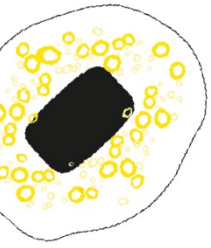


UNIVERSITY OF TWENTE.



PLAEX: A smart waste application

Design Report

Authors

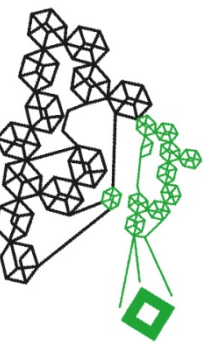
Vlad Goşa
Adrian Munteanu
Bogdan Moiş
Victor Zugravu
Ilja Kanišček

Supervisor

Dr. Le Viet Duc

Faculty of Electrical Engineering, Mathematics and Computer Science

TI Design Project Group 7



Contents

INTRODUCTION	4
DOMAIN ANALYSIS	5
2.1 GENERAL DOMAIN INFORMATION	5
2.2 INTENDED AUDIENCE	5
2.3 INTENDED USE	5
SYSTEM PROPOSAL	6
3.1 USER NEEDS	6
3.2 SOFTWARE ENVIRONMENT	6
3.3 PROPOSED IMPLEMENTATION	7
REQUIREMENT SPECIFICATION	8
4.1 REQUIREMENT ELICITATION AND DISCOVERY	8
4.2 USER STORIES	8
4.3 REQUIREMENT FORMULATION AND PRIORITIZATION	8
5.1 USER STORIES	9
5.1.1 <i>All Users</i>	9
5.1.2 <i>Disposer</i>	9
5.1.3 <i>Cleaner</i>	10
5.1.4 <i>Supervisor</i>	11
5.2 SYSTEM REQUIREMENTS.....	12
SYSTEM ARCHITECTURE	18
6.1 GLOBAL PERSPECTIVE	18
6.1.1 <i>Data Flow</i>	18
6.1.2 <i>Database and Server</i>	20
6.1.3 <i>Scenario-based vulnerability analysis</i>	20
6.2 LOCAL PERSPECTIVE.....	21
6.2.1 <i>Application Architecture</i>	21
6.2.2 <i>Monorepo</i>	23
6.2.3 <i>Feature Breakdown</i>	23
6.3 SERVER-SIDE ARCHITECTURE.....	26
6.3.1 <i>System architecture</i>	26
6.3.2 <i>Disposer</i>	27
6.3.3 <i>Cleaner</i>	29
6.3.4 <i>Supervisor</i>	29
6.3.5 <i>Scanning Feature</i>	30
6.3.6 <i>Account registration process</i>	30

6.3.7 Account deletion process:.....	30
6.3.8 Image storing.....	31
6.3.9 Endpoints Permissions	31
6.3.10 Admin Dashboard.....	31
6.3.11 System Changes.....	32
6.3.12 Bins	32
SYSTEM TESTING.....	33
7.1 TEST PLAN.....	33
7.1.1 Test Plan Approach	33
7.1.2 Software Risk Assessment	35
7.1.3 Features to Be Tested.....	37
7.1.4 Features Not to Be Tested.....	37
7.1.5 Test Pass/Fail Criteria.....	38
7.1.6 Test Schedule	41
7.2 Test Results	42
7.2.1 Unit Testing.....	42
7.2.2 Integration Testing.....	42
7.2.3 Usability Testing.....	43
7.2.4 Post-hoc Risk Analysis.....	43
REFERENCES	44
FLUTTER CLEAN ARCHITECTURE MODEL	45
STAKEHOLDER ONION MODEL.....	46
CLASS DIAGRAM.....	47
USE CASE DIAGRAM.....	48
SEQUENCE DIAGRAM DISPOSER.....	50
SEQUENCE DIAGRAM CLEANER	51
SEQUENCE DIAGRAM SUPERVISOR	52
ACTIVITY DIAGRAM TRASH DISPOSING	53
SYSTEM REQUIREMENTS.....	54
TEST SCHEDULE TABLE.....	59
MEETINGS	60

Chapter 1

Introduction

The world in which we live is growing day by day. By now, the overall population of the planet, according to the United Nations, is estimated to be around 8.1bn¹. With this ever-increasing trend in population, our society needs to find new and ingenious ways of sustaining life on Earth. One of the most important goals is not only to assure constant supply of our most trivial needs, but to also preserve the planet which hosts us. For this exact reason, the United Nations developed a list of Sustainable Development Goals², or *SDG* for short, which ensures that policies should be enforced to protect the planet and its population.

Among these goals, the consensus is that the world needs to not only improve their carbon footprint, but to also find new and innovative ways to encourage people to participate in this movement either by recycling or by recommending better consumption patterns. This is where our client, PLAEX Technologies, a company which focuses on sustainable IT services, steps in. Their goal is to develop smart waste management solutions to reduce waste-related expenses and greenhouse gas emissions, therefore positively impacting the environment.

Their solution involves three products which aim to help organizations to increase their sustainability metrics while also reducing costs. Their products range from sensors which show the fill-levels of containers all the way to smart bins which automatically identifies, classifies, and sorts the waste, thus reducing confusion and assuring correct waste management. All their products transmit data to an online dashboard which allows sustainability impact tracking and auditing.

The main goal of the design project is to create a mobile application solution which helps organizations manage and track their waste management and encourages users to recycle through gamification.

¹ Source: <https://www.worldometers.info/world-population/>

² Source: https://sdgs.un.org/#goal_section

Chapter 2

Domain Analysis

2.1 General Domain Information

The domain in which the system is introduced concerns the ever-growing need of sustainable practices within organizations and aims to provide them with tools to improve towards a better environmental resource management. The system would therefore allow organizations to track and audit their waste-related data, giving them a better oversight over their waste handling techniques. As a result, their waste management expenses could be reduced by increasing recycling rates, as well as cutting the number of cleaning personnel where needed. Moreover, organizations could use the solution as a mean to reward user's contribution to a sustainable environment through the gamification process.

2.2 Intended Audience

The system is developed as a service-based solution for enterprises that wish to improve their environmental footprint by increasing recycling yields and reducing their greenhouse gas emissions. The project aims to offer this service to public organizations such as stadiums, event halls and offices that wish to reduce cleaning costs after their events or work weeks, respectively.

2.3 Intended Use

There are several organization layers which could benefit from the implementation of such a system. Firstly, the business layer of an organization could extract viable data that could be used for machine learning purposes, such as user consumption patterns, cleaning efficiency or even attendance for an event by using the built-in sensors of the products that PLAEX offers to its customers. The system not only collects the data, but also displays key indicators that could be used by several departments inside the business layer of an organization. On a wider scale, an efficient distribution of bins would not only add business value, but also increase customer satisfaction by creating cleaner environments. Please refer to [Figure A.1](#) in Appendix A for a list of potential stakeholders within this domain.

Chapter 3

System Proposal

In this chapter, we aim to give a brief overview of user's needs by representing the three roles that will interact with the application. Moreover, we will discuss the software environment deployed by the client and what our proposed approach towards creating the aforementioned product will be.

3.1 User Needs

The system is designed for three types of users. The first type is the **supervisor**, which is assigned by the organization. The supervisor's role is to oversee the totality of the data collected and to manage the schedules of the cleaning staff. The second type is the **cleaner**, which is an employee of the organization and therefore will use the system to locate the bins that need to be emptied and to record his actions such as emptying a bin. The third type is the **disposer**, which unlike the other two roles doesn't necessarily belong to the organization. This last type of user will interact with the app not only to keep track of his waste disposal, but also to earn points from the organization which uses the system. These points would represent the way organizations reward users for their contribution to a sustainable environment.

3.2 Software Environment

PLAEX uses a web application in form of a dashboard where their users can monitor the data gathered from their products (e.g. in case of the 'Garby' product, which is a smart bin that automatically sorts waste, the dashboard shows, among others, what type of waste was detected in the bin). The current framework used for the dashboard that manages PLAEX's products is created in Django, a web framework built on top of the Python programming language. The database they are using is PostgreSQL, a well-known SQL-based DB. Since the client's preferences clearly mention the need of a mobile application which is supported on both Android and iOS, our system must be developed using either two separate codebases, each for one operating system, or a cross-platform compatible UI framework such as Flutter. Our choice is to use the latter framework since all requirements can be fulfilled using this framework and it will allow the transition of the application to a different platform with minimal effort. More detailed information about the technologies used for the system can be found in [chapter 6](#). Due to the short amount of time that is allocated to the development of the product, our focus is to deliver a fully functional android application written in Flutter, with a well-defined design pattern which allows easy integration of iOS specific interactions in the future.

3.3 Proposed Implementation

Our proposed implementation of the system is to develop three mobile applications, one for each user role mentioned in section 3.1, that will interact with an extended version of the existing backend server. The detailed overview of the system design, development process and post-hoc discussion will be discussed in the upcoming chapters.

Chapter 4

Requirement Specification

In this chapter, the process of requirements elicitation, formulation and analysis is formally discussed. According to the Agile methodology, requirements are first elicited from the client using various elicitation techniques, after which a set of user stories are composed. From the composed user stories, more quality requirements are developed to define the system's functionalities.

4.1 Requirement Elicitation and Discovery

Requirement elicitation represents the building block of every agile-based project. Using a multitude of techniques, the development team can elicit and formulate requirements that represent the functionalities that the final system must support.

Our approach for elicitation is composed of a combination of interviews with the representatives of PLAEX, as well as creating a Hi-Fi prototype using [Figma](#) to allow visual inspection of the proposed ideas. Moreover, since the client demands novel and creative designs for their application, brainstorming and cross-referencing comparable applications also enables us to discover new requirements that our client might omit during the meetings.

4.2 User Stories

User stories represent a major component of the Agile principle due to their informal but effective approach of describing user needs. All the user stories in this report are formulated based on the commonly used *Connextra template* (Lucassen, 2016) which facilitates easy conversion to system requirements.

4.3 Requirement Formulation and Prioritization

Based on the user stories formulated in accordance with our stakeholders, system requirements are created according to the S.M.A.R.T model (Doran, 1981) which defines that all requirements must be specific, measurable, achievable, relevant and time bound. Using this framework results in clearly defined requirements that are essential not only for precise iterative evaluation of the system, but also for setting clear goals for the development process.

Prioritization is an essential step in any agile-context due to time and budget constraints. In this design project's case, prioritized requirements are necessary to achieve the best balance between quality software and client satisfaction. The framework of choice for this task is MoSCoW (Clegg & Barker, 1994), which defines that requirements are to be assigned to one of the following categories: Must, Should, Could and Won't. A clear definition of each category is offered in section 5.2.

Chapter 5

Requirement Formulation and Analysis

In this chapter, user stories and quality functional and non-functional requirements are formulated and analyzed in accordance with the specifications set in the Requirement Specification section.

5.1 User Stories

5.1.1 All Users

1. As a user, I want to access PLAEX's webshop from the mobile application.

This user story was proposed by PLAEX in the first meeting of this project, but it was soon agreed that it will not be implemented because firstly, this webshop does not exist yet, and secondly, because the time does not allow such broad requirements to be designed and implemented.

2. As a user, I want to navigate inside buildings to a bin where GPS signal is not available.

Analogous to the prior user story, this required extra hardware and embedded development that is not directly linked to our project and scope. Therefore, we agreed upon a GPS-based implementation as an alternative. Of course, the disadvantage is that GPS can be unreliable in high density environments.

3. As a user, I want to authenticate as one of the three available roles.

Each application is destined for one role, disposer, cleaner or supervisor. Each type of user must be able to access his specific application by logging in. To be noted that supervisors and cleaners cannot create their accounts manually, and they specifically have to be created through the admin dashboard. Disposers can sign-up from their application.

4. As a user, I want to see the locations of each bin on a map.

We have provided feedback to our client that users could find useful the ability to navigate to specific bins via a map incorporated into the applications. Each bin will be shown on the map containing the fill levels of each stream, along with its identifier and specific location. Moreover, users will receive instructions on how to navigate to that bin.

5.1.2 Disposer

5. As disposer I want to receive points when I dispose of trash.

PLAEX's idea is to offer organizations that implement a smart bin network the ability to reward users for using their bins. This process of gamification awards points to disposers based on the quantity of waste they dispose. Of course, organizations that wish not to participate can opt out of this feature and not give any points.

6. As disposer I want to see my disposal history.
When a disposer throws waste using the application, that information should be stored in the form of a timestamp, location, type of waste and the number of points awarded. This data will then be shown on a dedicated history page on the disposer application.
7. As disposer I want to see my CO2 footprint and savings from using the system.
To help the user understand the importance of recycling, PLAEX specified the requirement of showing the user the amount of carbon emissions saved by using the smart bin. This data will be shown in the application in the form of a chart that contains the emissions with and without using the bin, and the cost savings calculated by dividing the two values.
8. As disposer I want to see the total amount of points, I received from disposing trash.
Every point awarded to a user will be visible on the disposer application on a screen that displays the ranking of points for every participating organization whose bins were used by the user.
9. As disposer I want to scan a QR code to identify in which location I dispose of trash.
Disposers can scan a QR code which has an endpoint and bin identifier embedded in it. After the scan, a 60 second period will start in which the user can throw waste into the bin. After the bin synchronized its waste data, the user can see the awarded points on the application.

5.1.3 Cleaner

10. As a cleaner, I want to see what bins need to be emptied.
This user story was first thought of as a notification system that would push a notification to a cleaner whenever a bin needs to be emptied. Due to lack of time, we only implemented a filtering system which shows the fullest bins on the top of the "Bin Status" screen that cleaners interact with on their application.
11. As a cleaner, I want to see the quickest path towards the bins that need to be emptied.
Due to time constraints and complexity, we decided to migrate the functionality of routing on the map to the Google Maps / Apple Maps applications. Whenever a cleaner wants to navigate to a bin, they can press the pin of that particular bin on the map, and then the "directions" button that is shown on the info window. The application will then redirect the user directly to Google maps / Apple maps (depending on the operating system) at the exact location of the bin (longitude, latitude).
12. As a cleaner, I want to be assigned to different companies.
Cleaners are usually employees of a cleaning company. If the cleaning company has contracts with multiple organizations that have smart bins, then the cleaner should be able to interact with the bins of all those organizations.
13. As a cleaner, I want to see the location and timestamp of the bins that I emptied.
Cleaners will be able to see on the application the location, timestamp, bin identifier and type of waste compartment that was cleaned on a dedicated history screen.

14. As a cleaner, I want to scan a QR code to manually confirm the emptying of a bin.
Cleaners can scan a QR code which has an endpoint and bin identifier embedded in. A record will be saved in the database specifying the time, location, cleaner, bin identifier and the compartments of the bin which have been emptied.
15. As a cleaner, I want to indicate what streams have been emptied manually.
The application will open a modal after a QR code was scanned which will allow the cleaner to manually confirm the emptied compartments (referred to as streams).
16. As a cleaner, I want to see an overview of fill levels for each bin.
Cleaners must have an indication of the fill levels of each bin that they are responsible for. The application will therefore show the fill levels for each particular compartment (stream) inside the bins.
17. As a cleaner, I want to see the streams I have emptied at a certain bin.
The application will have a dedicated screen that will display the timestamp, location, emptied streams, and bin identifier of each cleaning action performed by the user which queries the data.

5.1.4 Supervisor

18. As a supervisor, I want to monitor the activity of all cleaners of my organization.
PLAEX envisions supervisors as the managers of the smart bin infrastructure for a company. They should have access to data such as cleaner activity and waste analytics such as how much waste was thrown in a particular bin. However, the application is meant to serve as a quick way for a supervisor to check relevant data. More information about the state of the smart bin network can be found on a dedicated web dashboard.
19. As a supervisor, I want to see the amount and type of waste collected in each bin.
Supervisors will have a dedicated screen which shows the amount of waste collected in each bin for multiple time ranges. The type of waste collected will not be shown on the chart after rounds of discussions with the client, since it will make sense to show this data on the web dashboard instead.
20. As a supervisor, I want to see an overview of the fill levels for each bin.
Supervisors will have a dedicated screen called "Bin Status" which will display the fill levels of each bin in the form of color-coded icons (green is empty, yellow is half full and red is full). Moreover, each bin will have a dedicated chart that will show the fill levels of each compartment of a bin.
21. As a supervisor, I want to see the CO2 footprint and cost savings of all bins for the organization.
This will not be implemented for the supervisor inside the application, but rather will be shown on the dedicated web dashboard.

22. As a supervisor, I want to see the ranking of points distributed to each user.
This user story will not be implemented due to lack of time. However, the data will be readily available in the database, therefore PLAEX could implement this later inside the application or on the web dashboard.
23. As a supervisor, I want to add devices to my organization by scanning a QR code on the respective devices.
Each bin must first be registered on the administrative dashboard by PLAEX. Afterwards, if a bin is delivered to an organization, the supervisor can scan the QR code on that bin to register it to his company.
24. As a supervisor, I want to add cleaners to the organization that I supervise.
Supervisors can request additional cleaners by contacting PLAEX. The only way in which cleaners can be assigned to the bins of an organization is to manually do it via the administrative dashboard.
25. As a supervisor, I want to assign a location for each bin of my organization.
The location of bins can be specified in longitude and latitude on the administrative dashboard. If a bin has such data, then it will be shown on the map in the form of a pin.
26. As a supervisor, I want to manage my smart bin system from an administrative page.
We realized there was a need for an administrative dashboard where supervisors can manage cleaners and smart bins inside their organization.

5.2 System Requirements

Must

The requirements labelled as *Must* are critical to the success of the product. These represent the building blocks of the overall project and therefore the failure of one such requirement will deem the product unusable. Moreover, the software is to be considered an MVP³ if it satisfies all the *Must* requirements.

1. The system must allow users to sign-up and login with their account.
Disposers can sign-up and login directly from their application. Supervisors and cleaners will receive their account directly from PLAEX. This behavior is by choice since any other implementation would imply big security risks that were too complex to be solved in such short time.
2. The system must authorize users based on their credentials.
Each user role will only have access to their own application. Supervisors can only access the supervisor app if they have valid supervisor-role credentials. The same applies to the other roles. Each application only supports one role.
3. The system must show the disposal history for each disposer.
The disposer application has a dedicated "History" screen that shows the disposal history of the user. The data is shown in the form of a list containing items with the timestamp, location, bin

³ Minimum Viable Product

identifier, type of waste thrown, and the number of points received for the action. The history screen can retrieve data from the past day, week, month, or all time.

4. The system must allow disposers to scan a QR code prior to throwing waste in a bin with the purpose of linking the waste items to them.

Disposers can scan a QR code located in the bin which has a dedicated API endpoint and a bin identifier encoded. The application validates the request and displays a 60 second window in which the user must throw their waste. After 60 seconds, or after pressing "OK", the user is greeted by a celebration. A scheduler runs on the main server which links the thrown waste to the disposer by matching the period in which the action was performed. More details can be found in chapter 6 as to why we implemented the scheduling mechanism instead of direct feedback.

5. The system must inform cleaners of which bins need to be emptied.

Originally thought of as a notification system, this requirement could not be implemented as first intended due to lack of time. Instead, we agreed with PLAEX that the best course of action is to display the fullest bins first on the "Bin Status" screen inside the cleaner's application. In this way, cleaning staff can easily check and see which bins need to be emptied first.

6. The system must allow the assignment of multiple companies to a single cleaner.

The administrative dashboard allows PLAEX to assign multiple organizations to a cleaner. This implies that the cleaner application will have access to all the bins of the organizations that are linked to the cleaner entity.

7. The system must show the bin cleaning history of each cleaner.

The system keeps track of each QR code action that a cleaner performs when a bin is emptied. The data can be retrieved on the cleaner and supervisor apps. For cleaners, they can see historical data on the dedicated "History" screen, and supervisors can see the activity for all cleaners working for their organization on the "Cleaners" screen.

8. The system must allow cleaners to register the action of emptying a bin by scanning a QR code.

Cleaners can scan a QR code located on each bin to register what streams / bin compartments have been emptied. The request is then saved on the server indicating which bin was cleaned, along with the timestamp, location and type of streams that were emptied. Currently, the server does not implement an automated way of checking the validity of these cleaning actions. This is mainly because of the high polling rate of the bins which does not allow instant checking. More details can be found in chapter 7, in the risk analysis subsection where we describe why we did not implement such a feature, and how PLAEX could solve it in the future.

9. The system must allow cleaners to select the bin compartments that they cleaned.

Each cleaner needs to manually specify what compartments they have cleaned. This is done by scanning a QR code on the bin itself, which opens up a modal inside the cleaners' application. In there, cleaners can specify each emptied stream.

10. The system must show the fill levels of each bin.

The supervisor and cleaner applications have a "Bin Status" screen which displays the fill levels of each bin. The supervisor has access to all the bins of his organization, whereas the cleaner can see the bins of all organizations he works for. Both supervisors and cleaners can see the fill levels for

each individual stream (PMD, Paper, Residual and Organic) of a bin in the form of 4 vertical bars that are filled accordingly. A green bar signifies a level under 30%, a yellow bar is between 30% - 90%, and red signifies anything over 90%.

11. The system must show each compartment per bin that was emptied, on the cleaning history of a cleaner.

Each cleaned compartment is shown in the form of an icon on each list item on the "History" screen of the cleaners app, and the "Cleaners" screen of the supervisor app. Organic is represented by a trash can icon, PMD by a leaf icon, Paper by a document icon, and Residual by a glass cup icon.

12. The system must show the cleaning history of each cleaner in the company in which the user is a supervisor.

The supervisor app has a dedicated screen named "Cleaners" which displays a calendar and a filter dropdown. The cleaning history for all cleaners can be retrieved by pressing on a date inside the calendar. The dates span for up to one year in the past.

13. The system must allow filtering based on date and cleaner names for the history preview.

When a date is selected, the cleaning history of all cleaners is shown for the selected date. Furthermore, if the supervisor only wants to see the history of one or more cleaners, he can do so by selecting their names in the filter dropdown. The calendar can retrieve data from up to a year in the past.

14. The system must implement a function that allows the retrieval of CO2 footprint and cost savings data for each bin inside the organization supervised by the user.

Initially a must requirement for the application, this requirement was left for the web dashboard in agreement with the client since they realized that a supervisor would use a laptop for this purpose.

This requirement was replaced by the following requirement.

15. The system must implement a function to show the bin waste history and increase percentage for each bin inside the organization supervised by the user.

When a bin item is tapped from the list of bins in the screen 'Bin Status', the supervisor is redirected to the bin overview screen which bears the name of the selected bin. On this screen the user can see the current fill levels of the bin followed by a chart displaying the total waste and the waste history of that bin. The user can also select from how far in the past the data is shown (the options are: 'day', 'week', 'month', 'year').

16. The system must link each thrown waste object to a user if a QR code was scanned at the time of disposing the item.

On the app-side the user scans a QR code and will see a 60 second timer with instructions regarding disposing of waste. After the synchronization of the bins with the server, a scheduler will detect the connection between the user's scan and the waste from the bin and create the link between the two. The waste thrown away will then be shown as history for the user together with the points received for that waste if any.

17. The system must link supervisors and cleaners through a company.

This requirement was introduced by the client around the middle of the implementation phase. The client realized the need of the 'company' entity which serves as parent for both supervisor and cleaner. We agreed to fit this requirement in the remaining time, thus refactoring the progress made up until then to satisfy our client's needs. We introduced the 'company' entity that now serves as a parent for both supervisor and cleaner roles and links them.

18. The system must have an administrative page managing all the entities and relations.

We realized the need for an administrative page where the admins can manage all the entities and the relationships between them. PLAEX did not have a straightforward method of managing data, e. g. adding new companies, supervisors. Therefore, we created an administrative page where an administrator can create such entities and modify them together with their relationships, rather than having to insert directly into the database.

Should

These are requirements that should be implemented, but they are not necessary for the successful delivery of the product. They are important due to the benefit of increased product value, but they do not deem the product unusable if not implemented.

