This allowed us to make a user friendly and intuitive user interface, and at the same time reminds us of the functionality we need to implement.
The navigation buttons on the demo are working and allows viewers to switch between pages like the final product will.

The demo is publicly available here:
https://app.visily.ai/projects/cfb02153-dedf-428a-b08c-45524b5bd1d9/boards/1122957/presenter?play-mode=Prototype



We decided to make a simple 3D model of the hardware side, to further explain how we are planning to realise the hardware side of this project.

We chose to use a bullet bin with a push lid, as it allows us to easily implement the automatic opening feature, lights servicing lid latches and other electronics without exposing them to everyday users.

The main control unit, a Raspberry Pi in this case, will be placed behind the lid opening servo. Because the lid opens inwards, this will block access to the main control unit and protect it from trash buildup.

The rim will have servo latches on the inside, allowing the service lid (the entire top cover) to be locked and unlocked.

Since all electronics will be attached to the top cover, trash will be separate from the hardware and trash bags will be placed as usual.

5. Prevention/Mitigation Criteria (Security Controls)

a) Prevention criteria

a. **Input Validation and Sanitization**: strict input validation and sanitization on all user inputs prevents injection attacks like SQL Injection and XSS.
b. **Hardware malfunction:** install extra distance sensor to cross-check readings and prevent from incorrect data.
c. **Data backup**: perform regular automated backups of the database, with an encrypted copy stored in a secure offsite location.

b) Mitigation criteria

a. **Strong password policy**: a robust password policy including rules like minimum length, combination of types of characters and timeout on multiple wrong logins will help with avoiding the risk of unauthorized actors getting access to accounts with more permissions using often-used passwords and/or brute-forcing.
b. **Password hashing and salting:** hashing and salting passwords before storing them in the database helps to avoid the risk of people being able to identify user passwords when they have access to the database, even when they know the password-hash relation of specific passwords.
c. **Permission levels**: making sure different user accounts have different permissions (such as only admins being able to edit details of trashcans, not normal users) will help to mitigate the risk of actors doing things they are not authorized to do. It also causes user input from authenticated users more trustworthy, although security controls like input validation and sanitization should still be used.

6. The cost involved (if any):

The mentioned security controls are worth almost nothing in terms of money-related cost, except for one extra distance sensor (€5.50); however, it will take a lot of man hours to implement even the smaller part of them. Permission levels is the only criteria needed for a minimum viable product, which will take ~10-20 hours. To implement all security controls, we will need much more efforts and it will take even more time

7. Conclusion:

This document gives a detailed overview of the plans and design of the system we will develop in the phases ahead. A noteworthy decision is to have several permission levels built into the system, each with their own restrictions. Another decision that is important, is to use multiple distance sensors to determine the actual trash level in a trashcan, to mitigate the risk of trashcans being wrongly marked as full. Some challenges we might potentially face are: lack of time, lack of understanding of what the final product should look like. The more likely challenge that we will face is struggling with the sensors, because not all of us had worked with them before. Nevertheless, with due effort all mentioned challenges are surmountable.

8. **Reference:**
   https://en.wikipedia.org/wiki/Secure_by_design

   https://safetyculture.com/topics/risk-mitigation/

9. **Usage of AI tools:**

Our team used ChatGPT to receive feedback on initial drafts and to brainstorm ideas for the assignment, both with regards to the assignment itself, and the diagrams. We utilized this tool to refine our concepts and enhance the clarity of our writing, but we did not copy any large sections of generated text.