19. The system should assign points to disposers for each disposed of waste.

Each company can specify whether they give points for the use of their bins. If points are given and the user uses their bins, they will receive the amount of points the company has set and see this information on their history page. If no points are given by the company, instead of showing the number of points, a suggestive text ('NP') is shown instead.

20. The system should show the CO2 footprint for each disposer and organization in the form of a chart.

After consulting with the client, we concluded that the CO2 footprint of the organization will be shown on the web app to its supervisors. The disposers can view their CO2 footprint in the 'Statistics' screen under the 'CO2 emissions' tab. They will be presented with a time selection dropdown, the total emissions, the increase percentage of the emissions in the selected time frame and a chart. The chart shows the emissions with and without the Garby product and the savings made (the difference between the previous two).

21. The system should allow supervisors to register new bins by scanning the QR code of an unregistered bin.

Bins need to first be registered in the system by PLAEX through the administrative dashboard. Once delivered to an organization, the supervisor can scan the QR code located on the bin. Once scanned, the bin automatically gets assigned to the company of the supervisor. If the bin is already registered by another company, then an error message will be displayed. If the bin is already registered with the company of the supervisor, then the bin fill level will be displayed.

22. The system should let supervisors assign a latitude and longitude location for each bin.
Currently, there is no method that allows supervisors to modify the latitude and longitude of a bin from the mobile application or web dashboard. In order to do so, they need to contact PLAEX to assign these values from the administrative dashboard.
23. The system should allow for storing user's uploaded profile images.
Every user can modify his profile image by tapping on their profile picture. A screen will show that contains an "update" button. On press, the button will allow the user to change his profile image. The images are stored in PNG format on the file system of the server. In the case that the profile image cannot be sent to the server, an error message will be shown, and the old picture will remain unchanged. If the profile image of a user cannot be retrieved, then the application will display a default user icon instead.

Could

These are requirements that could be implemented since they represent nice-to-have features for the final product but are not essential for the project.

24. The system could be divided into 3 apps for each type of user.
Originally, the system was to be designed as a single mobile application for all three roles. After rounds of discussion with our supervisor, we agreed to design the solution as three applications, one for each role. This method allows us to reduce the likelihood of unauthorized access to a specific role, increase the scalability and maintainability of the application and to lower the downtime for an application when a new update needs to be pushed. More details can be found in chapter 6.
25. The system could show a pin for each bin on a map.
Each application shares the same implementation of the map screen by implementing it from the core library. The map displays a pin for each bin that is accessible by the role of that specific user. Each pin is displayed in the form of a red circle that has the bin identifier attached to it. Whenever a user taps a pin, an info window will be displayed which shows the identifier, location, and fill levels of each stream of that bin. Moreover, the user can navigate to the bin by tapping on the "directions" button inside the info window.
26. The system could show the total number of points for each disposer.
The disposer application has a dedicated screen named "Statistics" that implements this feature. When users press on the "Points" tab, a ranking is shown which displays the total number of points achieved at each participating organization.
27. The system could show a path towards the bins that need to be emptied.
Due to limitations in the Flutter framework, we decided that the most effective way of implementing this feature is to redirect the user to its platform specific navigation app. For example, an Android user can click the "directions" button on the info window of a bin and Google Maps will be opened at the exact coordinates of that bin.

28. The system could implement a ranking system that keeps track of all points allocated to a disposer under the organization supervised by the user.

First taught as a must, this requirement was changed into a could in agreement with the client since it was overruled in importance by the other requirements. Due to this change in priorities this requirement was left to be implemented later by the client.

Won't

These requirements will not be implemented in the final product due to time constraints or the complexity of implementation. However, these requirements highlight the potential improvements that could be added to the project in the future.

29. The system won't fully integrate PLAEX's webshop implementation

PLAEX's plans are to introduce a webshop in the future to sell their products. One of the requirements listed by the customer was to link this webshop to the mobile applications. We marked this requirement as won't because this webshop is not yet implemented. Regardless of that, the design pattern of our application should allow easy integration with web-shops, or in the least a redirect to the web-based shop in the future.

30. The system won't use ultra-wide band to locate bins

This requirement would require a hardware adaptation for all the bins, and it would exceed the time limits imposed by the design project.

31. The system won't do path-tracking inside all buildings towards bins that need to be emptied.

Analogous to the last requirement, the reason behind this classification is due to the fact that this would have required extra hardware (ultra-wide band) installed on the bins. It is important to also acknowledge that this requirement came from the idea that GPS signal can be unreliable in cases of high-density events such as football games.

32. The system won't redirect anonymous user to a custom advertisement page of the PLAEX mobile application for disposers.

Since the general scope of the project was working on developing the mobile application and its server side, we did not create a separate web page affecting legacy implementation of client's web dashboard, leaving it for them for several reasons: 1) it has to be well thought out from marketing view 2) this web page has to have direct references to corresponding app stores depending on OS platforms. Hence, instead of redirecting to custom web page, anonymous users scanning QR codes are referred to plaex.net. (allowing to change it later)

[Appendix C](#) contains a detailed overview of all the user stories and the requirements that were consequently created and linked to each story. The table contains additional information such as **criteria**, which defines the acceptance criteria discussed in chapter 7 for each requirement, **performance** and **bandwidth** which define the performance requirements for each requirement to be accepted, **priority** and **source**.

Chapter 6

System Architecture

This chapter presents the architecture of the system from two major perspectives: global and local. The global overview of the system oversees the architectural design choices of the high-level system architecture where each sub-system is defined as a component with its own specific role. The local perspective delves into the details of each sub-system, its design patterns and what its role is into the well-functioning of the system.

6.1 Global Perspective

6.1.1 Data Flow

Data stands at the core of any software system; therefore, it is of utmost importance to document and analyze how data flows between sub-systems. The purpose of this section is to describe the data flow in the system using a high-level design which aims to provide scalability and modularity in the long term.

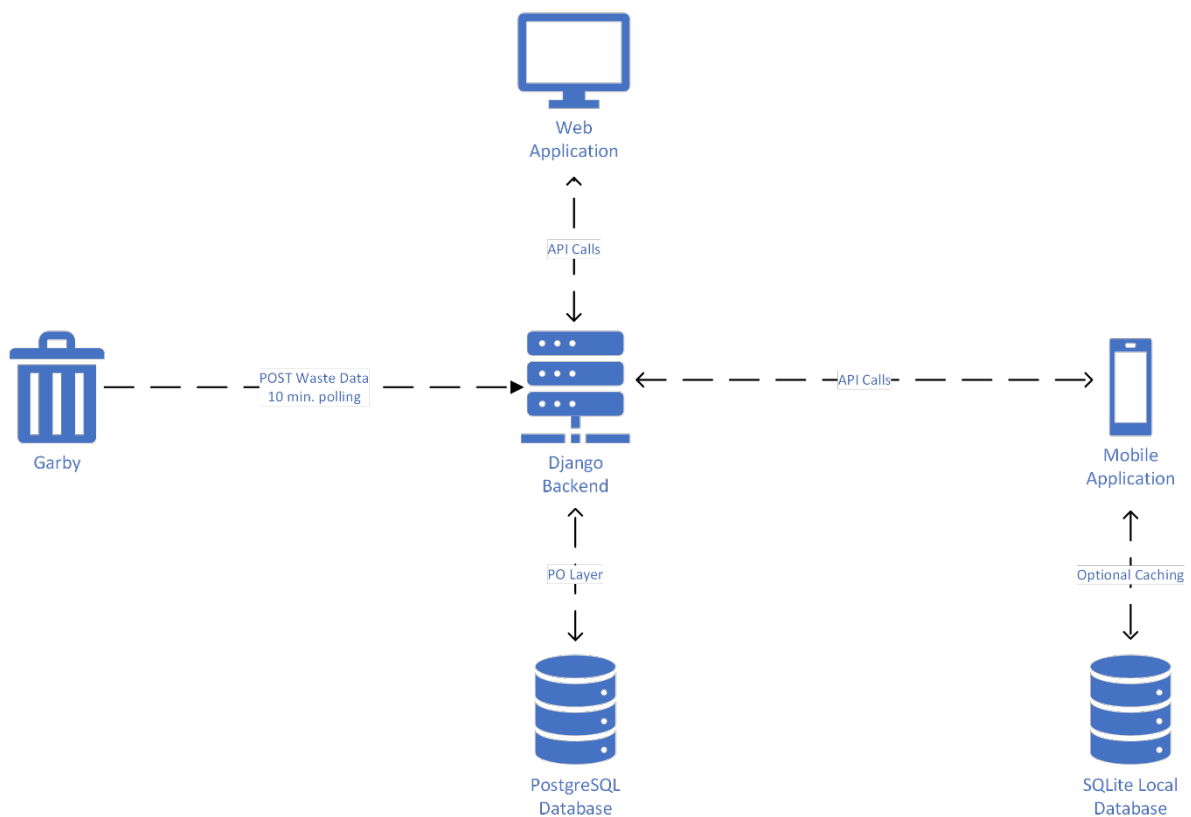


Figure 6.1 Global Data Flow Design

In figure 6.1, a high-level description of the data flow in the system is presented. The system is composed of a centralized backend system which acts as a REST API server that acts as a gateway for all the data flowing through the application. Since it uses a Django Python framework, it allows object relational mapping, meaning that the database is manipulated directly using objects in python code. This offers the advantage of abstracting from direct SQL-based queries that could easily become hard to read and implement as the system scales vertically. The backend server handles incoming requests from the “Garby” smart bins which send a POST request with new data every 10 minutes. The clients, which are represented by the web application and mobile application in this design, are considered to be the same in terms of the global system since they both interact with the backend server in a similar way. The clients request and send data through standard HTTP methods that are handled by the backend server prior to being stored or retrieved in the database, thus ensuring proper exception handling and integrity of data.

Advantages of the chosen design

The design of the system’s data flow is dictated mostly by the requirements imposed by the client, which require the system to be highly scalable, modular, and compatible with a future implementation of a microservice architecture that can be deployed in containerized applications.

The main advantage of the proposed data flow design is that it offers a loosely coupled architecture, meaning that each sub-system can be independently developed without affecting any other components of the system. The centralized backend server offers an interface-focused communication system, meaning that the database is abstracted by offering clients immutable copies of data in the form of documented entities. As long as each client sub-system defines the interfaces needed to communicate with the server, the implementation can be abstracted from.

Secondly, using only one backend implementation does offer the advantage of easier processes for deployment, testing, scalability, and development since developers will only have to modify one repository to offer functionalities to both clients, regardless of platform.

Thirdly, as previously mentioned, PLAEX envisions a microservice architecture that would serve even more clients in the future, therefore, having a centralized REST API server will significantly reduce the amount of refactoring needed to deploy this solution.

Disadvantages of the chosen design

Regardless of how many advantages this solution might bring, we cannot abstain from acknowledging the potential disadvantages that this architecture might pose. Firstly, having a centralized server handling all the clients’ requests would mean that in case of a downtime, all clients of the application will be affected by it. Secondly, this design would increase the load on the server that is hosting the database and backend deployment since more requests would have to be handled due to the increased number of clients. This means that in order to scale, PLAEX would have to either deploy the server on a more powerful container which is able to handle more requests, or to horizontally scale their backend to allow the use of parallel server instances.

6.1.2 Database and Server

High-level description

The system's backend consists of a REST API architecture which allows CRUD operations on database objects through HTTP methods. The framework used in the server allows the usage of persistent objects which can be directly manipulated in code instead of handling the requests through raw SQL queries. This approach serves to improve the security, integrity, and scalability of the global system.

Frameworks

The server running the REST API is based on a Python framework called Django⁴ which according to their developers, "encourages rapid development and clean, pragmatic design". It handles requests, security, and database management automatically, leaving them out of the developer's sight. Of course, it is a very high-level framework which could lead to potential issues when a system needs functionalities that are not supported natively by Django, but for the purposes of the deployed system, it is more than suitable.

The database that stores the entirety of the data for the system is using PostgreSQL⁵, an open-source SQL database that is well-known for its reliability and performance. There is no outside access to the database other than the connection used by the Django backend, therefore, all the requests have to go through the backend server first before reaching the database.

6.1.3 Scenario-based vulnerability analysis

Similar to how risk analysis is performed to determine the potential issues that might arise during the development or even after-deployment phases of the project, scenarios can be created to analyze and discuss potential solutions to issues that might arise using the current software architecture. We propose a scenario-based vulnerability analysis which lists each vulnerability related to the global architecture that might arise in the aforementioned phases, and the solutions that could be employed to solve these risks if they are to be encountered.

- **Database deadlocks caused by increased API request throughput**

Database deadlocks are a real risk that must be addressed in every system that has scalability as a key design element. This implies that we must take a pragmatic approach towards handling transaction isolation levels, meaning that we must choose a default level such as serializable or repeatable read which places a lock on every row affected by a transaction. Of course, by the time the application will scale up, performance will start to degrade if transactions are kept at a very restrictive isolation level.

⁴ Django Framework - <https://www.djangoproject.com/>

⁵ PostgreSQL - <https://www.postgresql.org/>

- **Framework limitations caused by high-level interfaces not offering low-level code support**

As programmers, we agree that the faster a product comes to fruition, the better. This means that the use of high-level frameworks allows quicker development times while letting programmers abstract from the details involved in data management, security, and other low-level code technicalities. This approach is suitable for a small-scale application, but as demands grow, these frameworks tend to show their limitations. A solution that we propose to this is to design everything with a containerized-focused architecture, thus avoiding the pitfalls of monolithic architectures such as ever-growing codebases that become hard to maintain. This approach would let developers abstract from the framework in use, since regardless of the server backend, as long as the communication interface remains the same, the system can abstract from the implementation.

6.2 Local Perspective

6.2.1 Application Architecture

Framework and Design Pattern

For our project we decided to go with the Flutter Clean Architecture (Martin, 2012). According to its author, this design pattern offers a set of rules which ensure that the resulting code is intrinsically testable, scalable, and modular.

The architecture consists of three main layers which form the so-called dependency chain of the application (see [Appendix A](#)). The rule states that each inner layer must be separated from its outer layers, therefore, all source code dependencies can only point inwards, not outwards. In the following paragraph, we will enumerate the three layers that compose an application inside our architecture:

```
1. feature/  
2. |-- data/  
3. |-- |-- datasources/  
4. |-- |-- |-- remote/  
5. |-- |-- |-- Local/  
6. |-- |-- models/  
7. |-- |-- repositories/  
8. |-- domain/  
9. |-- |-- entities/  
10. |-- |-- repositories/  
11. |-- |-- usecases/  
12. |-- presentation/  
13. |-- |-- pages/  
14. |-- |-- widgets/  
15. |-- |-- blocs/
```

Figure 6.2.1 Architecture Structure

Presentation Layer

The presentation layer is responsible for modelling the view and controller elements of the application. The views are represented in Flutter in the form of pages that contain multiple widgets. The controller is modelled in the form of a BLOC ⁶ which holds all the states, events and business logic that render changes on the view layers. Every action performed by a user is handled by the BLOC, which tells the pages how to render accordingly.

Domain Layer

The domain layer is responsible for holding all the model data used by the application, as well as offering an abstract view of the data querying mechanism represented in the form of a repository. Each entity represents an immutable object that is used by the presentation layer to read the current state of the database and to act accordingly. The usecases folder offers a set of abstract classes that model a specific use case in which data is retrieved. For example, the process of retrieving the profile picture of the user can be represented as a single function inside a usecase class. In this way, the data layer can be changed without affecting the domain layer used by the application.

Data Layer

The data layer offers the direct method of fetching data from either a remote database, or from a local cache. The datasources folder holds the connection to either one of the aforementioned data sources, and all classes modeled in this folder act as a singleton. The models folder holds all classes that are bridges between the data layer and the domain layer. When a request is being made, a model is responsible for parsing the received JSON schema and creating the immutable entity to be passed inwards to the presentation layer. The repositories folder holds all the implementations for the abstract repositories created in the domain layer, therefore allowing the domain layer to be fully separated from the data querying methods used in the data layer.

⁶ BLOC - <https://bloclibrary.dev/>

6.2.2 Monorepo

During the early stages of development, it was figured that even though the application must be able to support three different user roles, most of the functionalities are shared between all of them. Usually, in a web application environment, it is easier to model and render different states of an application based on authorization levels, but in a mobile application environment that must be developed on a tight schedule, it is not as feasible. Therefore, in accordance with our supervisor's proposal from the standpoint of a domain expert, we decided to split the application design into three; one application for each user type. The shared elements of the applications are developed on a separate "core" library.

Usually, different projects are developed, maintained, and released using separate repositories, but in the case of our application, the development had to be carried out on a single repository. To link our shared core library with each role application, we used the solution offered by Monorepo⁷ which allowed a single repository to contain multiple projects with well-defined relationships.

6.2.3 Feature Breakdown

This subchapter aims to review, analyze, and explain each feature of the system in a short and concise manner. For a more detailed description on how to use each feature, please refer to the product manual.

Shared Features

Navigation Drawer

The main navigation menu used amongst all the applications is the navigation drawer. It can be accessed from any page inside the application by pressing on the burger menu icon located on the top left corner of the screen. For each application, it will offer a navigation button to each available screen.

Bottom Navigation Bar

The bottom navigation bar is a standardized widget used amongst all the applications. It is displayed on the bottom part of the screen as a row containing three buttons. Starting from right to left, the user can access the map, QR scanning tool or the settings page.

Profile

The profile screen is a standardized page used amongst all the applications. It displays a card containing the user's name, profile picture, and an upload button that allows the user to change his profile picture.

Settings Page

The settings page is a standardized page used amongst all the applications. It displays the user profile of a user and a list of settings menus. The settings menus are as follows: Language, Privacy and

⁷ Monorepo - <https://monorepo.tools/>

Accessibility. They are not implemented since the requirement for these settings were of a low priority, therefore they are left for the client to develop in the future. On the bottom of the screen, there is an option which allows the user to delete his account. This functionality is implemented, and it permanently deletes a user's account along with his stored details.

Map

Each application has access to a map which displays the user location and a pin for each bin accessible by the user role. When a pin is pressed, the screen will animate and zoom the camera onto that object. Moreover, an info window will appear at the bottom of the screen which displays the fill levels for each bin compartment, a direction button that redirects the user to the exact location on Google Maps, the bin identifier and location.

Bin Status

The bin status screen is used to show the status of each bin available to the user role. Only supervisors and cleaners can access this screen. On the screen, a list of items is provided. Each item shows four icons representing one of each stream (Organic, PMD, Paper and Residual), the online status of the bin and a dropdown which, on expansion, shows four vertical lines representing the fill levels for each compartment of the bin. The supervisor role has an additional feature, the total waste chart, that can be accessed by tapping on any list item.

Disposer Features

History

The history page displays a list of disposed items. Its purpose is to offer the user a way to query and visualize all his disposed items. Each item is presented in the form of a card containing the type of waste, time of disposal, location, bin identifier and the number of points earned. The page is presented in the form of an endlessly scrolling list, which paginates the results in batches of 10 items. The data can be queried using four time ranges: past day, week, month, or all time.

Statistics

The statistics page is used to display the total number of points earned at each organization, and the amount of CO2 emission savings in kilograms. The points sub-menu shows a list of organizations and the points earned at each of them, ordered by the highest number. The CO2 emissions sub-menu shows a chart that highlights the CO2 emissions in kilograms that a user emitted using the smart bins, a theoretical estimate of CO2 emissions without the bin, and the CO2 savings which is calculated by taking the difference between the two. The data shown in the chart is queried from all the bins that a disposer used, regardless of the organization. Moreover, the data can be queried using four time ranges: past day, week, month, 3 months or 1 year.

QR Scanning

The QR scanner on the disposer app acts as a bridge between the waste thrown in the bin, and the disposer history. When a user scans a QR code on the side of a bin, a 60 second timer starts, and the application informs the user to throw his waste into the bin. After 60 seconds, or if the user confirms his action before the timer ends, a celebration screen will appear. Due to the polling rate of the bins with the main server, it might take between 0 and 10 minutes to link the disposal history to the user's account. A user might also throw waste in a bin without scanning a QR code, but in that case, he will not be awarded points.

Cleaner Features

History

The history page displays a list of emptied bins. Its purpose is to offer cleaners a way to query and visualize all their emptied bins in a specific time range. Each item is presented in the form of a card containing the type of emptied stream (PMD, Paper, Residual or Organic), the time at which the action was performed, and the identifier of the bin. Moreover, on the corner of each card, a cleaner can press on a map icon that redirects the user directly to the bin location on the map screen. The page is presented in the form of an endlessly scrolling list, which paginates the results in batches of 10 items. The data can be queried using four time ranges: past day, week, month, or all time.

QR Scanning

The QR scanner on the cleaner app enables cleaners to register their task of emptying a bin. When a cleaner scans a QR code located on the side of the bin, a modal is displayed with a checklist containing the waste type streams in a bin. The user can check from one to four streams and then submit his choice. This system allows cleaners to track their progress, and supervisors to oversee them.

Supervisor Features

Total Waste Chart

As discussed in the bin status screen overview, when a supervisor touches a bin list card, a screen is displayed which contains a total waste overview. This overview is composed of the current fill levels of each stream of a bin, the total amount of waste thrown in a selected time range, the percentage of increase in the amount of waste thrown, and a total waste chart which shows the number of waste items, by category, thrown into the bin. The data can be queried for the past day, week, month, or year.

Cleaners

A supervisor oversees the activities of the cleaning staff responsible for cleaning the bins of their organization. In order to do so, the application offers supervisors a screen to query and audit the cleaning history of each cleaner linked to their company. The screen contains a calendar that allows the selection of a single day in a time range spanning one year in the past. For each day, supervisors see a list of all the cleaning tasks, where each list item shows the name of the cleaner, the bin identifier of

the cleaned bin, the timestamp of the action and the streams that have been cleaned. Moreover, supervisors can filter by name through a dropdown located under the calendar widget.

QR Scanning

The QR scanner on the supervisor app enables supervisors to register a bin for the company they supervise. Whenever a bin is scanned, the system checks if it is unregistered. If so, then the bin will automatically be assigned to the supervisor's company. If it is registered under his company, then the user will be redirected to the bin status screen.

6.3 Server-side architecture

6.3.1 System architecture

Considering, we built mobile application logic within the same code environment for the client's preexisting web Dashboard logic, in order to separate concerns, we have differentiated between priorly created URL routes (endpoints) and newly created ones by us for the mobile application logic in `app_urls.py`. Moreover, we have created separate files for views (`app_views.py`) and serializers (`app_serializers.py`), calling corresponding methods and querying database for specific mobile app requests, allowing for easier navigation and maintenance abstracting from the prior logic as much as possible.

Main client's request was to introduce the main entity in the whole mobile app hierarchy architecture – Company. Administrators can create new company with its corresponding unique name and unique identifier code, its own point system (for each type of waste thrown away using this company's bin, a specific number of points can be chosen to be assigned for a disposer). Each company can have a lot of bins, a lot of cleaners and supervisors. The company is the main relational entity between the above-mentioned ones.

Another of the main requests of our client was to create different entities for each specific user. Since we decided on building on top of the existing client's server architecture, we abstracted as much as possible to be able not to compromise their existing endpoints and server logic which is being used for their existing priorly created Dashboard web application. The server side uses Django framework, that is why we exploited it further on. Later in sections, the solutions and their reasoning are going to be presented for specific client requests.

Until now there existed only one type of user called "Custom User" which stored the main information about the user account. Because of the certain specific of how Django allows for only one certain user model to be authorized, since it handles the session, cookie token creation on its own, as well as password hashing and salting, we were forced to extend the existing "Custom User" model to allow for other user models to inherit all of the trivial user management functionality, such as logging in, logging out, password resetting, forgetting.

Moreover, it was not feasible to delete “Custom User” model since it would have compromised their whole existing web application. Therefore, we created the Disposer, Cleaner and Supervisor entities that extend from the Custom User model. That is why any of the new user models directly referencing Custom User model, and on creation, deletion both are manipulated. This not only allows for 3 more independent entities in the system, but also allows for client’s old application to see all the statistical data as it has been envisioned prior by them (all disposers’ throwing trash away actions are reflected on their dashboard application).

6.3.2 Disposer

As per requirements, the disposer can track and register their recycled waste by scanning the QR code. Moreover, they receive points based on how much waste they have recycled if claimed by scanning the QR code on a bin. A disposer can also see detailed statistics about their CO2 emissions, collected total amount of points and points per company (to which a bin belongs to) and has access to a map showing locations of nearby bins.

Initially the disposers are the users who are supposed to discover the application via the advertisement on a bin and scanning the corresponding QR code, redirecting them to a corresponding PLAEX website page yet recognized as anonymous users. The web page has corresponding routes to download the app either in Google Store or App Store. Subsequently installing the app and registering their account, they get access to certain disposer’s role functionality.

Waste assignment architecture

Per client’s request, to bind a thrown away waste with a disposer, we have come up with the following solution: a disposer needs to scan a QR code of the bin, and all the litter thrown away in this bin within one minute after the scan is registered to this disposer.

The complexity of this is that all the bins are not synchronized with the database simultaneously the moment the waste is being recognized within a bin, instead it is saved in the local memory of a bin. A bin is polling the database with waste records every 10 minutes. Moreover, since we have no control over the software system architecture in a bin itself, it poses another problem of having a disposer not being associated with waste thrown away explicitly.

That is why we have come up with the solution of the automatic scheduler running in the background of the server on a separate thread. Scheduler's straightforward purpose is to find the connection between the bins and waste and according to timestamps when a certain scan has been registered and a certain waste has been thrown away, respectively. Every disposer’s scan is being saved in the database with a timestamp when the scan has happened, a bin ID the waste was thrown into, and a reference to a disposer. Every waste

that has been thrown away is received by database within 0-10 mins (because of the polling system set up within a bin). Thanks to that, the scheduler queries both scans of disposers and wastes records in the database, comparing timestamps and reference to the same bin in both scans and waste records. Every waste registered within 1 minute, starting from the scan's timestamp, is assigned to a disposer (the reference to a disposer is taken from registered scan record).

The scheduler runs every 2 minutes (time can be adjusted), which queries the records of scans and wastes registered maximum of 2 hours ago (time can be adjusted). By doing this, we account for the cases when the bins have been offline and have not been able to poll locally stored records of thrown away wastes. In order not to query the same records again and again which fall within two-hour period from the current time moment the scheduler runs, we set a flag 'points_assigned' to true in both scan records and waste records, efficiently reducing the querying time. Hence, the next time the scheduler queries only the records having 'points_assigned' flag equaling to false.

Points assignment to a Disposer

Disposers are assigned points for every company they have used a bin of, according to a specific point system of that company. Each time the records of waste and scan match according to the timestamp and bin ID, the points are assigned to a disposer for a corresponding company. The more and more disposers use specific company's bins, the more points they accumulate. For storing all the points for different companies for a single disposer, we store in the 'companies_points' column as list of inner Json objects, avoiding a lot of relational tables and reducing querying time. However, we must serialize the contents of this field every time, when there is update in the amounts of points for a certain company or a company is deleted at all. Anyway, it makes the database structure less complex.

CO2 Emissions values

Another requirement from our client was to create a chart showing the co2 savings score. The disposers will be able to see how much CO2 they have saved based on their thrown waste. To get the needed information all of the "scan tasks" that are linked to a disposer are being retrieved, after which the corresponding waste for every scan is being saved in a list (similarly to how the waste history is being retrieved). To calculate the score itself we multiply the waste's weight by a constant value that is different for each of the four waste types. The disposer can also filter the information he sees based on a preset period of time (day, 1 month, 3 months, 6 months and a year), and also waste type. Moreover, the chart will show the difference between the co2 emissions made by using a Garby bin and without it. This way the users will be able to see how big of a difference they are actually making by using our client's product.

6.3.3 Cleaner

The cleaner's entity is responsible for managing the bins. Every cleaner is assigned to a company they will get access to bins at. Multiple companies can be assigned to a single cleaner, since per client request, there are cleaner agencies, providing their workers, so we had to accommodate that in the system. <link to some screenshots with the screens>

Once a cleaner is assigned to a company which has bins assigned to it, cleaners are authorized to see all the relevant information such as fill levels of each subtype of waste in each bin, bin's location. Regarding actions, they empty a bin's certain compartments once they are full (PMD, paper, residual, organic) by scanning a bin's QR code and manually registering the emptied streams. They are also authorized to see all their emptying-bins history.

Since the fill levels are also polled with a 10-minute interval, in order to simultaneously register the emptied streams in the cleaner's history and accommodate for the cases when the bins might be not connected to the network, we chose to stick to the manual input from the cleaner about the emptied compartments. This allows not only for cleaners themselves to clearly see their history right away, but also to make it bulletproof for their supervisors that certain bins have been cleaned. In this case, however, we rely on truthful input from cleaners.

The alternative could have been to create another scheduler, checking saved cleaner's scan and the most recent fill level updates which are going to appear in the database after the polling, but there is no guarantee in this case that certain bins will ever report change of fill level status being offline. This could compromise the cleaner's work, even if they have emptied it, but a bin is not capable of polling due to the lack of a network connection.

6.3.4 Supervisor

The supervisor's entity is created to monitor the overview of their company's assigned cleaners and bins. Moreover, they can register new devices (bins) to the company they belong to by scanning the QR code on a new bin (refer to the next section). Each supervisor can be assigned to a single company and observe all company's bins current fill levels of various compartments and all emptying actions of cleaners to the bins belonging to the same company.

Considering that the current app modifications are supposed to be merged to their current application, to make the smooth merge as possible, we account for their existing "Custom Users", which are accounts for using the overview dashboard, meaning that they have more access to the data (to be more exact all the data irrelevant to any company). Hence, all the existing "Custom users" records in client's database are assigned supervisor role (even though role can be changed).

However, the supervisor entity has to be created on their own via the script as an example or manually via admin dashboard (refer to the next sections).

6.3.5 Scanning Feature

All entities have access to the scanning functionality. For the disposer it saves a scan record assigning a timestamp that marks the waste that belongs to them. The cleaner can register a scan featuring the time when a specific bin is being emptied by scanning a bin's QR code and invoking modal on the client side in the mobile application for a cleaner to indicate what streams are emptied. By scanning a QR code, the supervisor can register a bin to their company if it is not registered yet or view its status in case the bin is already registered within their company.

We also account for the scans being done from outside of either of 3 applications for disposer, cleaner, supervisor. We class them as anonymous scans. The use case is when an interested person wants to scan a bin's QR code, they will be redirected to an advertisement web page on the PLAEX website referencing links to Google and App stores. (Since it was out of our scope working on the web application itself, we left to the future development by the client; currently anonymous users are redirected to www.plaex.net)

QR generation

In order to integrate scanning into bins, the corresponding QR codes must be generated and stuck on them. The template of the QR code: '<current domain name>/api/scan/<bin id>'. For example, 'plaex.net/api/scan/5', where 5 is the existing bin id. Assuming, this very bin has been registered via admin dashboard with the id 5.

6.3.6 Account registration process

By the design, the disposer is the only user that can be registered straight from the app. The cleaner and supervisor accounts are created by the system administrator from the admin dashboard and assigned to the corresponding company.

6.3.7 Account deletion process:

If any users want to delete their account, they can do so from the settings page. On deletion of all of 3 roles, user reference to any related entity is being set to 'Null', allowing to leave scan records, or bin registrations to companies unaffected for statistical data being uncompromised. However, personal details including images are removed from the database and server memory.

6.3.8 Image storing

Since old implementation of the system had no infrastructure of dynamically storing uploaded images to a user profile, we decided to facilitate storing of all uploaded profile images and company images to the media/uploads folder of the project repository under the folder of the name of the entity they belong to and the folder of the name of the record itself, in which the image is stored. (media/uploads/<entity name>/<record name>/image.png). Different types of images are supported.

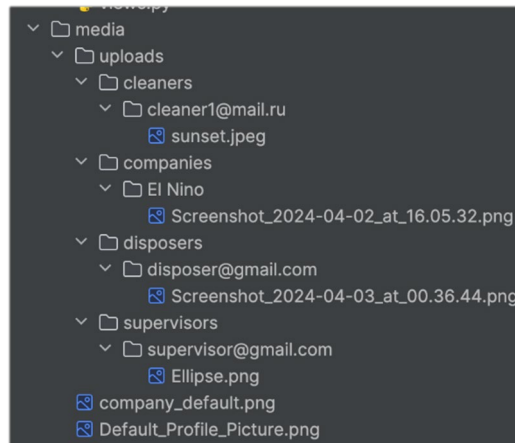


Figure 6.3.8 Server File System

Each entity has only one image stored in the corresponding folder. In case there is previously stored image, it is being removed (when a profile or company image is updated, as an example), hence minimizing the occupied memory storage on the server side.

6.3.9 Endpoints Permissions

Every app endpoint has been thought of specific permission access. We utilize custom permission infrastructure such as 'IsSupervisorOnly', 'IsCleaneronly', 'IsCleanerOrSupervisor', 'IsBinAssigned' along with default ones, provided by Django 'IsAuthenticated' or 'AllowAny'.

6.3.10 Admin Dashboard

Django framework features by having a lot of high-level templates for management applications. That is why we have made use of one of such as admin dashboard, since our client requested some kind of administrative environment where they can manage created entities rather than changing and modifying data in the database directly.

Hence we have built our mobile app application with the intention of utilizing admin dashboard, accessible via '/admin' slug following the domain name (depending on where the server side is hosted). The admin dashboard is accessible only by user accounts having 'is_staff' and 'is_superuser' flags set to True. To give access to any of the created user entities, their referenced record of a "Custom User" must have above mentioned flags set accordingly.

Creating, deleting, updating, uploading images, setting point system of points within a company are done via the admin dashboard, specifically under 'Company' entity. 'Cleaners' are assigned to a company inside the company management interface.

Creating, deleting, updating, uploading images for different kinds of users is also done the same way, more specifically, for 'Cleaners' and 'Supervisors'. The 'Disposer' also can be managed via dashboard, however the main method of doing that is intended to be done by the user itself via the corresponding mobile application.

Company assignment for 'Supervisors' and 'Bins' is done inside management interface of those entities, opposite to 'Cleaners'. The reason for that is a difference between company type interconnection and its lower hierarchy entities.

Managing 'Bins' is also intended to be done via admin dashboard, specifically: creating new instances, updating, deleting, setting specific location of a bin, unique identifier, assigning a company.

In addition to that we added custom fields, showcasing the database records interconnection to better understand it or check for assigning mistakes. For example, all the bins assigned to company X are shown for cleaner Y, because cleaner Y is assigned to company X and vice versa.

6.3.11 System Changes

During the development we have made some changes to the database structure with the purpose of improving the scalability and sanitation of the data. Previously our original database had the "waste_type" column from the "Waste" table as a String field. This would lead to a lot of errors in the database since there were cases in which the created waste would not have the correct name (for example "organic" instead of "Organic"). For this reason, we created a separate table called "Waste_Type" where the correct names of different waste types are created. Thus, the "waste_type" column is now a foreign key referencing the "Waste_Type" table. This change will ensure that every waste from the database has a standardized name to avoid any inconsistencies.

6.3.12 Bins

We have used the old legacy logic from the dashboard endpoints on getting the online/offline statuses of the bins. It is adjusted to every user role accordingly (supervisor, cleaner, disposer), as different bins depending on a company or not at all apply. To determine if a bin is online, it checks if the last waste is registered within the last 3 minutes. We strongly recommend extending this period to more than 10 minutes at least, since it is the maximum time interval for polling. ('/api/get_bins_on_off_status' endpoint)

Another essential issue resolved is querying the current fill level of the bins. In the old legacy implementation, the most recent fill level is taken during that current day at the time of querying, which

imposes the following issue: in case during that same day, no one has thrown any trash in that bin, the fill level will be empty (which might be not true). Hence, we created a mobile app separate querying endpoint which takes the most recent fill level ever registered. (`/api/getlastestfilllevel_app`) This way, the levels will be shown empty, only if the most recent record of a fill level is empty. We strongly recommend reviewing dashboard fill levels retrieving implementation as it might impose confusion.

Chapter 7

System Testing

In this chapter, the focus is placed on the testing methodology used for validating the requirements of the system. The first section will cover a test plan which describes the types of tests used as acceptance criteria for the proposed system, whereas the second section will cover the test plan's results.

7.1 Test Plan

7.1.1 Test Plan Approach

The testing methodology chosen for this project will focus on providing sufficient tests to achieve maximum coverage of the system-related code, while also defining clear acceptance criteria for each system requirement. Unit tests will be used for automated testing of existing code, while also providing means of automated review of newly added functionalities in the form of CI/CD pipelines. Integration testing will be used to determine the overall system interoperability and usability testing will serve the purpose of defining clear and consistent UI and UX design patterns.

As per the IEEE-829 standard for software and system test documentation (IEEE, 2008), the test plan will include: a software risk assessment section, which will describe the potential risks that the system will encounter, and the solution proposed to address them, a feature-overview which describes the risks that could be encountered from a user perspective, test pass/fail criteria, and a schedule that will be based on realistic and also validated estimates as the project development continues. Moreover, a test results section will be added which will present how our testing procedure underwent and offer insights into additional risks that were discovered throughout this process.

Unit Testing

Unit testing will be conducted from the earliest stages of development to allow automated regression testing. The unit tests will be created on the frontend and backend frameworks with the tools made available by each framework. The purpose of these tests is to integrate them into the development pipeline to allow CI/CD operations, thus ensuring a consistent and bug-free coding process.

Integration Testing

Integration testing will represent most of our testing methodology due to the proposed system. The focus is to cover all the endpoints used by the application and their interaction with the UI placeholder created in the front-end framework. Due to the agile-based task management of our development phase, multiple errors could appear due to inconsistencies in code, data, or exception handling, therefore the scope of the integration tests are to protect the development process from these pitfalls.

Usability Testing

For usability testing, we based our methodology on the heuristic evaluation method proposed by Nielsen and Molich (Nielsen J., 1990) which offers a set of simple, yet effective rules used to identify problems associated with the user experience and overall design of the system. The choice of methodology was dictated by the efficiency of the framework in the limited time offered by the development process. Moreover, to address the disadvantages of this approach such as evaluation bias, we decided to perform these tests iteratively along with our client, thus allowing us to form an unbiased evaluation of the overall usability of the system.

Among the heuristics proposed by Nielsen, the following subset of rules represent the acceptance criteria set for the system:

- **Visibility of system status**

The system should always keep users informed about what is going on when an action is performed. Since the mobile application will interact with separate systems such as the smart waste bins service, if a timeout occurs in the transmission of data, the user must receive appropriate feedback within reasonable time.

- **Error prevention**

The user interface should have a careful design which prevents a problem from occurring in the first place rather than displaying error messages, wherever applicable. All irreversible actions such as deletion of data must present users with a confirmation option before committing.

- **Consistency and standards**

Users should find the design familiar after the first use of the application, without having to resort to the documentation for different situations that result in the same process. This implies that the interface must follow pre-defined system-wide conventions. This is especially true in our system architecture where three applications are developed rather than one.

- **Recognition rather than recall**

User's memory load should be minimized by reducing the number of actions that are hidden under nested menus or gestures. The user should not have to remember information from one interaction to another, whenever appropriate.

- **Aesthetic and minimalist design**

Along with a consistent design, the application must also have a minimalist design. This implies that dialogues, menus, and pages should not contain information which is irrelevant or rarely needed.

- **Help and documentation**

Even though the system's purpose is to be as consistent and minimalist as possible, documentation should still be available to all application users. Any action provided by the system must be documented concisely in a separate report or FAQ section.

The user interface will be considered adequate only if it manages to follow the list of heuristics. The acceptance criteria will be reviewed and given by the stakeholders of the system; therefore, the application can only be accepted when the interface and usability satisfy the demands of the users.

7.1.2 Software Risk Assessment

In this section, a risk assessment will be undertaken by categorizing potential risks into several types such as: technical, security, scalability, performance, and schedule risks. By documenting all the risks at an early stage of the project design, our team can take proactive preventative actions during the software development process.

- **Technical risks**

- Unforeseen technical complexity of the backend currently in use by the client.

A measure taken as a consequence of the requirements laid down by the client is to re-use the existing backend deployment of our client's dashboard to allow rapid development of the proposed system. This approach could arise unforeseen issues such as increased complexity of code or incorrect API behavior.

- Integration issues with the REST API server currently in use by the client.

The proposed system will offer additional functionalities that are not currently supported by the backend REST API server that the system will use. By extending or creating new functionalities, integration issues might arise which would require careful testing and mitigation.

- Incomplete or inaccurate requirements

Issues inside the requirements elicitation process might lead to incomplete or even inaccurate requirements that could be left unnoticed until the client reviews a minimum viable product. This could lead to additional requirements that must be implemented, hence increasing development time.

- **Security risks**

- Insecure API communication

Security risks might arise due to the inclusion of sensitive data inside URL's, request bodies or even headers that might get exposed during API calls.

- Insecure coding practices

Even though Python and Dart are much safer than older languages such as C in terms of memory leak related attacks, there are still some ways of exposing sensitive data by not following standard security practices in these languages. One example might be keeping a database connection, hostname, and password in a visible-text format String inside the source code.

- Scalability risks
 - Poorly designed architecture and design patterns

Abstraction, modularity, and maintainability are the three key elements that set the base of the proposed system architecture. By decoupling the backend REST API completely from the mobile application, we can interact with it by only using interfaces that describe the communication objects used by the API. Moreover, the mobile application will use a scalable architecture that separates concerns between models, views, and controllers.
- Performance risks
 - Network latency issues

API endpoints can be subject to network latency issues. Since the mobile application will be connected to the internet using WI-FI or cellular, there is a possibility that API requests could fail. The system must address these issues to reduce as much as possible the impact on the end-user.
 - Excessive use of system resources

A mobile environment dictates that the application must run on a variety of hardware combinations. Poor coding practices such as database connections that are left unclosed could lead to memory leaks that severely impact the user experience, or even crashes of the application.
 - Poor database performance

By choosing to use the same backend as the currently in use dashboard of our client, database performance could be affected due to the increase in the number of requests. Poor data management could lead to data loss, database locking or slow response times.
- Schedule risks
 - Improper requirements prioritization

Requirements prioritization is not a trivial task. Wrong prioritization could cause scheduling issues such as increased development time or even requiring changes to what requirements can be implemented in the given amount of time.
 - Poor task-management skills

Not all team members have the same experience in terms of design, development, or report-writing skills. Failing to acknowledge this fact by not discussing each team member's strengths and weaknesses could lead to poor task delegation and therefore delays in the proposed schedule.

7.1.3 Features to Be Tested

This section describes the functionality of the system from a user’s perspective and provides a prioritized overview of features based on a simple rating scale: *High (H)*, *Medium (M)*, *Low(L)*, which represents the level of risk for the respective function. A higher risk level represents a higher priority for testing that function. The feature list is extracted from the current requirements of the system; therefore, it may change during the development process. Refer to [Appendix C](#) for a list of all system requirements.

Feature to Be Tested	Related System Requirements	Risk Level
Login with shared credentials between the application and dashboard	3.1, 3.2	H
Logout	3.1	H
Dispose trash using the app	9.1, 9.2	H
Check disposal history	6.1	H
Check bin fill levels	10.1, 16.1, 20.1	H
Register a cleaned bin to the bin cleaning history	13.1, 14.1, 15.1, 17.1	H
Check cleaning history	13.1, 17.1, 18.1, 18.2	H
Assign cleaners to multiple organizations	12.1	H
Link supervisors and cleaners only through a company entity	24.1	H
Add devices to organization	23.1, 26.1	M
Receive points when disposing trash using the application	5.1, 9.1, 9.2, 22.1	M
Check total of earned points	8.1, 22.1	M
Search a bin on the map	4.1, 11.1, 25.1	M
Query user-related data such as CO2 footprint and point rankings	7.1, 8.1, 19.1, 21.1, 22.1	M
Determine precisely what trash is assigned to a disposer	9.2	L
Query the frequency of bins filling up	19.1	L

Table 7.1.3 Features to Be Tested

7.1.4 Features Not to Be Tested

This section is aimed at features that are important for the functioning of the final system, but which are not required to be thoroughly tested because either previous testing was performed or is linked to a system outside the scope of the mobile application.

One purpose of the application is to display relevant data regarding the system of deployed sensors and devices that send waste-related data to the central server. Since the way in which the sensors and devices send data to the server is outside the boundaries of the mobile application, our team will not test this feature.

Another feature that will not be tested is the accuracy of the machine learning model implemented by the “Garby” devices aimed at identifying and classifying waste.

7.1.5 Test Pass/Fail Criteria

To assess the effectiveness of our testing, a pass/fail criterion must be established which describes in detail the methodology of analyzing the success of each test case. Since this project makes use of more than one type of testing, the pass or fail criteria for a test item is dependent on the type of testing being performed on that item.

Unit Testing

The pass/fail criteria for unit tests are straightforward as they will be determined by the testing frameworks of our choosing: Flutter and Django's main testing frameworks. Based on our chosen criteria for assert statements, the frameworks will indicate for each unit tests whether it passed or failed.

The unit testing section will be completed only if all test cases pass, and if they cover the entirety of the main functionalities of the application and API. A notable exception from the industry standard is that we will not aim for high code coverage outside our own created API endpoints or classes since all the code offered by the client will be tested using Postman, thus treating the backend as a 'blackbox'.

Integration Testing

Integration testing, due to its broad scope, does not offer a clear evaluation framework for each and every project. Since every system has different specifications, communication protocols and technology stacks, the assessment criteria must be chosen according to the proposed system's own architecture. In the case of this project, we deemed sufficient that first, all the existing endpoints present on the backend server offered by the client are to be tested and documented only if they are used to request and send data from and to the mobile application. Moreover, integration testing would also imply that we must test to the extent that is possible, the backwards compatibility of newly added functionality on the backend side with the deployed web application that PLAEX is using. The integration testing section will be considered complete only if the newly added functionalities are compatible with both the web application, and the mobile application.

Usability Testing

Usability testing, as integration testing, represents a wide category of tests and criteria to oversee. The subjectivity that is involved in this category of testing renders the Boolean outcome used in the previous two testing types obsolete. Therefore, our approach is to use what is called a System Usability Scale (SUS) scoring system to quantify the usability of our application. The first phase of testing will consist of three to four meetings with our client to fine-tune the look and feel of the mobile application design using a wireframing tool. The second phase will represent the final stage of testing where the minimum viable product and subsequently, the finished product will be presented to the client. This test will be complete only if the results of our testing will return a satisfactory SUS score.

Acceptance Testing

For this testing type, each of our features listed in Table 7.1 will be tested and reviewed. These features represent the aggregation of all system requirements. The criteria for pass or failure will be described in a table under the section “Test Results” for each listed feature. This section of testing will be completed only if all the features have been tested according to the listed criteria. The minimum requirement of acceptance is that all features with a high priority must be fully functional.

All test cases of high priority in Table 7.1 must meet the criteria without exceptions. Medium priority cases are important features that must be implemented, but do not affect the core functionality of the app, and such scarce exceptions are accepted. Low priority features will be implemented last and should work reliably in most scenarios.

1. Login with shared credentials between the application and dashboard
 - Pass Criteria: Successful login with shared credentials results in access to both the application and dashboard.
 - Fail Criteria: Failure to login with shared credentials or inability to access either the application or dashboard.
2. Logout
 - Pass Criteria: Successful logout redirects the user to a logout confirmation page or the login screen.
 - Fail Criteria: Any issues encountered during the logout process or failure to redirect to the expected page after logout.
3. Dispose trash using the app
 - Pass Criteria: Trash disposal process completes successfully, and the user receives a confirmation message or notification, together with the number of points they have gained for the specific organization.
 - Fail Criteria: Failure to dispose of trash or encountering errors during the disposal.
4. Check disposal history
 - Pass Criteria: Accessing the disposal history displays a list of previously disposed items with accurate timestamps and details.
 - Fail Criteria: Inability to view disposal history or discrepancies in the displayed information.
5. Check bin fill levels
 - Pass Criteria: Accurate representation of bin fill levels is displayed in real-time.
 - Fail Criteria: Inaccurate or inconsistent representation of bin fill levels.
6. Check work schedule
 - Pass Criteria: Accessing the work schedule displays upcoming shifts and details about them accurately.
 - Fail Criteria: Inability to view the work schedule or discrepancies in the displayed information.
7. Register a cleaned bin to the bin cleaning history
 - Pass Criteria: Successful registration of a cleaned bin updates the cleaning history with relevant details.
 - Fail Criteria: Failure to register a cleaned bin or inaccuracies in the recorded cleaning history.

8. Check cleaning history
 - Pass Criteria: Accessing the cleaning history displays a list of previously cleaned bins with accurate timestamps and details.
 - Fail Criteria: Inability to view cleaning history or discrepancies in the displayed information.
9. Add devices to organization
 - Pass Criteria: Successful addition of devices to the organization results in their inclusion in the organization's device list.
 - Fail Criteria: Failure to add devices or errors encountered during the addition process.
10. Receive points when disposing trash using the application
 - Pass Criteria: Users receive points upon successful trash disposal, and the points are accurately matched to the type of trash disposed and the amount of points the organization offers for that specific type of trash
 - Fail Criteria: Failure to receive points or discrepancies in the awarded points.
11. Check total of earned points
 - Pass Criteria: Accessing the earned points displays the total points earned by the user accurately for each individual organization.
 - Fail Criteria: Inability to view earned points or discrepancies in the displayed total for any of the organization.
12. Search a bin on the map
 - Pass Criteria: Successful search displays the location of nearby bins accurately on the map.
 - Fail Criteria: Inability to locate the bins or errors in the displayed map information.
13. Query user-related data such as CO2 footprint and point rankings
 - Pass Criteria: Retrieval of user-related data, such as CO2 footprint and point rankings, provides accurate and up-to-date information.
 - Fail Criteria: Inability to retrieve user-related data or discrepancies in the displayed information.
14. Determine precisely what trash is assigned to a disposer
 - Pass Criteria: Successful determination provides a clear list of trash items assigned to the disposer.
 - Fail Criteria: Inability to determine assigned trash items or inaccuracies in the displayed information.
15. Query the frequency of bins filling up
 - Pass Criteria: Accurate retrieval of data regarding the frequency of bins filling up.
 - Fail Criteria: Inability to retrieve frequency data or discrepancies in the displayed information.
16. Check cost savings and CO2 footprint
 - Pass Criteria: Accessing cost savings and CO2 footprint information displays accurate calculations and metrics.
 - Fail Criteria: Inability to view cost savings and CO2 footprint data or discrepancies in the displayed information.

This layout provides clear pass/fail criteria for each feature, ensuring that testing is conducted thoroughly, and results are evaluated objectively according to the defined criteria.

7.1.6 Test Schedule

To make sure that we fit in the given time frame, we scheduled to do all the tests during our 4th sprint. The following table contains the weekly schedule of the 2-week testing sprint

Feature to Be Tested	Period of time
Login with shared credentials between the application and dashboard	01/04 - 05/04
Logout	01/04 - 05/04
Dispose trash using the app	01/04 - 05/04
Check disposal history	01/04 - 05/04
Check bin fill levels	01/04 - 05/04
Register a cleaned bin to the bin cleaning history	01/04 - 05/04
Check cleaning history	01/04 - 05/04
Assign cleaners to multiple organizations	01/04 - 05/04
Link supervisors and cleaners only through a company entity	07/04 - 12/04
Add devices to organization	07/04 - 12/04
Receive points when disposing trash using the application	07/04 - 12/04
Check total of earned points	07/04 - 12/04
Search a bin on the map	07/04 - 12/04
Query user-related data such as CO2 footprint and point rankings	07/04 - 12/04
Determine precisely what trash is assigned to a disposer	07/04 - 12/04
Query the frequency of bins filling up	07/04 - 12/04

Table 7.1.6 Test Schedule Overview

7.2 Test Results

7.2.1 Unit Testing

The Flutter framework offers a suite of packages and tools to perform unit testing for BLOC components which handle the business logic of the application, and for widgets that are the screens displayed to the user. Our team implemented a basic suite of unit tests that artificially imitate the requests and responses from our various API endpoints and then check whether the state of the screens changed accordingly. However, the requirements of the project underwent constant changes during each weekly client meeting, therefore a test-driven approach was not appropriate for our development environment. The repository offers examples of how to implement such unit tests for future development of the application. The results are displayed in the form of Boolean pass or fail. Additionally, tests can also be included in the development pipeline to avoid introducing bugs inside the repository.

The bulk of unit testing is represented in the form of API endpoint testing using Postman⁸ where each newly developed endpoint is automatically tested using Postman's JavaScript testing framework. Collections of API endpoints are therefore automatically tested after every backend modification by checking authorization levels, and if the response adheres to a pre-defined JSON schema. Analogous to Flutter's unit testing, the results are Boolean pass or fail.

7.2.2 Integration Testing

Integration testing represented a continuous process throughout the whole duration of the project. To accelerate the development integration of both frontend and backend, we deployed a docker container to host a server accessible for everyone. Every backend change was released, tested, and used from the shared server to assure a consistent development environment.

The first layer of testing includes GitHub pipelines which test every pull request before merging the oncoming changes onto the development branch for linting issue. This approach allowed us to avoid introducing any hard to track bugs into our code.

The second layer is represented by manual testing during development. A test-driven development method would have been preferred over manual testing, but the lack of development time and the flexibility of requirements did not allow us to use this methodology. Our manual testing methodology included testing features, as mentioned in section 7.1.3, where each linked requirement was checked according to the table in [Appendix C](#).

⁸ Postman API Endpoint testing tool - <https://www.postman.com/>

7.2.3 Usability Testing

Usability testing was performed weekly during our meetings with the client. During each session, all three applications were thoroughly reviewed, analyzed, and tested to ensure that they are first, what the client needs, and secondly, usable. After each session, a list of feedback was composed, and it was transformed into additional features or fixes. More details can be found in the Appendix section where short drafts of each meeting are presented.

7.2.4 Post-hoc Risk Analysis

Section 7.1.2 lists a brief overview of generalized risks that were thought of during the design phase of the project. As development started and eventually stopped, multiple risks were discovered. This section aims to list all the discovered risks and how they were either solved, or what solutions were discussed for future development to address.

Current risks

- **Cleaner misuse of the scanning feature**

Scanning a bin only registers the timestamp at which the cleaner supposedly emptied a bin, therefore a situation in which an employee might lie to the system and not empty the bin might occur.

One way to fix this issue would be by creating a new feature to our already running scheduler that would respond to every scan request from the cleaner and verify the fill level of a bin, in the case in which a bin's fill level hasn't changed within 5 minutes of the scan request then it could be marked as a "false scan".

- **Disposer misuse of the throwing waste assignment feature**

Since the scheduler assigning waste to a disposer is working based on a timestamp, a malicious user might try to continuously scan the QR-code of a bin confusing the scheduler and getting assigned other people's waste.

To mitigate this, a new feature could be added to the scheduler, so it checks that every scan request was sent after a certain amount of time from the first one, so that the disposer was not able to do multiple scans.

- **Backend overload threat**

Currently our system does not implement any rate limiting services, therefore our server and database are susceptible to being overflowed with requests from malicious users.

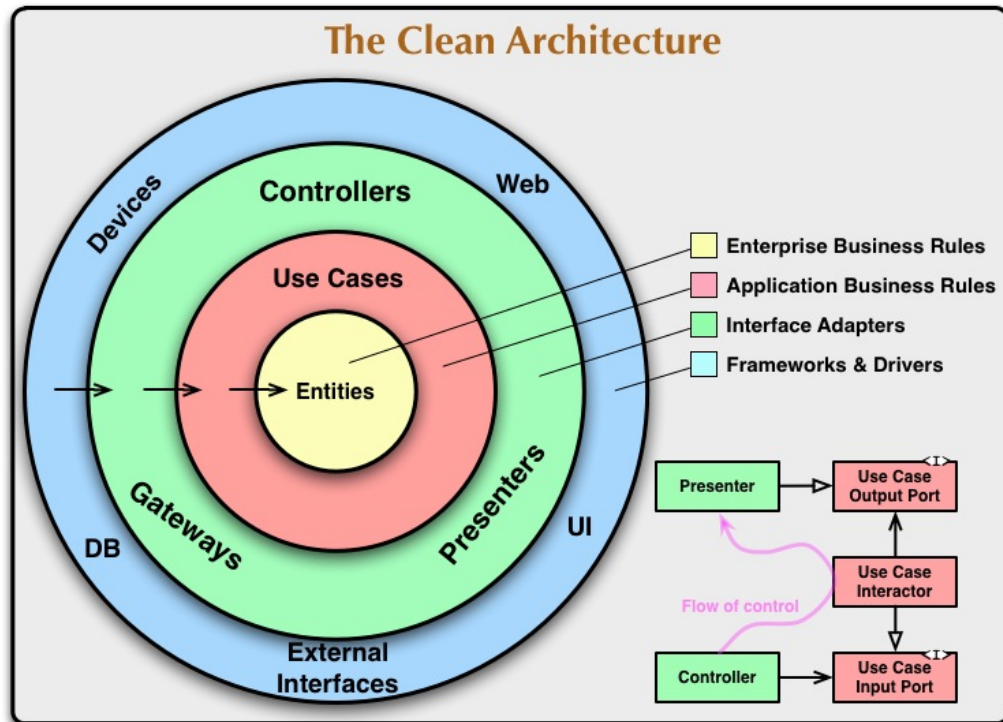
Similarly to the previous risk, the easiest way to alleviate this threat would be to set a maximum number of requests that can be made by a user in a given time period.

References

- Clegg, D., & Barker, R. (1994). *Case Method Fast-Track: A RAD Approach*. Addison-Wesley.
- Doran, G. T. (1981). *There's a S.M.A.R.T way to write management's goals and objectives*. Retrieved from community.mis.temple.edu:
<https://community.mis.temple.edu/mis0855002fall2015/files/2015/10/S.M.A.R.T-Way-Management-Review.pdf>
- IEEE. (2008). IEEE Standard for Software and System Test Documentation. *IEEE Std 829*, 1-150.
- Lucassen, G. D. (2016). The Use and Effectiveness of User Stories in Practice. *Requirements Engineering: Foundation for Software Quality, Lecture Notes in Computer Science, Springer International Publishing, vol. 9619*, 205-222.
- Nielsen J., M. R. (1990). Heuristic evaluation of user interfaces. *ACM CHI'90*, (pp. 249-256). Seattle.
- UN. (n.d.). *SDG Goals*. Retrieved from Sustainable Development Goals: <https://sdgs.un.org/>

Appendix A

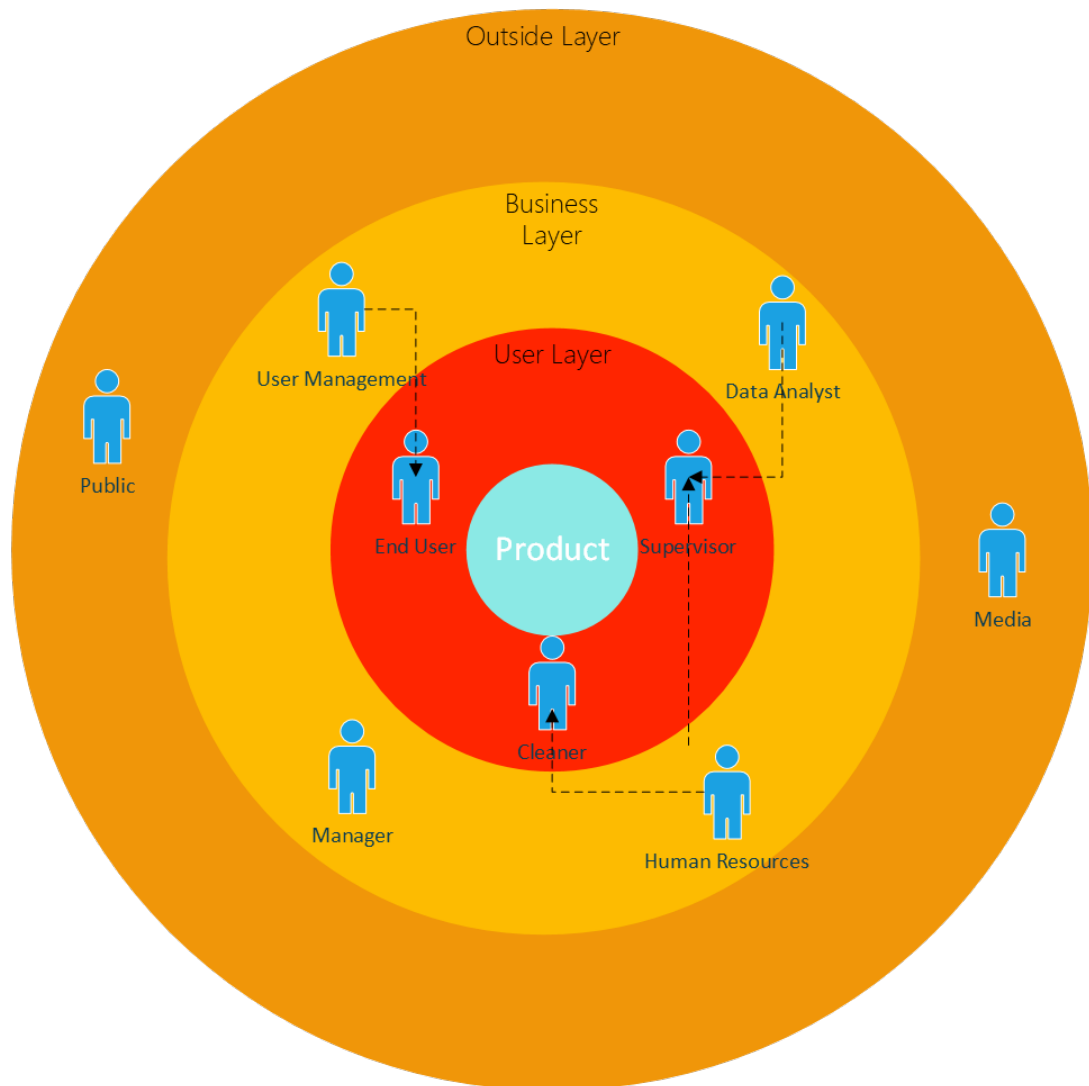
Flutter Clean Architecture Model



A.1 Clean Architecture Model⁹

⁹ The Clean Architecture <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>

Stakeholder Onion Model



A.2: Stakeholder Onion Model

Appendix B

Class Diagram

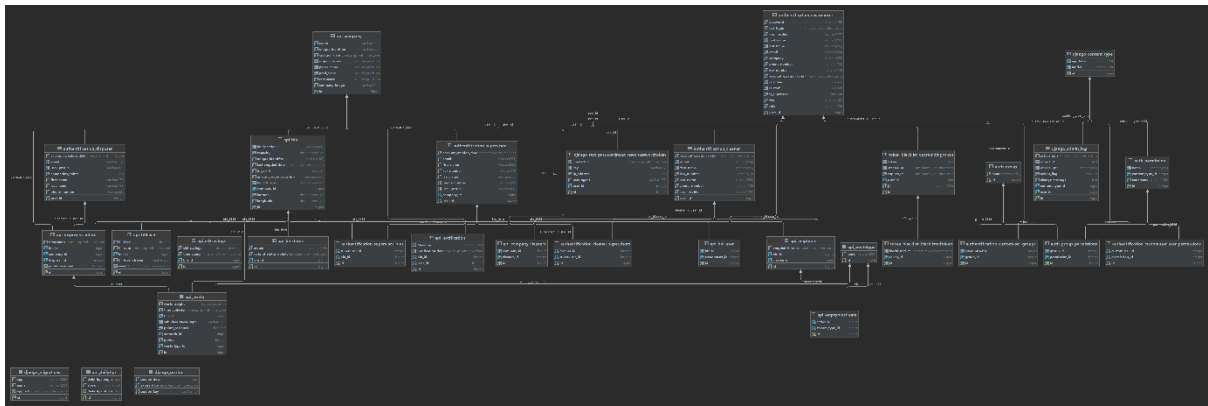
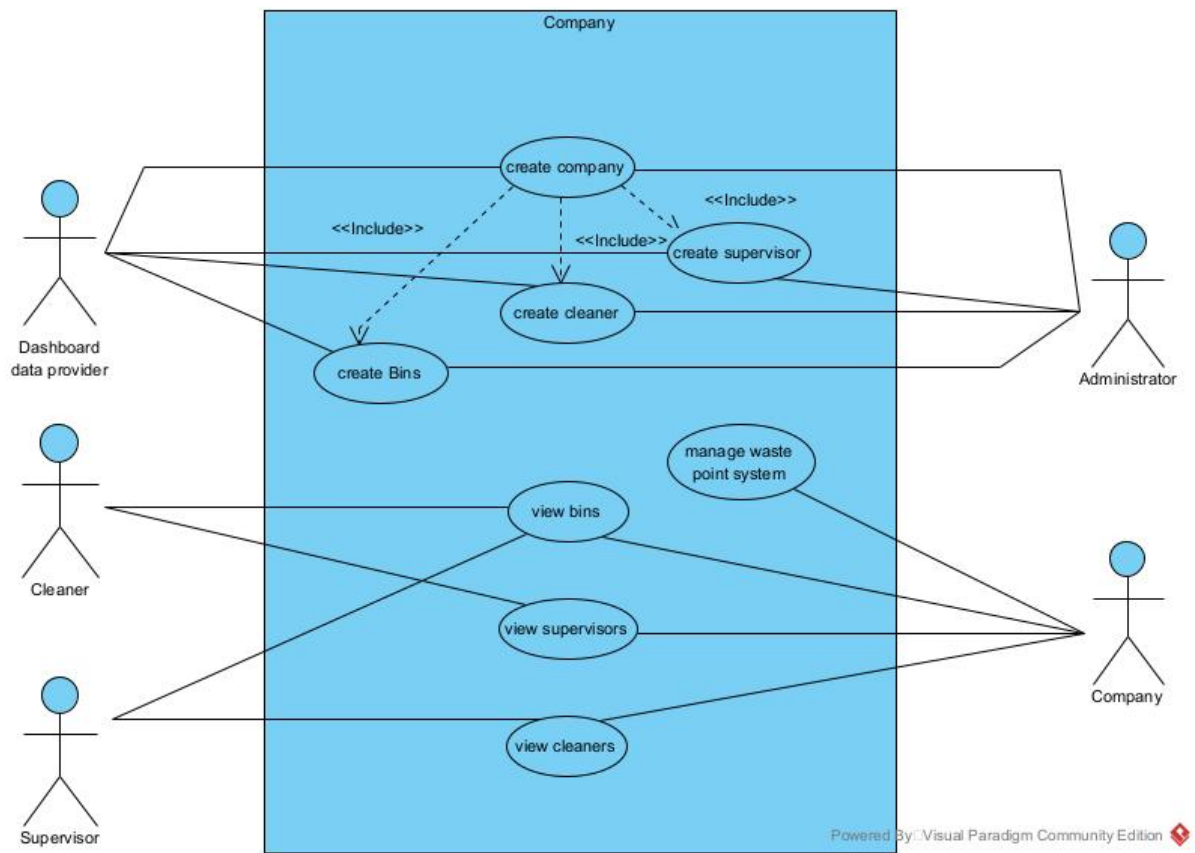


Figure 2 Class diagram for the entire system

Use Case Diagram



B 1: Use Case Diagram for the Dashboard

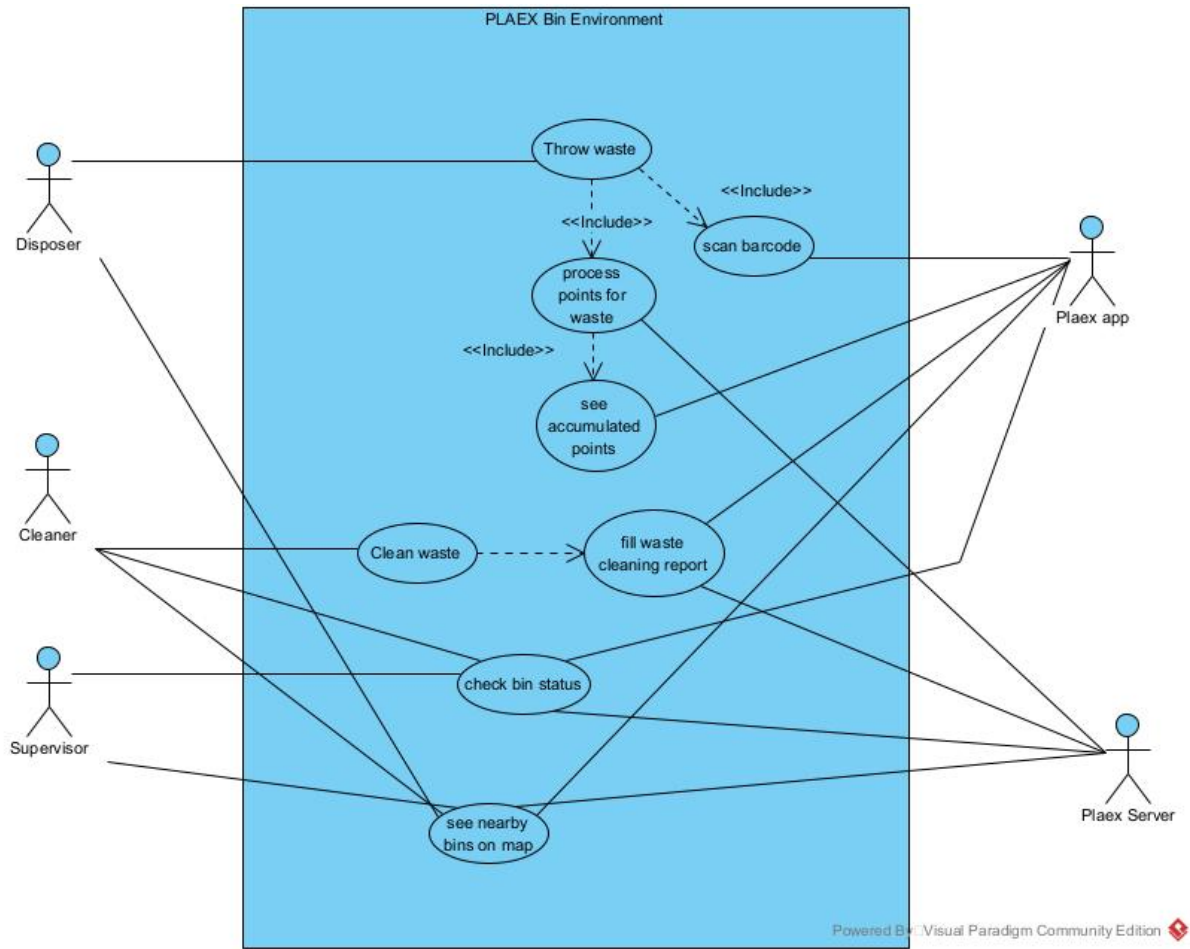


Figure 3 Use Case Diagram for Bin functionality

Sequence Diagram Disposer

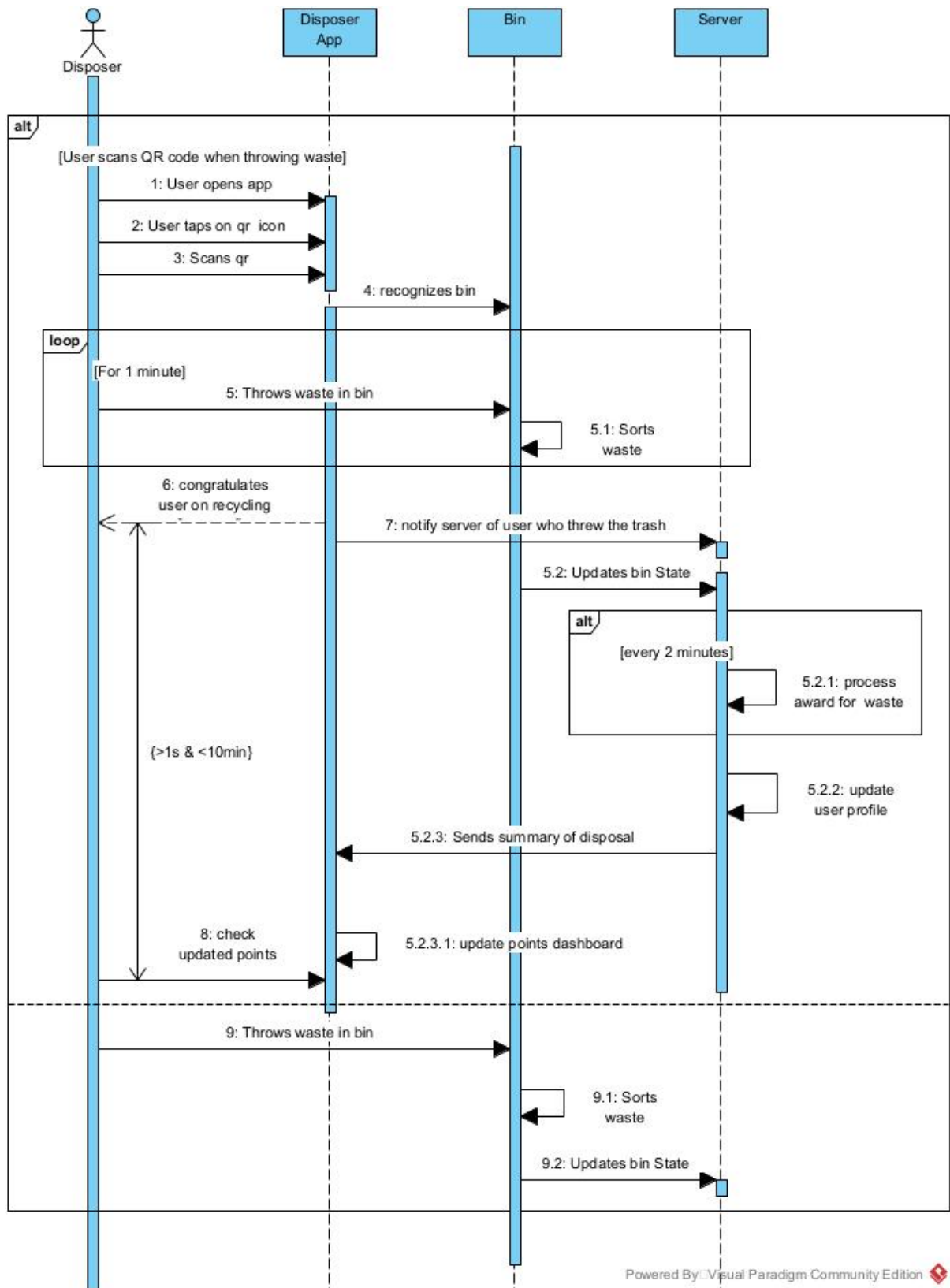
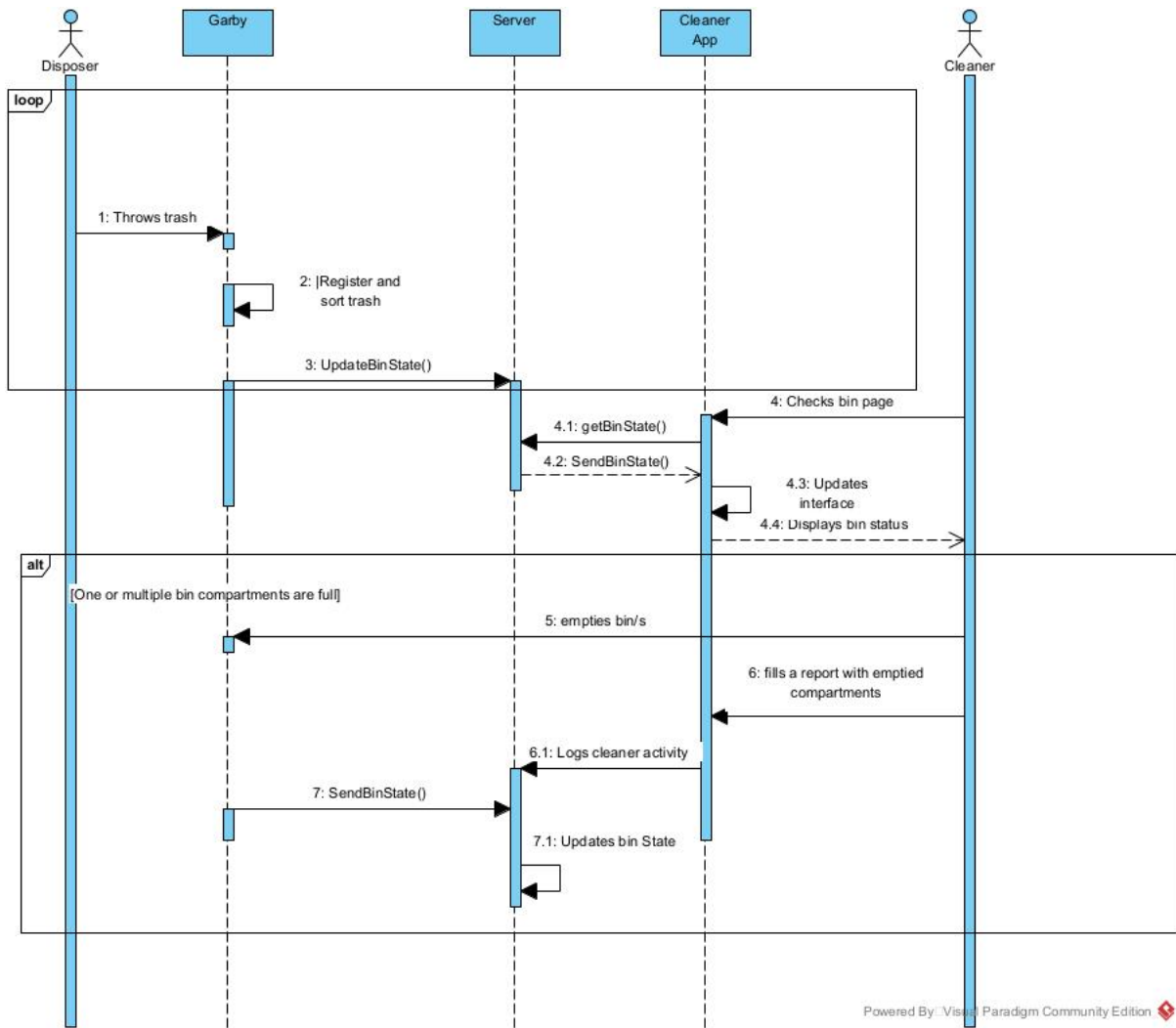


Figure 4 Sequence Diagram for Disposer functionality

Sequence Diagram Cleaner



Powered By: Visual Paradigm Community Edition

Figure 5 Sequence Diagram for Cleaner functionality

Sequence Diagram Supervisor

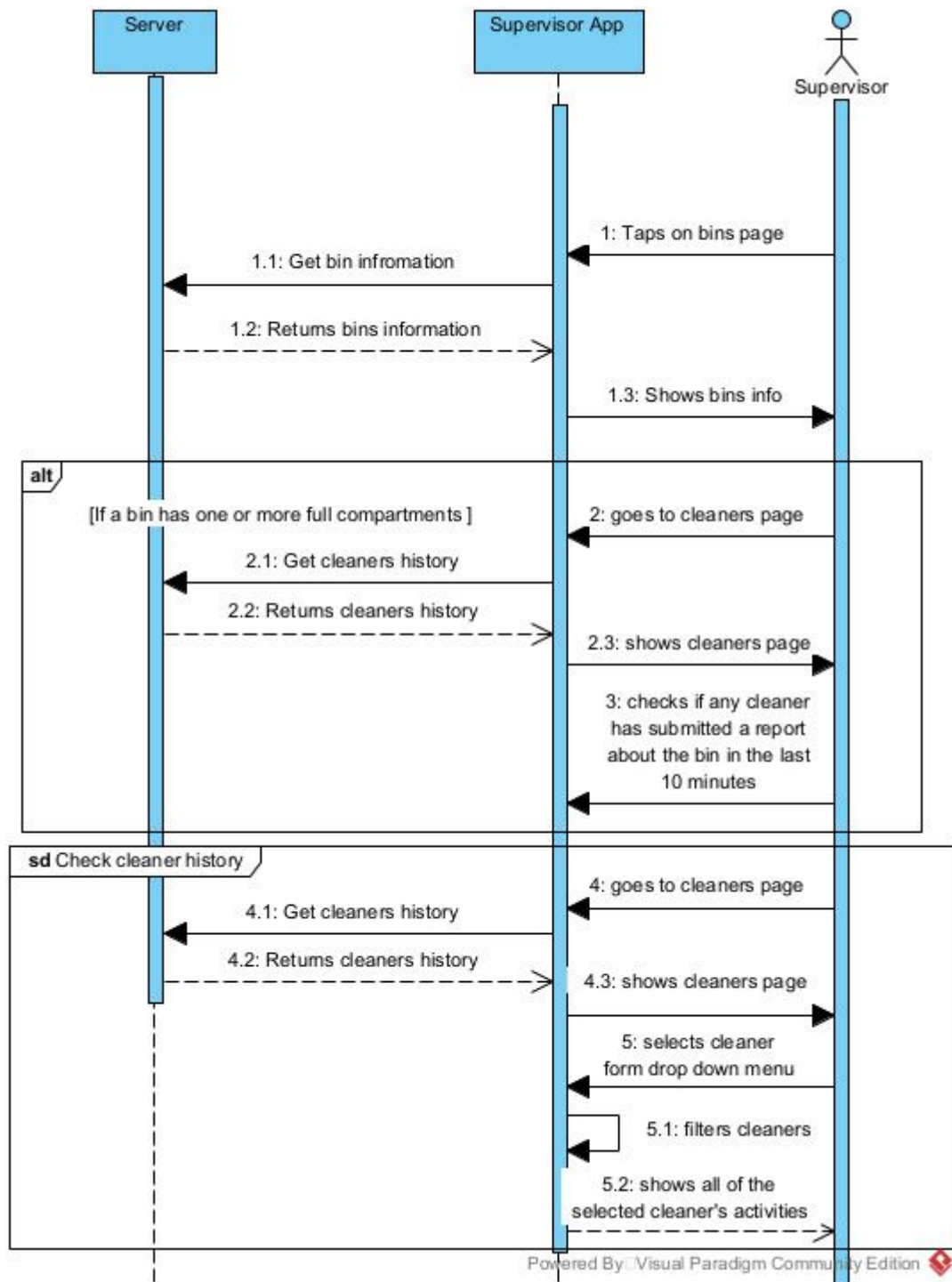


Figure 6 Sequence Diagram for Supervisor functionality

Activity Diagram Waste Disposing

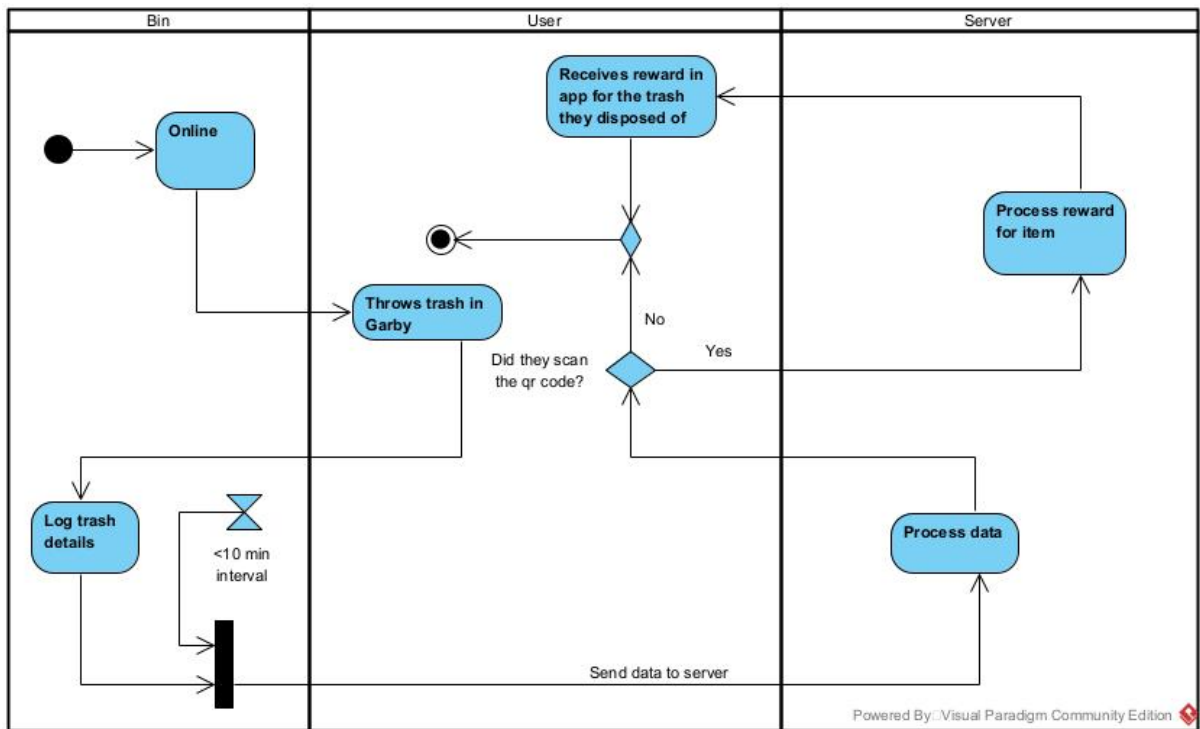


Figure 7 Activity Diagram for showing the waste disposing process

Appendix C

System Requirements

ID	Description	Criteria	Performance	Bandwidth	Priority	Source
1	As a user I want to access PLAEX's webshop from the mobile application					
1.1	The system won't fully integrate PLAEX's webshop implementation	They do not have a functional webshop implementation yet.	-	-	Won't / No Priority	PLAEX / Developer
2	As a user I want to navigate inside buildings to a bin where GPS signal is not available					
2.1	The system won't use ultra-wide band to locate bins	Lack of time and support to implement hardware side	-	-	Won't / No Priority	PLAEX / Developer
2.2	The system won't do path-tracking inside all buildings towards bins that need to be emptied	-	-	-	Won't / No Priority	PLAEX / Developer
3	As a user I want to authenticate as one of the three roles (supervisor/disposer/cleaner)					
3.1	The system must allow users to sign-up and login with their account	Time to successfully log in and out	Maximum of 10 seconds	+/- 5 seconds	Must / High	Developer
		Successful sign-up and login process	Pass or fail	-		
3.2	The system must authorize users based on their credentials	Time to authorize a user based on his credentials	Maximum of 10 seconds	+/- 5 seconds	Must / High	Developer
3.3	The system could be divided into 3 apps for each type of user	Successful division of applications and library implementation for common parts	Pass or fail	-	Could / Low	Domain Expert
4	As a user (supervisor/disposer/cleaner) I want to see the locations of each bin on a map					
4.1	The system could show a pin for each bin on a map	Usable UI & time to load pins on the map	Maximum of 15 seconds	+/- 10 seconds	Could / Low	Developer

5	As a disposer I want to receive points when I dispose of trash					
5.1	The system should assign points to disposers for each disposed of waste	Time to assign points after dispose action	Maximum of 10 minutes	+/- 20%	Should / Medium	Developer
6	As a disposer I want to see my disposal history					
6.1	The system must show the disposal history for each disposer	Successful implementation	Pass or fail	+/- 5 seconds	Must / High	User
		Time to retrieve history data	Maximum of 10 seconds			
7	As a disposer I want to see my CO2 footprint and savings from using the system					
7.1	The system should show the CO2 footprint for each disposer and organization in the form of a chart	Time to retrieve the data and display it on the chart	Maximum of 15 seconds	+/- 5 seconds	Should / Medium	PLAEX / Developer
8	As a disposer I want to see the total amount of points, I received from disposing trash					
8.1	The system could show the total number of points for each disposer	Time to retrieve and display the total points	Maximum of 10 seconds	+/- 5 seconds	Could / Low	PLAEX
9	As a disposer I want to scan a QR code to identify in which location I dispose of trash					
9.1	The system must allow disposers to scan a QR code prior to throwing waste in a bin with the purpose of linking the waste items to them	Time to register action after scan and to initiate the process of waste disposing	Maximum of 5 seconds	+/- 2 seconds	Must / High	User
9.2	The system must link each thrown waste object to a user if a QR code was scanned at the time of disposing the item	Successful implementation	Maximum of 10 minutes	+/-20%	Must / High	Developer
10	As a cleaner, I want to see what bins need to be emptied					
10.1	The system must inform cleaners of which bins need to be emptied	Successful implementation of filtering fill levels to show most full bins first in the bin list	Pass or fail	-	Must / High	Developer

11	As a cleaner, I want to see the quickest path towards the bins that need to be emptied					
11.1	The system could show a path towards the bins that need to be emptied	Successful implementation of routing to a map location	Pass or fail	-	Could / Low	Developer
		Time to redirect to the application which handles the routing	Maximum of 10 seconds	+/- 5 seconds		
12	As a cleaner, I want to be assigned to different companies					
12.1	The system must allow the assignment of multiple companies to a single cleaner	Successful implementation of linking multiple companies to one cleaner entity	Pass or fail	-	Must / High	PLAEX
13	As a cleaner, I want to see the location and timestamp of the bins that I emptied					
13.1	The system must show the bin cleaning history of each cleaner	Successful implementation of cleaning history bookkeeping	Pass or fail	-	Must / High	PLAEX
		Time to retrieve cleaning history	Maximum of 10 seconds	+/- 5 seconds		
14	As a cleaner, I want to scan a QR code to manually confirm the emptying of a bin					
14.1	The system must allow cleaners to register the action of emptying a bin by scanning a QR code	Time to register action after scan and to initiate the process of bin cleaning	Maximum of 5 seconds	+/- 2 seconds	Must / High	Developer
15	As a cleaner, I want to indicate what streams have been emptied manually					
15.1	The system must allow cleaners to select the bin compartments that they cleaned	Successful implementation	Pass or fail	-	Must / High	User
16	As a cleaner, I want to see an overview of fill levels for each bin					
16.1	The system must show the fill levels of each bin	Time to retrieve fill levels per bin	Maximum of 3 seconds	+/- 1 second	Must / High	User

17	As a cleaner, I want to see the streams I have emptied at a certain bin					
17.1	The system must show each compartment per bin that was emptied, on the cleaning history of a cleaner	Successful implementation	Pass or fail	-	Must / High	Developer
18	As a supervisor, I want to monitor the activity of all cleaners of my organization					
18.1	The system must show the cleaning history of each cleaner in the company in which the user is a supervisor	Time to retrieve cleaning history for all cleaners in a day	Maximum of 5 seconds	+/- 2.5 seconds	Must / High	PLAEX
18.2	The system must allow filtering based on date and cleaner names for the history preview	Successful filtering based on names / cleaning timestamps	Pass or fail	-	Must / High	Developer
19	As a supervisor, I want to see the amount and type of waste collected in each bin					
19.1	The system must implement a function to show the bin waste history and increase percentage for each bin inside the organization supervised by the user.	Time to retrieve waste related data per bin and display on the chart	Maximum of 10 seconds	+/- 5 seconds	Must / High	PLAEX
20	As a supervisor, I want to see an overview of the fill levels for each bin					
20.1	The system must show the fill levels of each bin	Time to retrieve the fill level per bin	Maximum of 5 seconds	+/- 2 seconds	Must / High	User
21	As a supervisor, I want to see the CO2 footprint and cost savings of all bins for the organization					
21.1	The system must implement a function that allows the retrieval of CO2 footprint and cost savings for each bin inside the organization supervised by the user	Time to retrieve and parse the data to display on a chart on the supervisor app	Maximum of 10 seconds	+/- 5 seconds	Must / High	PLAEX / Developer
22	As a supervisor, I want to see the ranking of points distributed to each user					
22.1	The system must implement a ranking system that keeps track of all points allocated to a disposer under the organization supervised by the user	Successful implementation of point tracking system	Pass or fail	-	Must / High	PLAEX / Developer

23	As a supervisor, I want to add devices to my organization by scanning a QR code on the respective devices					
23.1	The system should allow supervisors to register new bins by scanning the QR code of an unregistered bin	Time to perform action after scanning	Maximum of 10 seconds	+/- 5 seconds	Should / Medium	PLAEX
24	As a supervisor, I want to add cleaners to the organization that I supervise					
24.1	The system must link supervisors and cleaners through a company	Successful implementation of company superclass that links these types of roles	Pass or fail	-	Must / High	PLAEX / Developer
25	As a supervisor, I want to assign a location for each bin of my organization					
25.1	The system should let supervisors assign a latitude and longitude location for each bin	Successful implementation of geographical data storage on the bin entity	Pass or fail	-	Should / Medium	Developer
26	As a supervisor, I want to manage my smart bin system from an administrative page					
26.1	The system must have an administrative page managing all the entities and relations	Successful implementation of the administrative dashboard through the Django framework	Pass or fail	-	Must / High	Developer

Appendix D

Test Schedule Table

Feature to Be Tested	Time Period
Login with shared credentials between the application and dashboard	01/04 - 05/04
Logout	01/04 - 05/04
Dispose trash using the app	01/04 - 05/04
Check disposal history	01/04 - 05/04
Check bin fill levels	01/04 - 05/04
Register a cleaned bin to the bin cleaning history	01/04 - 05/04
Check cleaning history	01/04 - 05/04
Assign cleaners to multiple organizations	01/04 - 05/04
Link supervisors and cleaners only through a company entity	07/04 - 12/04
Add devices to organization	07/04 - 12/04
Receive points when disposing trash using the application	07/04 - 12/04
Check total of earned points	07/04 - 12/04
Search a bin on the map	07/04 - 12/04
Query user-related data such as CO2 footprint and point rankings	07/04 - 12/04
Determine precisely what trash is assigned to a disposer	07/04 - 12/04
Query the frequency of bins filling up	07/04 - 12/04

Meetings

First meeting – February 9th, 2024

For our first meeting, we began by introducing ourselves to Victor Okoro (the co-founder of PLAEX). Victor started presenting to us the current state of the PLAEX web application and mentioned several mandatory requirements that need to be present in our application. We also discussed some possible “nice to have” features and afterwards, we established a channel communication during this project, specifically how often and where we can contact one another. Victor added us on their team’s slack channel for app development, where we were able to get in touch with their full-stack developer and database administrator. We were also provided with some Figma mock-ups of their web application, which would prove useful when creating our Figma prototype for the app. We must mention that we were surprised to see our client eating during the meeting, but it did not affect the flow of the conversation.

Second meeting – February 19th, 2024

In the second meeting we focused on showing the progress we had made on the design of the app, using Figma. We got some pointers into what direction to develop, such as to not stick by the pre-existing design and to try to think outside the box. He liked the idea of showing the bins of the map, so we will stick to it. For the disposer role the starting page should show the total points for each organization, and in the history tab, each type of disposed trash should be mentioned, together with the location if possible. The cleaner should be able to see if a bin is online or offline and should not be able to take part in the points system. For the supervisor, one should be able to see separate bin fill streams for each waste, and a history page with cleaning history would be more useful than a future schedule.

Third meeting – February 26th, 2024

This meeting we presented the final design in Figma before we would move on to the next phase of development. We received more feedback about how big the bar graphs should be, the color scheme of the app, and other small details, and we achieved a consensus from both parties (us and them) on how the app will look. We encountered some difficulties in communicating with the client, as we were focused on contouring the functionalities of the app through the design elements, and Victor was more focused on minute details of the appearance such as colors and font sizes. This, added to the fact that our client liked to engage in conversation in a colloquial tone, which was a one-sided channel of communication that was not agreed by both parties involved, made the collaboration difficult at times. Once the main ideas for the design were established, we turned our attention towards the system requirements, discussing what functionalities would be needed for each user.

Fourth meeting – March 4th, 2024

During the 4th meeting we glossed over the requirements again with Victor to finalize them, and then we informed him that building 3 separate apps would be the best course of action. A concern was raised regarding access to the database, and after some further discussion Victor gave us the green light to copy their database to use it for developing and to build upon it where necessary. We expressed our interest to schedule a meeting with someone from the technical department, which we did the next day.

Fifth meeting – March 5th 2024, with Minase

During the meeting with Minase, he handed us the credentials for the database, and he explained details about the backend, how it is made in python, and the hardware in the bin. Unfortunately, due to network connection errors, we could not discuss in detail all our points of interest, but we have received enough

information to put us on the correct path. He gave us access to the GitHub repo and let us know that there is documentation from the previous team members. Minase agreed to update the documentation for the dashboard website on how to set up the environment.

Sixth meeting – March 11th, 2024

Over the course of the weekend we sent the APK to Victor, so he could get a feel for the app. He was not happy about the fact that the app deviated from the original Figma design, and he was worried about how the UI would scale up on different device sizes. We have also received the feedback on the supervisor page, the cleaners page must be changed to a more intuitive approach.

Seventh meeting – March 18th, 2024

During this meeting we presented Victor with a first implementation of the supervisor bin screen, which was an adaptation of the web UI into a mobile interface. Victor was dissatisfied with the result, and gave several points of feedback, such as modernizing the graph, making 3d elements that look like buttons seem more flat and modern, and reduce the size of the time selection buttons. He also mentioned that we need to add a company entity on top of the supervisor, and each device should be linked to a company and supervisor, but cleaners should not be linked to supervisors. Subsequently, all supervisors should be able to see the work of any cleaner within the company. For the next meeting, he requested another APK.

We did not have a meeting the following week, as we focused on the back-end and such we had no update for Victor.

Eight meeting – April 8th, 2024

Victor was once again displeased about the chart, he wanted an approach that would seem to overflow from the sides of the screen, and to create another method for time selection that would save space, by implementing a drop-down menu. He also let us know that we need more details for the sign-up page for a new user, such as name, email, phone number. Additionally, he wanted a settings page with language, accessibility options and privacy policy buttons, and we discussed the way that cleaners would be added to the system. It was unfortunate that he revealed these requirements so late in our project, thus it was agreed that we will try to implement his wishes to our best ability in the short time span left.

Ninth meeting – April 16th, 2024

In our last meeting with the client, we presented the 99% finished end product, and he seemed mostly satisfied. Victor liked the overall feel of the app and how intuitive it felt. However, when we showed him the final chart design he was still dissatisfied, as it was not looking exactly how he envisioned it, and he was concerned with the fact that the data points did have a continuous shape. We informed him that the representation was due to not enough data points present in the graph, but he kept insisting that we handle the issue, so his attitude on the matter left us vexed. For extra security, Victor asked us if we could make a user verify their password when they create a new, account, but due to the time constraint we informed him that we most likely won't be able to implement that feature. Minase also joined in towards the end of the presentation so we could discuss handing in your codebase to Plaex when we will finish the project. In the end we said our goodbyes and thanked everyone for their involvement in the project.

Appendix E

PLAEX Mobile Application Manual

SETTING UP THE SYSTEM	63
SETTING UP ADMINISTRATIVE ACCOUNT	63
CREATING NEW ENTITIES	65
COMPANY	67
SUPERVISOR	68
CLEANER.....	70
BIN.....	72
QR CODE CREATION	73
USING THE APPS	74
DISPOSER.....	74
<i>Login</i>	74
<i>Disposer Navigation Drawer</i>	76
<i>Profile and Settings</i>	77
<i>History</i>	78
<i>Statistics</i>	79
<i>QR Scanning</i>	81
CLEANER.....	82
<i>Bin Status and Map</i>	82
<i>Navigation Drawer and History Screen</i>	83
<i>QR Scanning</i>	84
SUPERVISOR	85
<i>Bin Overview Screen</i>	85
<i>Cleaners</i>	86
<i>QR Scanning:</i>	87
<i>Navigation Drawer</i>	88
SETTING UP THE FLUTTER ENVIRONMENT FOR THE APPS	89

Setting up the system

None of the old legacy commands have changed.

To set up the system refer to 'docs/backend/setup_backend.md' file.

P.s. Requirements.txt file has been updated with new libraries.

Recap:

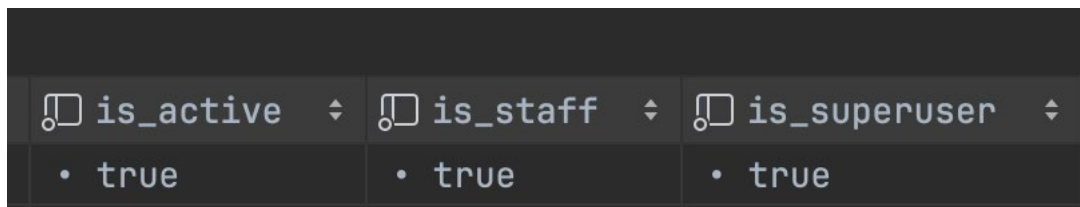
- Create virtual environment being in root directory ('Dashboard' directory in the terminal); activate the venv
- When venv is activated, navigate to 'plaex_backend' directory in the terminal
- Migrate the database ('python manage.py makemigrations' & 'python manage.py migrate')
- Run the server ('python manage.py runserver')

Setting up administrative account

When the server is running, you must either use the existing account with administrator permissions (1), give them in the database manually (2), or create a new Custom User account and grant it administrator permissions (3).

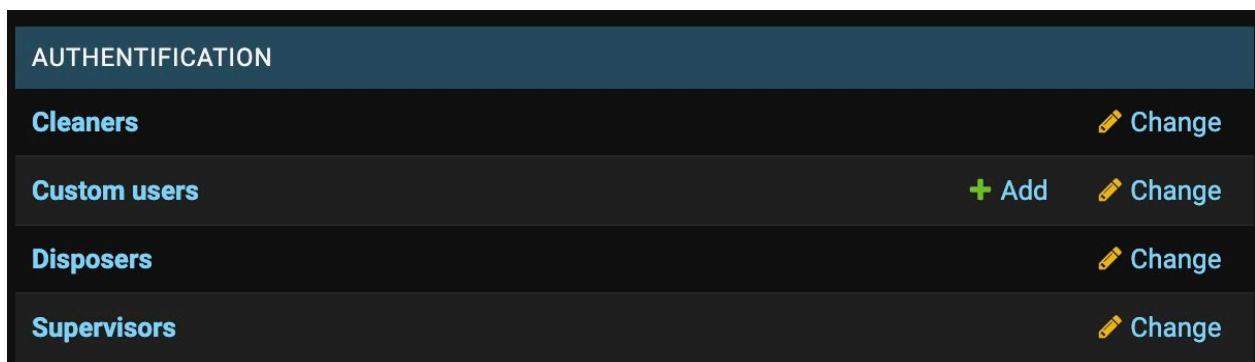
Administrator account is the Custom User account having 'is_staff' and 'is_superuser' fields set to true.

You can open modify record in 'authentication_customuser' table (2)



If you want to create the new Custom User, navigate to <domain name>/admin url.

Keep in mind, the following url is accessible only to administrative accounts.



In the opened interface, press “Add”.

Add custom user

First, enter a username and password. Then, you'll be able to edit more user options.

Email Address:

First name:

User profile: No file chosen

Last name:

Company:

Phone number:

Kik number:

Tier:

Role:

Password:
Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

Password confirmation:
Enter the same password as before, for verification.

Is active

Is staff

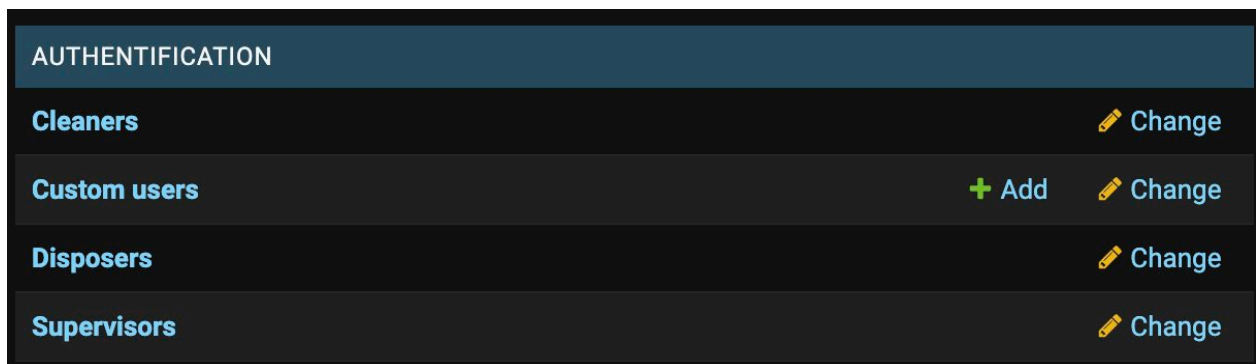
Fill in required fields, check 'Is active', 'Is staff' and click save. In the DB, the field 'is_superuser' has to be set to true manually anyway.

Creating new entities

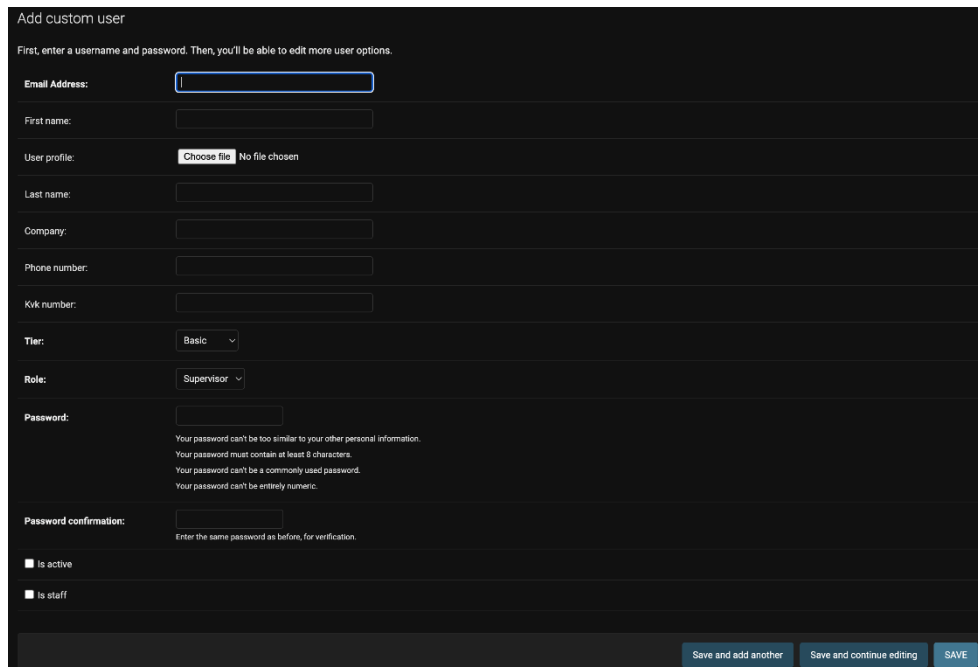
Admin Dashboard is a crucial interface upon which the system relies on. You are able to create new bins, companies, supervisors, cleaners, disposers and interrelate them between each other.

Disposer

For testing purposes disposers can be created manually via admin dashboard, even though, they are supposed to be created via disposer application.



In order to create a disposer, click "Add" for Custom User.



The "Add custom user" form includes the following fields and options:

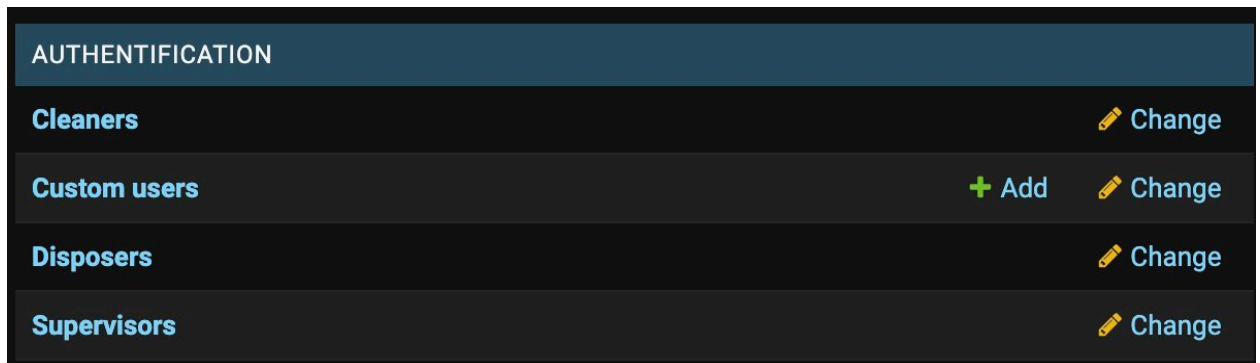
- Email Address:** Text input field.
- First name:** Text input field.
- User profile:** File upload button labeled "Choose file" and "No file chosen".
- Last name:** Text input field.
- Company:** Text input field.
- Phone number:** Text input field.
- Kvk number:** Text input field.
- Tier:** Dropdown menu with "Basic" selected.
- Role:** Dropdown menu with "Supervisor" selected.
- Password:** Password input field with validation rules: "Your password can't be too similar to your other personal information.", "Your password must contain at least 8 characters.", "Your password can't be a commonly used password.", "Your password can't be entirely numeric."
- Password confirmation:** Password input field with the instruction "Enter the same password as before, for verification."
- Active/Staff:** Two checkboxes: "to active" and "to staff".
- Buttons:** "Save and add another", "Save and continue editing", and "SAVE".

In the opened fields, the crucial part is to select the Disposer role.

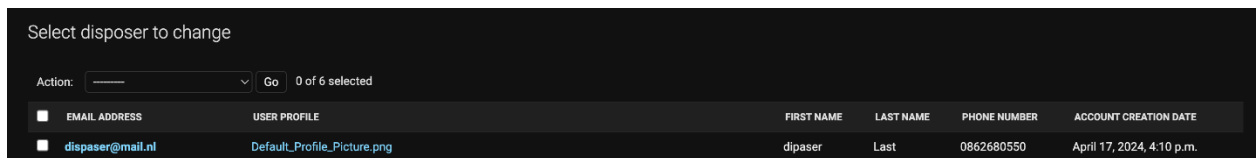
Keep in mind because of the Django specifics mentioned in the report, we did not impose any mandatory fields while creating users internally on the server side.

For disposer, enter email, first, last names, phone number, select role (Disposer), password, password confirmation and is active field must be set to true.

Click save.

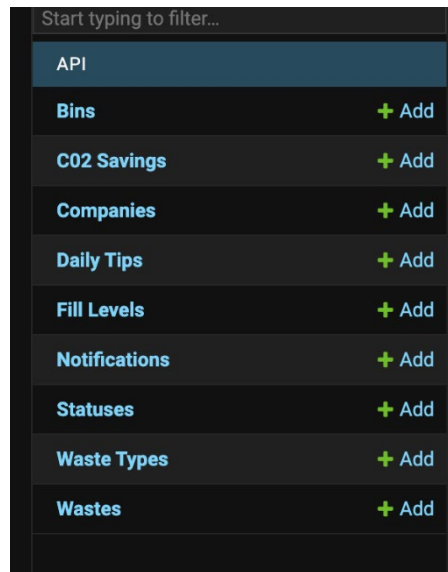


Now if you check the database table or 'authentication_disposer' or click 'Disposers',



you will see the created disposer. By clicking the record, it can be modified.

Company



In order to create a company, press 'Add' for 'Companies'.






The name and unique identifier fields must be unique.

The point system is configured under the company entity. By default, for every type of waste 5 points are assigned.

Keep in mind that the cleaners are assigned to the company via the company entity! You can select as many cleaners as you want. All assigned cleaners will see this very company's assigned bins and supervisors.

Fill in all the fields and press save.

Supervisor

AUTHENTICATION	
Cleaners	 Change
Custom users	 Add  Change
Disposers	 Change
Supervisors	 Change

To create supervisor, click 'Add' for Custom user.

Add custom user

First, enter a username and password. Then, you'll be able to edit more user options.

Email Address:

First name:

User profile: No file chosen

Last name:

Company:

Phone number:

Kvk number:

Tier:

Role:

Password:
Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

Password confirmation:
Enter the same password as before, for verification.

Is active

Is staff

For supervisor, the following fields are being used:

email, first, last names, user profile, phone number, kvk number, role (Supervisor), password, confirm password and administrative fields (is active, is staff).

Note: Company field in Custom User has no affect for Cleaner and Supervisor roles! It is an old legacy field; it is left so not to break existing web dashboard.

Make sure to choose Supervisor role and tick the is active field.

When fields are filled in, click 'save'.

The screenshot shows the 'AUTHENTICATION' section of an admin dashboard. It contains four main categories, each with a 'Change' button:

- Cleaners** (Change)
- Custom users** (+ Add, Change)
- Disposers** (Change)
- Supervisors** (Change)

Below these is a 'Select supervisor to change' section. It features a search bar, an 'Action' dropdown, and a table of supervisors. The table has columns for EMAIL ADDRESS, NAME, COMPANY, KVK NUMBER, PHONE NUMBER, USER PROFILE, ASSIGNED CLEANERS, and ASSIGNED BINS. Two supervisors are listed:

EMAIL ADDRESS	NAME	COMPANY	KVK NUMBER	PHONE NUMBER	USER PROFILE	ASSIGNED CLEANERS	ASSIGNED BINS
supervisor2@gmail.com	Super2 Visor	El Nino	2131231231	+31862680550	Default_Profile_Picture.png	Ilja Kasparov - cleaner1@mail.ru	F55, F2, Jhjih, hamza_bin_1
supervisor@gmail.com	Visor Super	Actfact	1321321321	+31686164670	uploads/supervisors/supervisor@gmail.com/Ellipse.png	Ilja Kasparov - cleaner1@mail.ru	J5, F17, H1, J1, J3

On the right, there is a 'FILTER' sidebar with options to filter by company (El Nino, Actfact) and by first name (Super2).

If you check for created supervisors in 'Supervisors' section or in 'authentication_supervisor' database table, you will see the created record. By selecting the record in admin dashboard, it can be modified. In order to assign a supervisor to a company or change their profile picture, you must refer to supervisor object itself from now on.





The 'Change supervisor' form shows the following details for the selected supervisor:

- Super2 Visor - supervisor2@gmail.com** (with a HISTORY button)
- Email Address:** supervisor2@gmail.com
- First name:** Super2
- Last name:** Visor
- Kvk number:** 2131231231
- Phone number:** +31862680550
- User profile:** Currently: Default_Profile_Picture.png (with a Clear button). Change: Choose file (No file chosen)
- Company:** El Nino (with a dropdown arrow, plus, minus, and refresh icons)

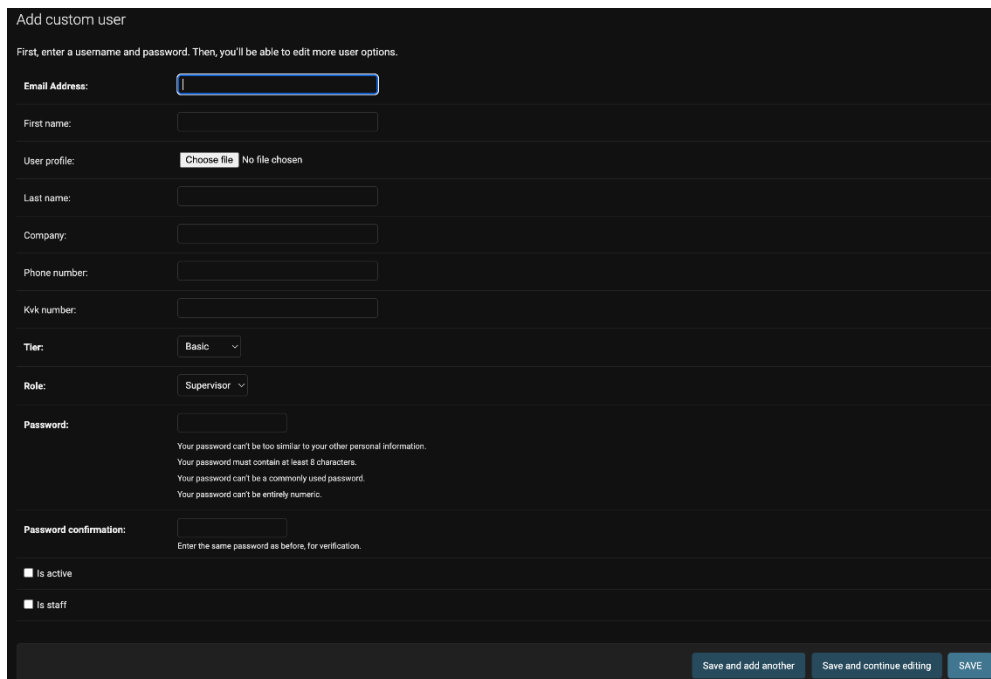
At the bottom, there are three buttons: Delete, Save and continue editing, and SAVE.

Select a company from a drop down under the 'Company' field. Make sure the companies are created prior to trying to assign a supervisor to them. Click 'save'.

Cleaner

AUTHENTICATION	
Cleaners	 Change
Custom users	+ Add  Change
Disposers	 Change
Supervisors	 Change

To create a cleaner, click 'Add' for 'Custom User'.



Add custom user

First, enter a username and password. Then, you'll be able to edit more user options.

Email Address:

First name:

User profile: No file chosen

Last name:

Company:

Phone number:

Kvk number:

Tier:

Role:

Password:
Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

Password confirmation:
Enter the same password as before, for verification.

is active

is staff

For cleaner creation, the following fields are relevant:





Email, user profile, first, last names, phone number, kvk number, role (Cleaner), password, password confirmation, administrative fields (is active, is staff).

Note: Company field in Custom User has no affect for Cleaner and Supervisor roles! It is an old legacy field; it is left so not to break existing web dashboard.

Make sure to select cleaner role and tick the 'is active' field.

When all fields are filled in, click 'save'.

AUTHENTICATION

- Cleaners**  [Change](#)
- Custom users** [+ Add](#)  [Change](#)
- Disposers**  [Change](#)
- Supervisors**  [Change](#)

Once cleaner is created, press 'Cleaner' to see all created records.

Select cleaner to change

Action: Go 0 of 2 selected

<input type="checkbox"/>	EMAIL ADDRESS	NAME	SUPERVISOR ASSIGNED	ASSIGNED BINS	ASSIGNED COMPANIES	KVK NUMBER	USER PROFILE	ACCOUNT CREATION DATE
<input type="checkbox"/>	cleanercool@mail.com	Clean Proper	-	-	-		Default_Profile_Picture.png	April 17, 2024, 4:22 p.m.
<input type="checkbox"/>	cleaner1@mail.ru	Ilija Kasparov	Super2 Visor - supervisor2@gmail.com	F55 - USA , F2 - Boat , Jhijh - USA , hamza_bin_1 - morocco	El Nino , Actfact	1321321321	uploads/cleaners/cleaner1@mail.ru/sunset.jpeg	March 18, 2024, 6:54 p.m.

2 cleaners

Once a record is pressed itself, you can modify it: change fields, image.

Change cleaner HISTORY

Clean Proper - cleanercool@mail.com

Email Address:

First name:

Last name:

Kvk number:

Phone number:

User profile: Currently: Default_Profile_Picture.png Clear
 Change: No file chosen

In order to assign companies to a cleaner, refer to the Company section.

Keep in mind that cleaners can be assigned to many companies.

Bin

In order to create a bin, refer to this section.

API	
Bins	+ Add
C02 Savings	+ Add
Companies	+ Add
Daily Tips	+ Add
Fill Levels	+ Add
Notifications	+ Add
Statuses	+ Add
Waste Types	+ Add
Wastes	+ Add

Click 'Add' for Bins.

Add bin

Bin location:

Is garfill

Capacity:

Unique Identifier:

Last emptied date: Date: Today | 📅
Time: Now | 🕒
Note: You are 2 hours ahead of server time.

Image google drive link:

Bin fill level:

Longitude:

Latitude:

User:

- cleanercool@gmail.com
- disposer@gmail.nl
- disposertest3@gmail.com
- disposertest2@gmail.com
- disposertest@gmail.com
- supervisor@gmail.com
- supervisor@gmail.com
- cleaner1@mail.ru
- rlshncsr@mail.com

Company: + Add

To create a bin, fill all the required fields.

Keep in mind: User field is referring to Custom User objects (not to Disposer, Supervisor, Cleaner). It is an old legacy reference of a bin to a Custom User, which is used on web dashboard.

Select a user you are creating a bin with.

'Longitude' and 'Latitude' are used to indicate the exact coordinates of the location of a bin.

A company to which a bin belongs to is selected from a drop down under 'Company' field.

When all the fields are filled in, click 'save'.

QR code creation

After a bin is registered on the system, the QR code can be generated.

The following link must be encoded:

<domain name>/api/scan/<bin id>

To get the bin id, look it up in the database. Look for 'api_bin' table and pick the correct id for the bin you want to generate QR code for.

	id	bin_location	is_garfill	capacity	unique_identifier	last_emptied_date	image_google_drive_link	bin_fill_level	company_id
1	7	USA	false	100	F55	2023-05-02 17:11:00.000000 +00:00	none		77
2	3	Boat	true	21	F2	2023-04-15 19:20:49.000000 +00:00	none		77
3	6	USA	true	100	Jhijh	2023-04-13 22:40:30.000000 +00:00	none		77
4	4	morocco	true	100	hamza_bin_1	2023-04-14 00:48:46.000000 +00:00	none		77
5	2	Kitchen	true	9	F1	2023-04-16 11:49:00.000000 +00:00	none		77
6	10	Donijn	false	280	J5	2023-09-25 11:30:41.000000 +00:00	none		77
7	9	Morocco	false	100	F17	2023-06-06 21:05:15.000000 +00:00	none		77
8	11	NetherLands	false	120	H1	2024-02-21 09:26:46.000000 +00:00	none		10
9	1	PLAEX Office	false	160	J1	2023-04-17 11:40:22.000000 +00:00	https://drive.google.com/embeddedf..		77
10	8	PLAEX Office	false	160	J3	2023-05-09 17:36:13.000000 +00:00	https://drive.google.com/embeddedf..		77

Example of the link you want to encode:

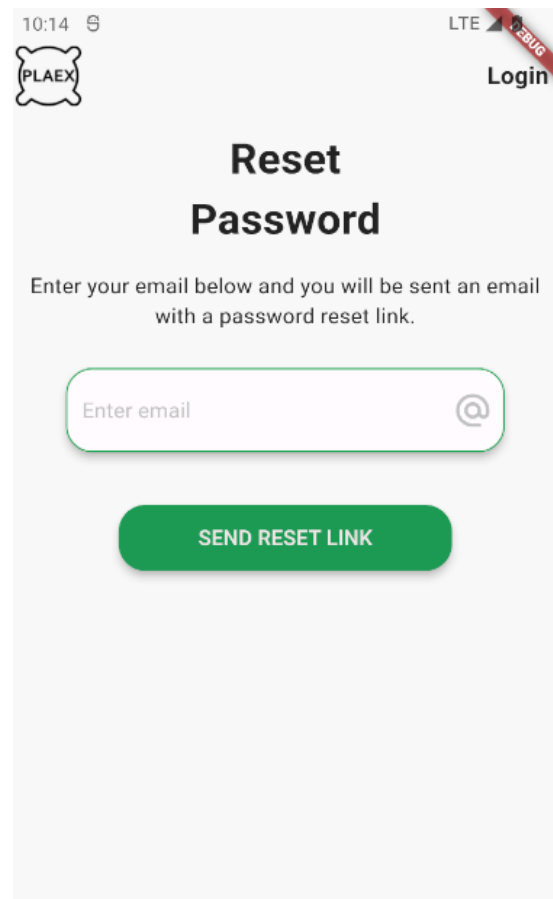
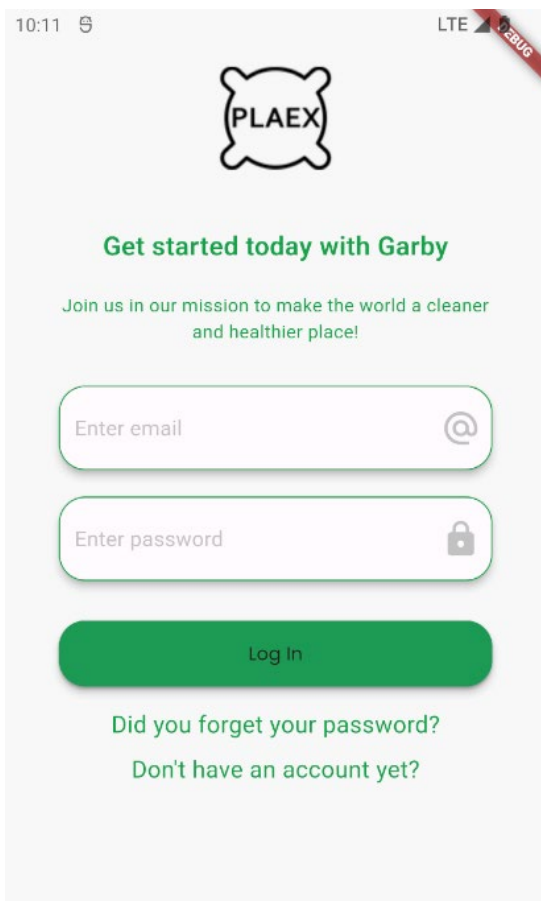
www.plaex.net/api/scan/10

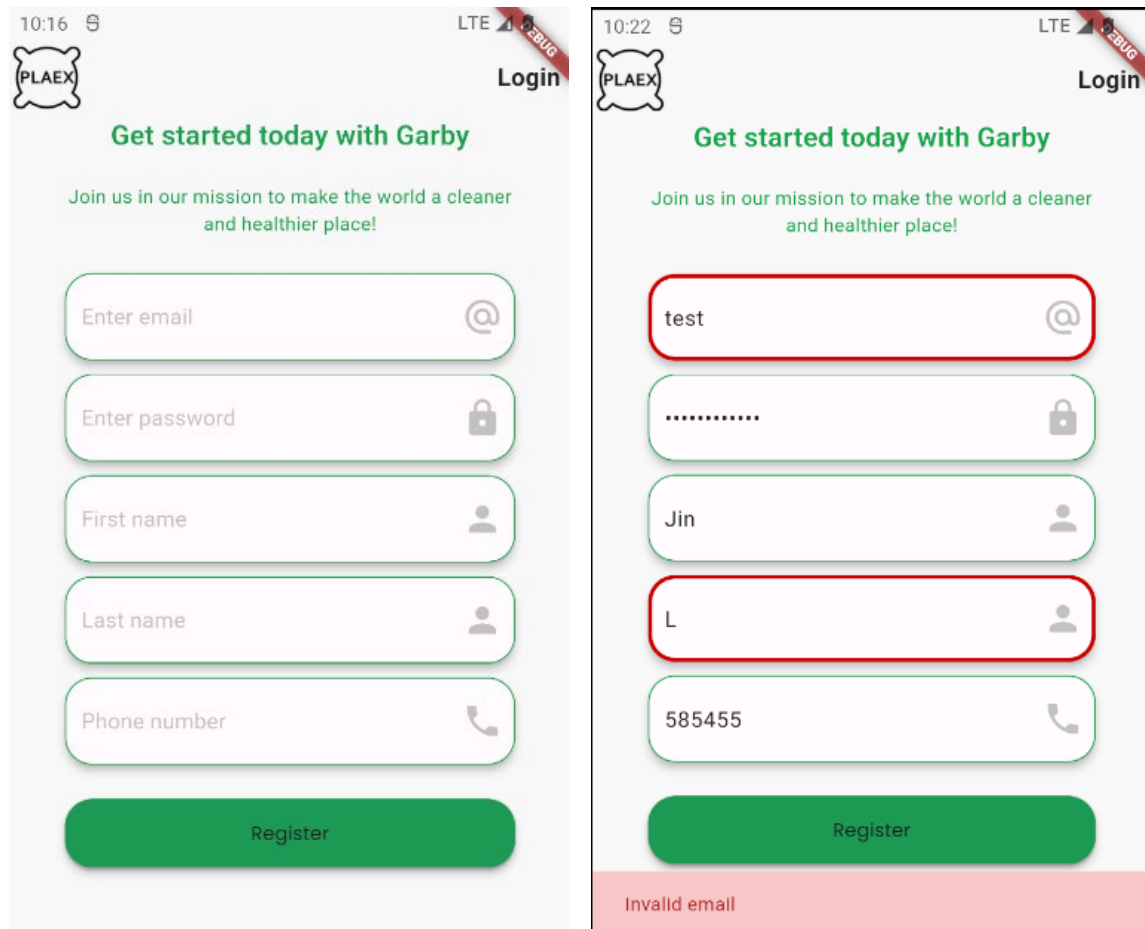
Using the apps

Detailed information on how to set-up the Flutter workspace for the apps is covered in the documentation of the code in the markdown files. This section will describe how to use the three apps. Authentication will only be covered in the Disposer section since it is the same for all three apps, except that the disposer role has the create account functionality. The profile screen, settings page and map are similar for all three apps.

Disposer

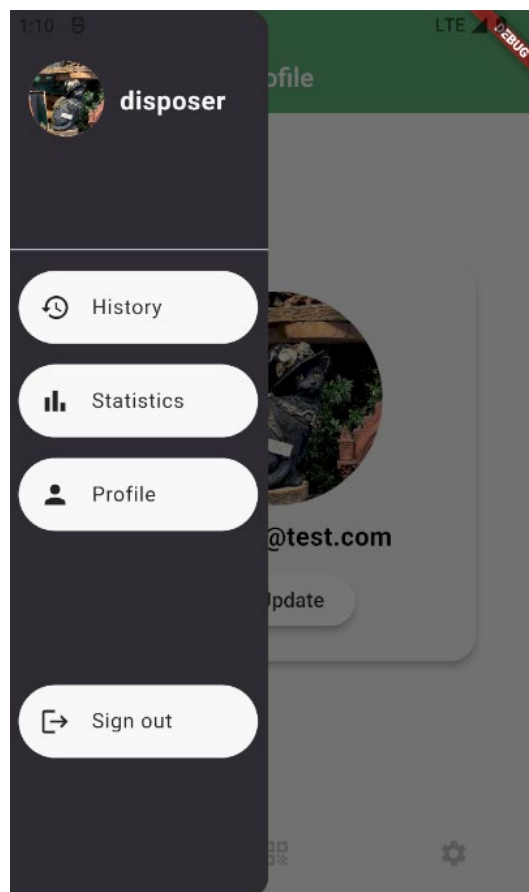
Login





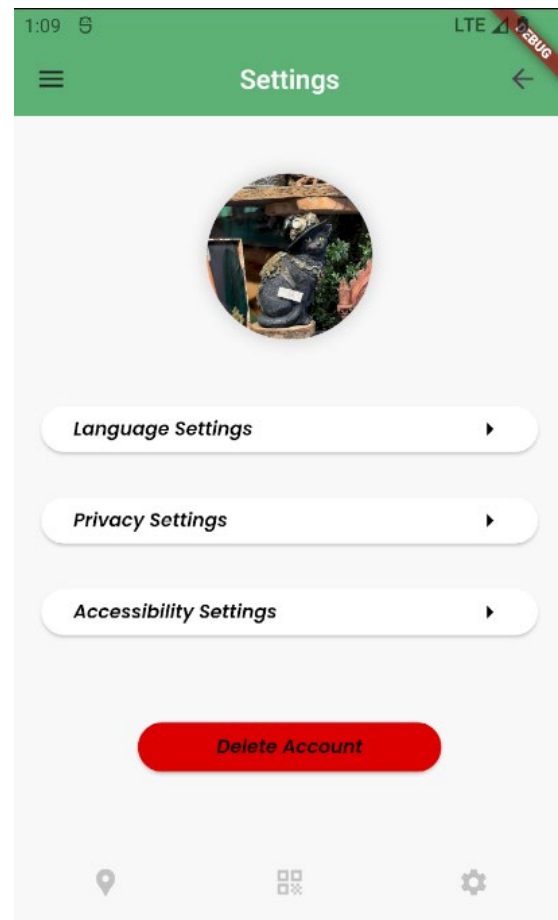
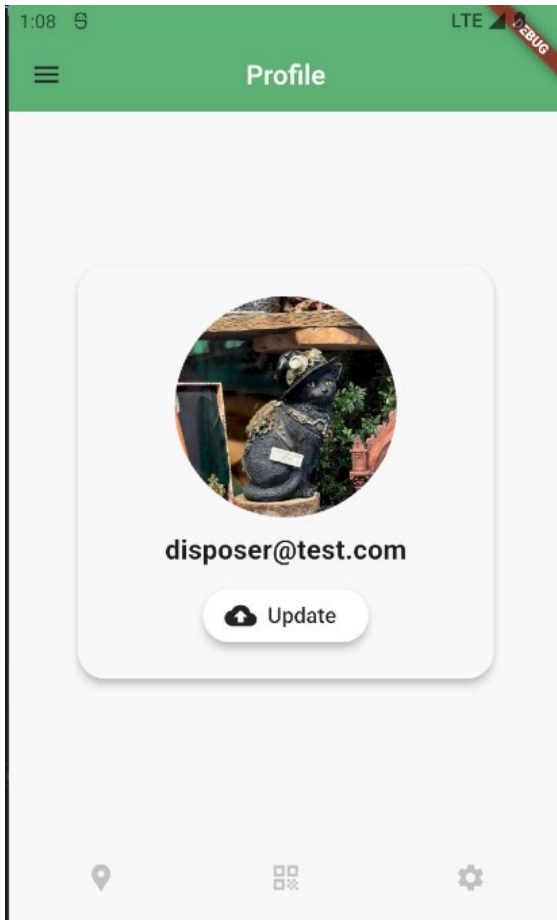
The Login page contains only email input and password input. Under the login form, there are two text links directing the user to the Reset Password and Create Account screens. The Reset Password screen only takes one input from the user and that is their email address. The Create account screen has several input fields and each of them has an individual validation method. The email needs to contain the '@' character, the password must be 6 characters at least, the first and last name must be at least 2 characters each and the phone number 6 digits. For the phone number a digit keyboard will be shown on input action. Each field validates its contents individually as can be seen from the second picture above. A suggestive message is displayed for the user when trying to log in with invalid fields. Once successfully created an account, the user is redirected to the Login page.

Disposer Navigation Drawer



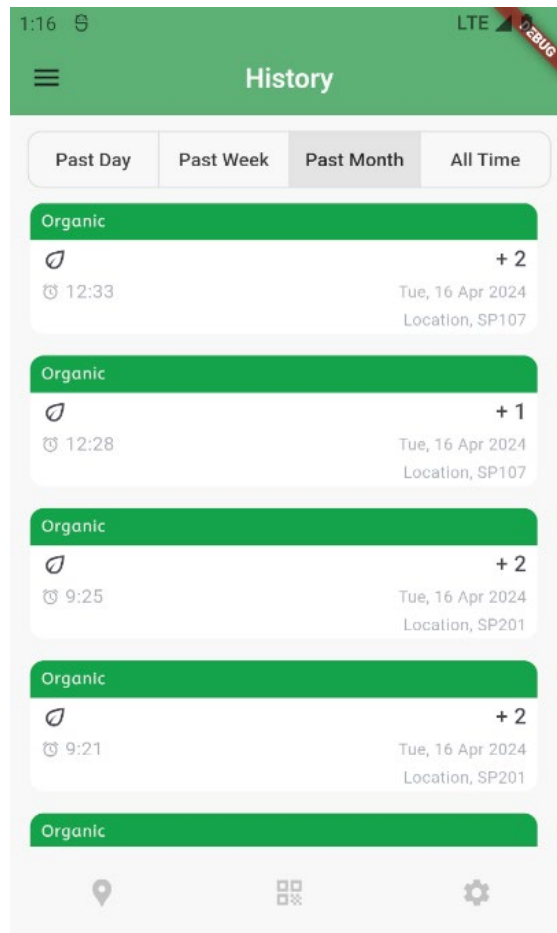
The Navigation Drawer for the disposer role has buttons to redirect the user to the History screen (which is also the home page of the disposer app), Statistics screen and Profile (which can also be accessed by tapping the profile picture (valid for all three roles)).

Profile and Settings



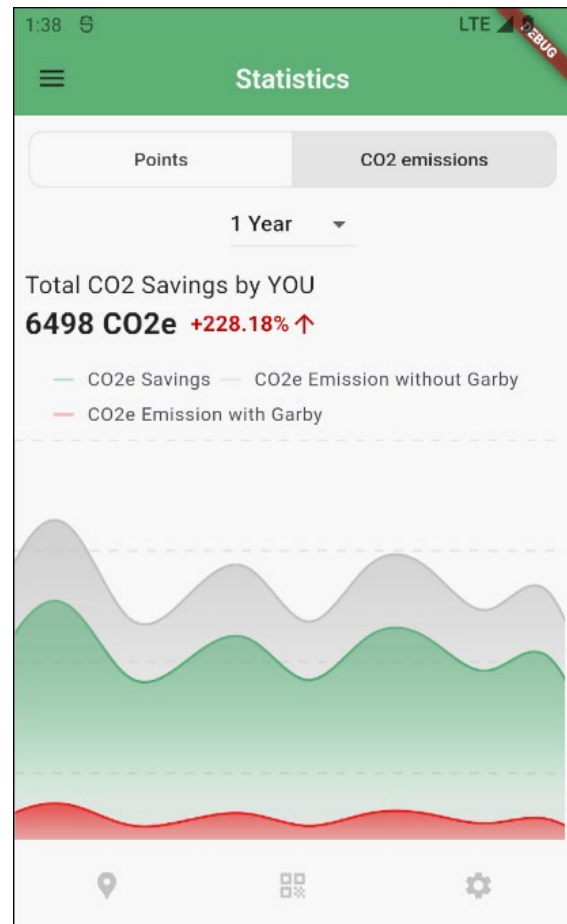
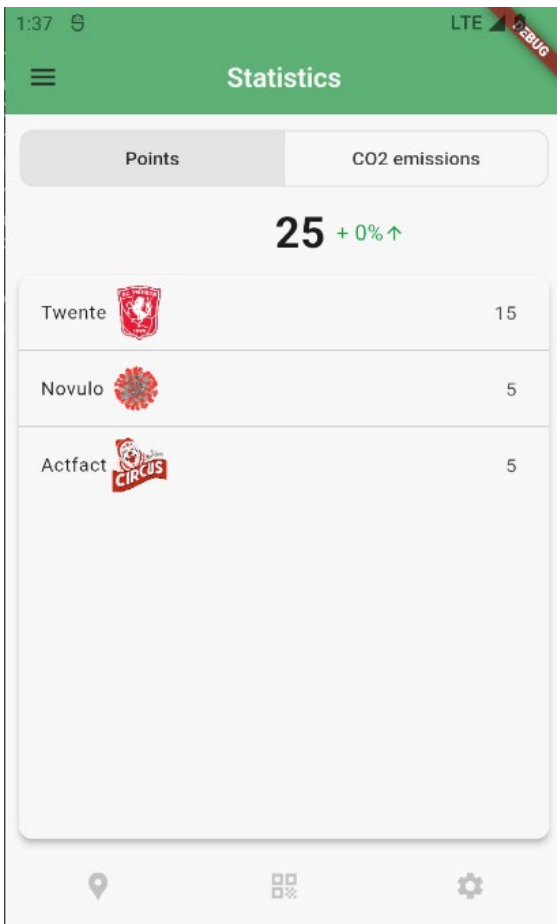
The profile screen can be accessed from the navigation drawer and the settings page from the bottom navigation bar. The profile picture can be changed using the 'Update' button. In the settings screen only the 'Delete Account' feature is functional. The other three buttons are left to be linked to your implementation based on your preferences. They can easily be replaced in the future or removed.

History



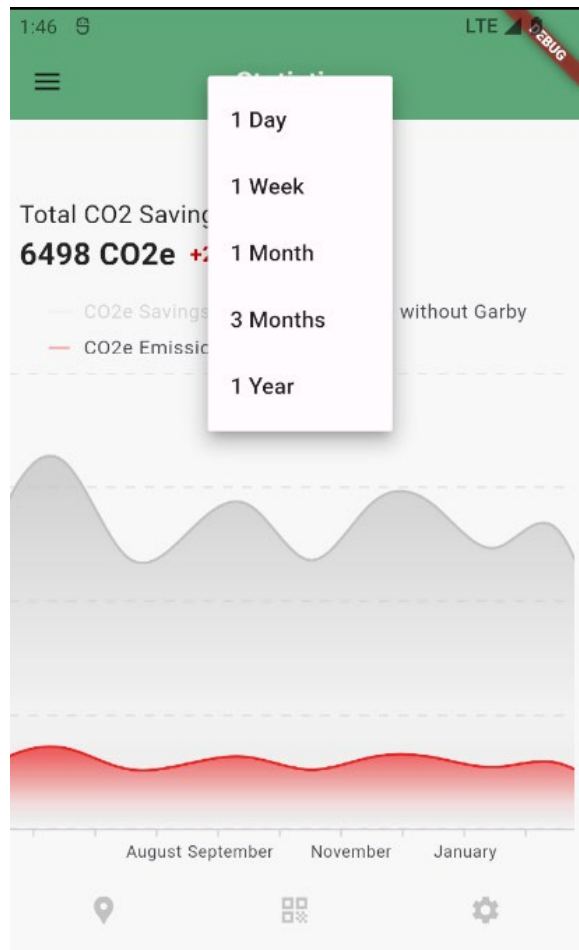
The history screen is the home page of the disposer app. The screen has time selection buttons at the top. The default selection of this page is 'Past Day'. This page always fetches only the first 10 entries and by scrolling to the end of the list the next 10 entries will be fetched to avoid waiting times for the user to receive an exceedingly long list of entries. Each entry will show the icon of the waste type, the date, location, and the number of points received. If there are no points received 'NP' will be shown instead.

Statistics

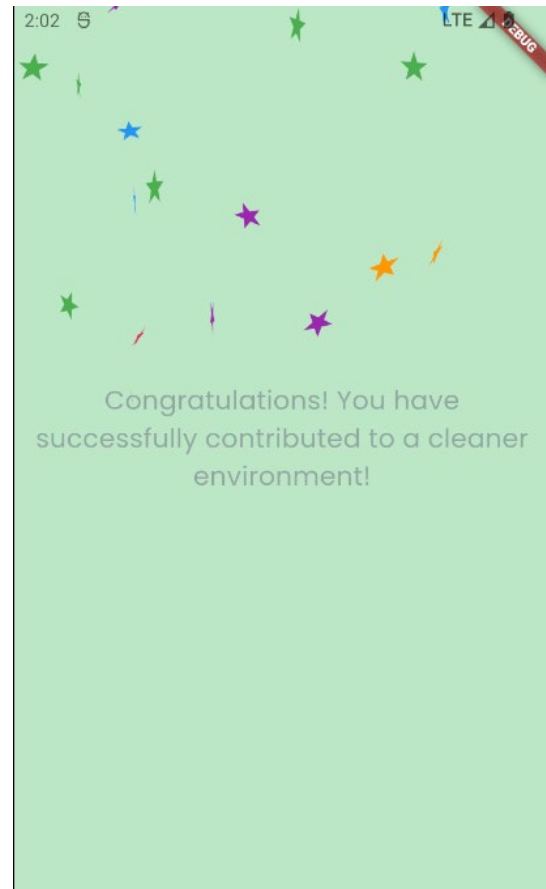
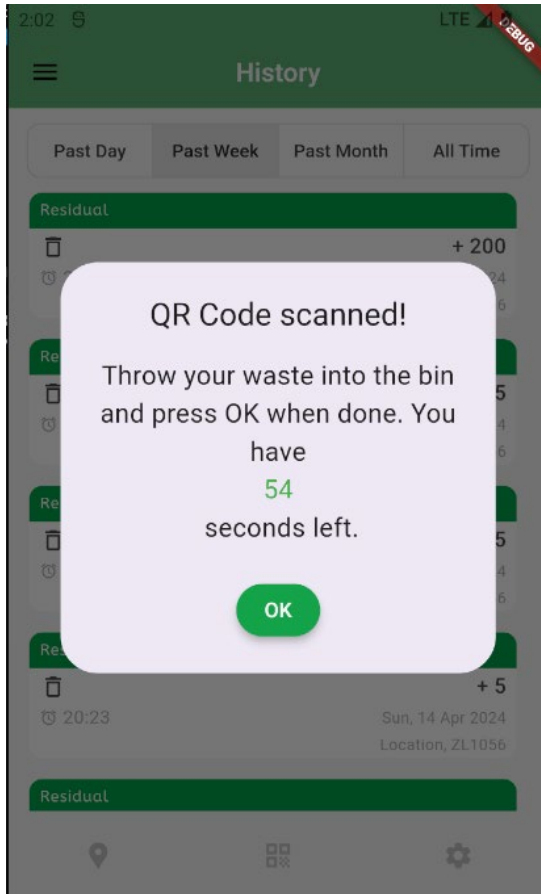


The statistics screen has two parts: points and CO2 emissions. There is a button to select the view at the top. The points view will show the total number of points gained by the disposer together with the percentage increase from the past week. A sorted list (from most points to least points) is shown underneath which pictures the companies where the user gained points and the total number of points gained at those companies.

The CO2 emissions view has a dropdown with a time selection ranging from day, week, month, 3 months, year. The time can also be changed by swiping left or right on the chart itself. The total number of emissions for the selected period will be shown together with the percentage increase. You can zoom in or out by pinching with your fingers on the chart or zoom in by double tapping. When you zoom in the chart also becomes scrollable on the horizontal axis. When you enter the screen all 3 measurements are shown in different colors (CO2e savings, emissions without Garby and with Garby), but you can deselect them by clicking on the legend item and they will disappear from the chart for a better view of a single metric.



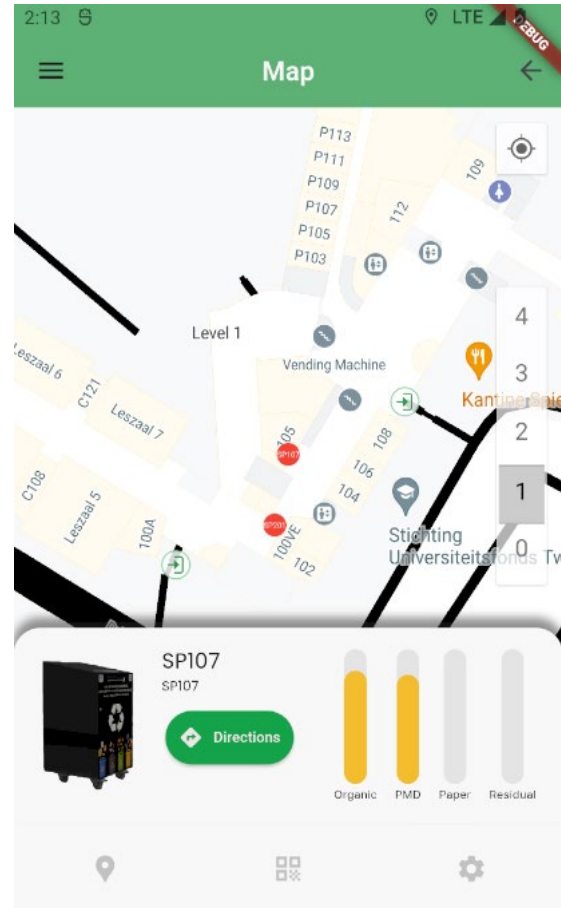
QR Scanning



The camera to scan a QR code can be accessed from the QR code icon in the bottom navigation bar. Once a valid QR code is scanned, the user will see a dialog box which shows them a 60 second timer indicating that they need to throw their waste within that amount of time. Once they are done throwing and click 'OK' or the time has passed, they will be encountered with a celebration screen which throws confetti, plays a sound and shows a suggestive message. The celebration screen takes 10 seconds to disappear, and the user will be redirected to the last screen visited.

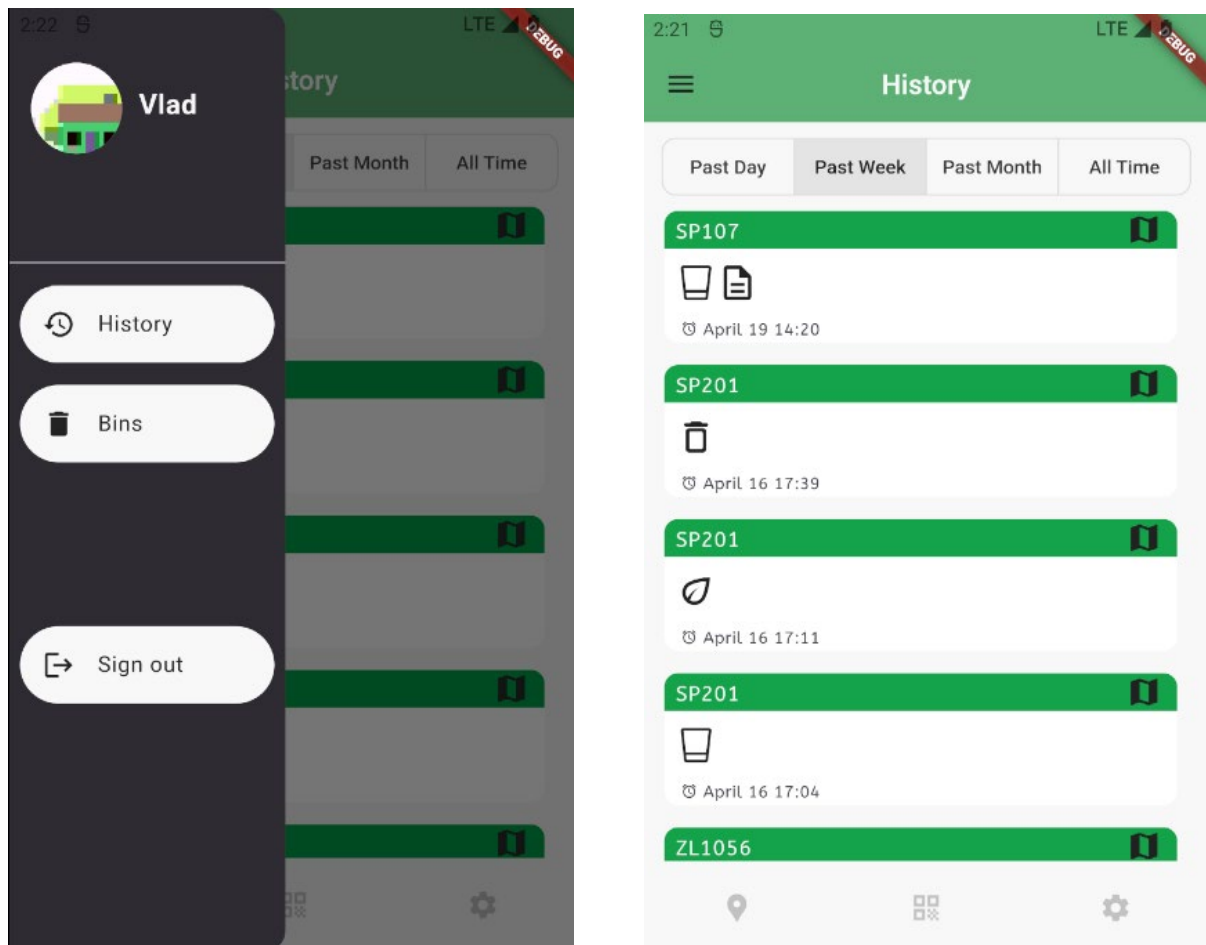
Cleaner

Bin Status and Map



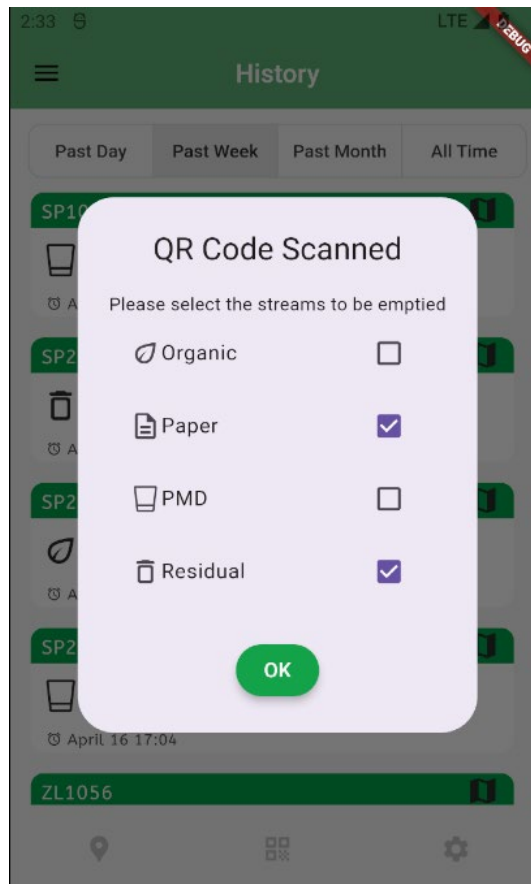
The bin status screen is the home page of the cleaner app. It will display all the bins from all the companies where they are registered. Each bin's icons are responsive to change depending on the fill level, if they are almost empty, they are green, if they are almost full or full they are red and if they are in between, they are yellow. Tapping the dropdown arrow will extend the tab and display the exact fill levels. Tapping the map button on a bin will open the map and navigate to the location of that bin on the map. If you tap a bin on the map, you will be presented with an overview of the fill levels again. On this overview there is a button called 'Directions' which once tapped will open Google Maps and create a route to that bin.

Navigation Drawer and History Screen



The history screen can be accessed from the navigation drawer. The History screen will open by default in the 'Past Day' time selection. It will fetch the first 10 entries and by scrolling to the end of the list it will fetch the next 10. Each entry represents a moment in time when a bin was emptied, the name of the bin together with a button to navigate to the bin on the map, and the icons representative of the streams of the bin that were emptied at that time.

QR Scanning

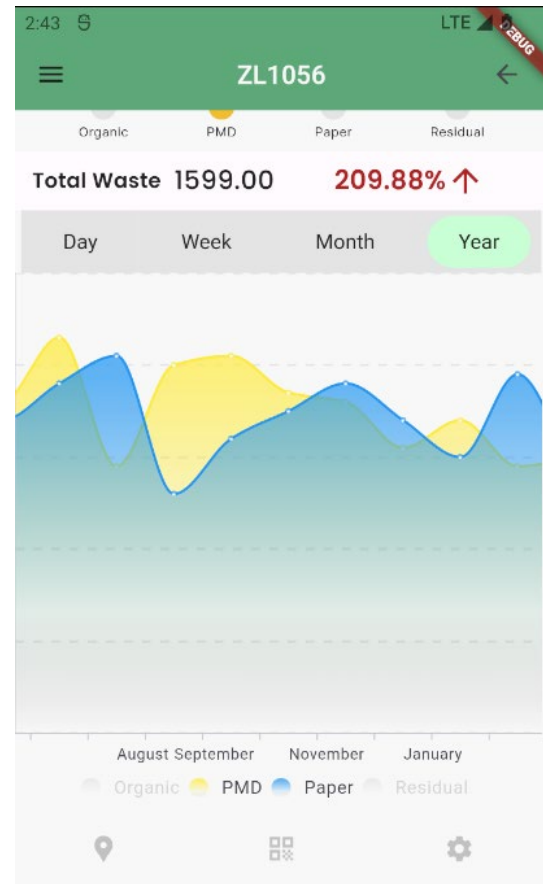


After scanning the QR code of a bin, the cleaner will encounter a dialog box in which they need to select the streams that they want to empty. You cannot select none of the options and then press 'OK', the app will not allow you to. You can close the pop-up window by tapping outside it, but that will not save an emptying action. If you select at least 1 stream and then submit the form, the emptying action will be saved and by refreshing the page shortly after you can already see the action in the list.

Supervisor

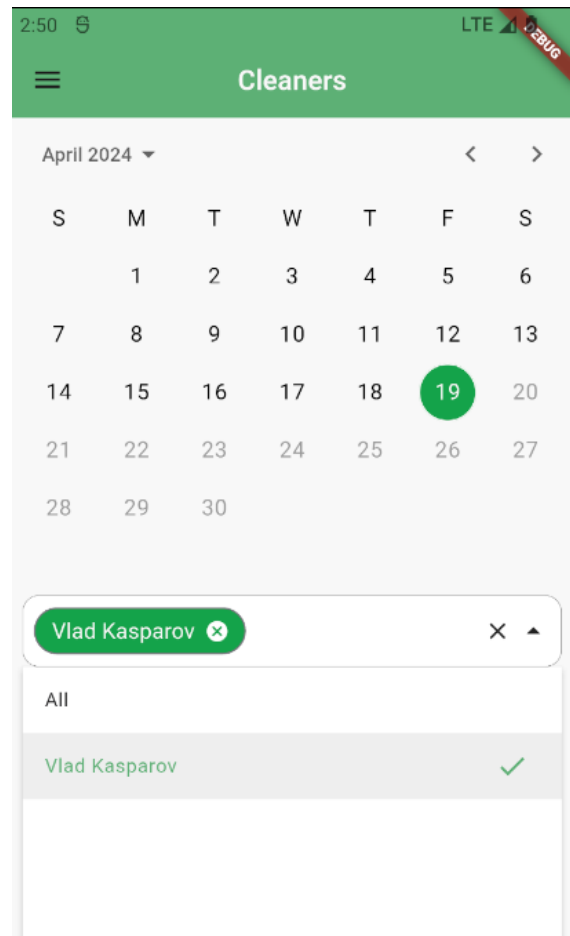
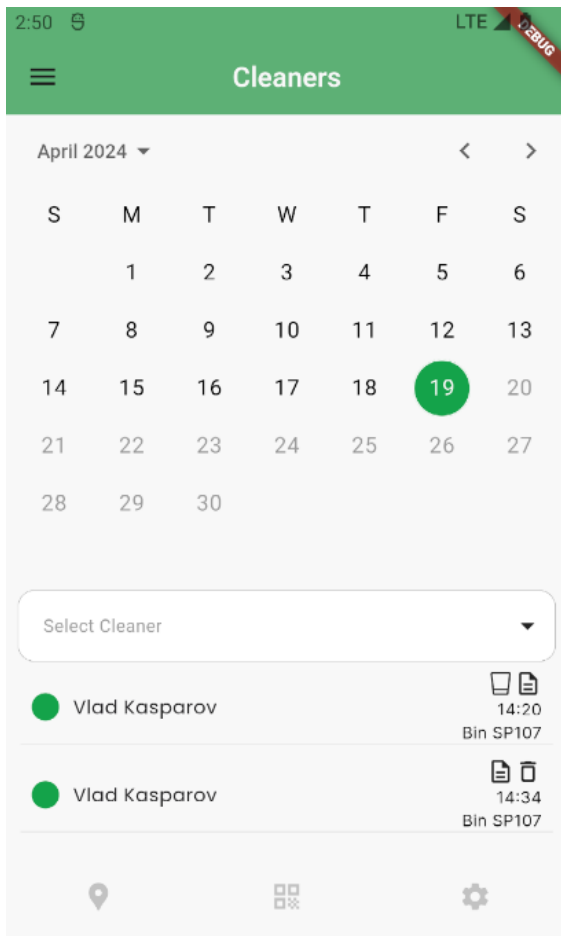
For the supervisor role, the main screen is the 'Bin Status' screen which is the same as for the cleaner. The map and route to the bin works in the same way. The only difference is that for the supervisor, if you tap a bin entry it will redirect you to a screen with an overview over that specific bin.

Bin Overview Screen



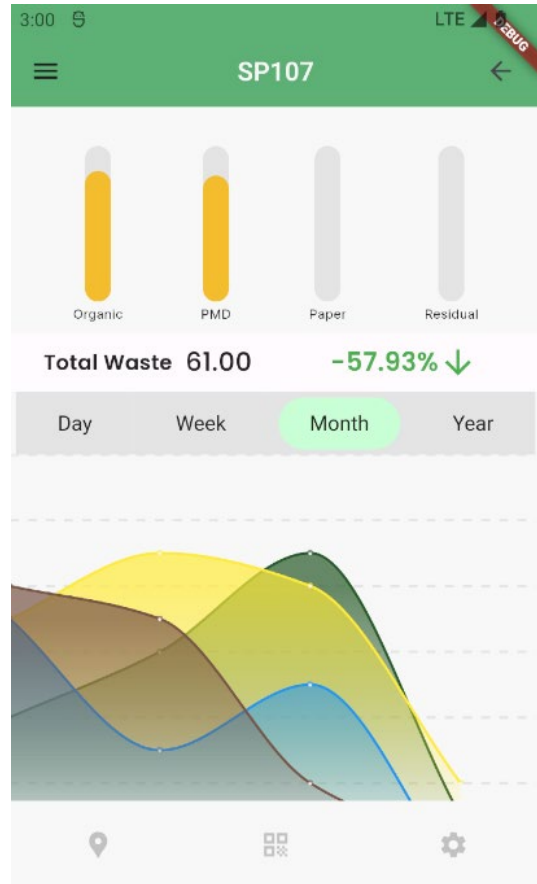
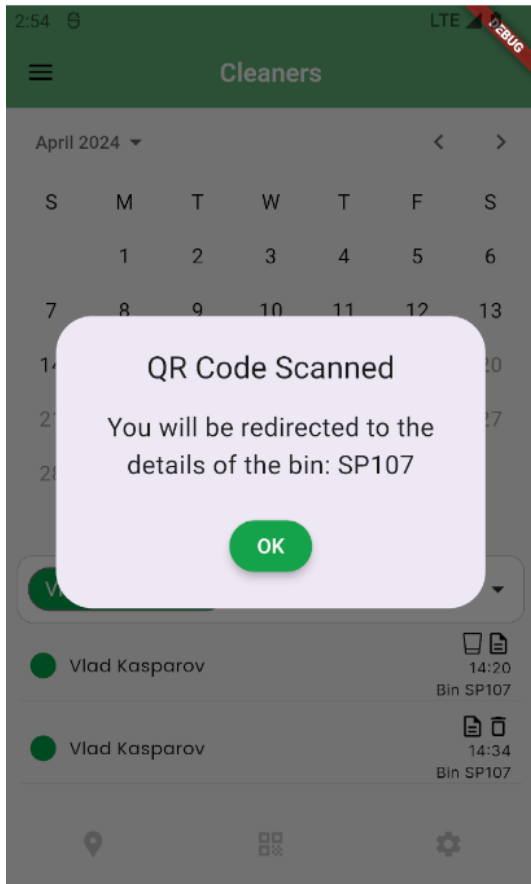
The bin overview screen will show the details of the bin from which you navigated here. The current fill levels are shown together with the total waste thrown in that bin for the selected time frame and the increase percentage. Underneath there is a chart showing the total waste per type of waste. By default, it will show all 4 types of waste, but you can deselect any and the chart will only show your selection. You can zoom in the chart by double tapping or pinching.

Cleaners



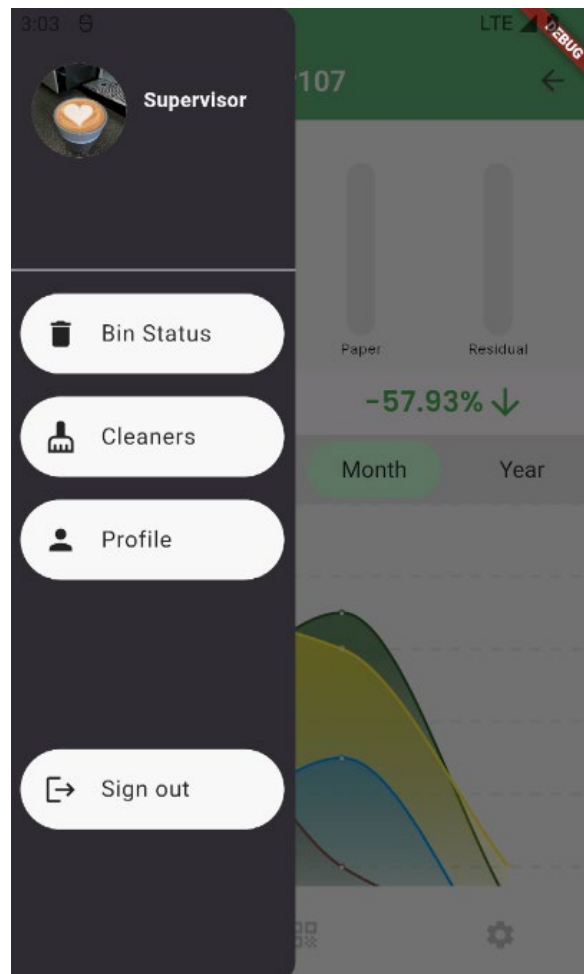
The cleaners page can be accessed from the navigation drawer. The screen opens with a calendar showing the current date and the emptying actions done by all the cleaners on this day. A dropdown is present to select a certain cleaner or a group of cleaners to be displayed. If you change the selection only the cleaners you selected will be displayed on any date you choose. Each entry will display the cleaner, the emptied streams, the time, and the bin.

QR Scanning



When you scan a bin as a supervisor it will redirect you to the overview screen of that bin. If you just received the bin and it is not yet assigned to the company, when you first scan the bin, it will also automatically assign it to the company you are a supervisor for and then redirect you to the details of the bin. If you don't see the bin directly in the bin status screen list, you may need to refresh the page and it will appear in the list.

Navigation Drawer



The navigation drawer of the supervisor contains buttons to redirect the user to the bin status page, the cleaners page, and the profile page.

Setting up the Flutter environment for the apps

All the information needed to set up the Flutter environment, build APKs, deploy the apps and make changes is documented in the code itself. The repository of the apps contains markdown files with instructions and references for all the steps needed.