Food Waste

Arentas Meinorius,
Jaunius Tamulevičius,
Martinas Mačernius,
Pijus Petkevičius

Matematikos ir informatikos fakultetas
Vilniaus universitetas
Lietuva
April 6, 2022

Summary

The primary objective of the second laboratory assignment is to design the system and required changes. While in the first laboratory work we analysed business and all its processes, this time the attention on existing system and the changes.

The main tasks of our project are:

1. Use UML 4+1 framework for organizing the architecture document.
2. Implement the planned changes in the system.
3. Introduce a CI/ID process.

Contents

1 Context

For a system to be successful, it must be developed with the intention of solving a real-world problem, which, in our case, is reducing food waste in restaurants and shops. The software is useless if it does not solve required problem. In this part we analyse our problem and how it is intended to be solved.

1.1 Goal of the system

Reduce food waste by distributing it.

1.1.1 The problem

Not all food products are sold before spoiling, sometimes restaurants do not use all the food they have bought.

1.1.2 Solution

Prepare a plaftorm that would stand as a middle man helping people sell excess food while allowing others to buy it cheaper.

1.1.3 Main User Goals

Monetary gain:
+ Providers: utilising over-booked food
+ Users: acquire food cheaper
Reducing food waste

1.2 Planned changes

To further develop and increase the functionality of the existing system we were given several tasks of implementing changes and features. The improvement consists of adding food status to facilitate consumers lifes and by adding restaurant/grocery store edit option.

1.2.1 Change list

- Product status implementation, when it is bought etc.
- Restaurant accounts should be able edit their data.
- Add tech support.

1.2.2 Impact of changes

The newly added functionality of ordering products will help automate the process of food receiving. Users will no longer need to personally contact food owners or have cash prepared as transactions will go online to payment provider. Additionally, newly added tech support will be able to help customers with any concerning questions that might arise while using the system. That will further improve the system usability for all types of users. Finally, the ability for restaurants to easily edit their data should provide better up to date information which is always important to avoid misunderstandings with the users.

1.3 Current system analysis

The current system only worked as a showcasing tool for the products, therefore it was hardly usable and inconvenient. The need to manualy contact food owners and negotiate made the process slow

1.3.1 System environment

The main web-application server will be run on a linux machine using IIS Express. The database will be on the same server. Users will be able to connect through any type of modern browser. Support for older browsers (e.g., Internet Explorer) is not included.

1.3.2 Tools and Technologies

The main development tools and technologies, which are .NET Core Software framework, ASP.NET server-side application and razor framework. The database management system chosen by the original developers was Docker and and PostgreSQL. Overall, the tools and technologies are well chosen for the system in development.

1.3.3 Existing problems

User interface is too simplistic and lacks visual representation. There is too much text and zero appeal

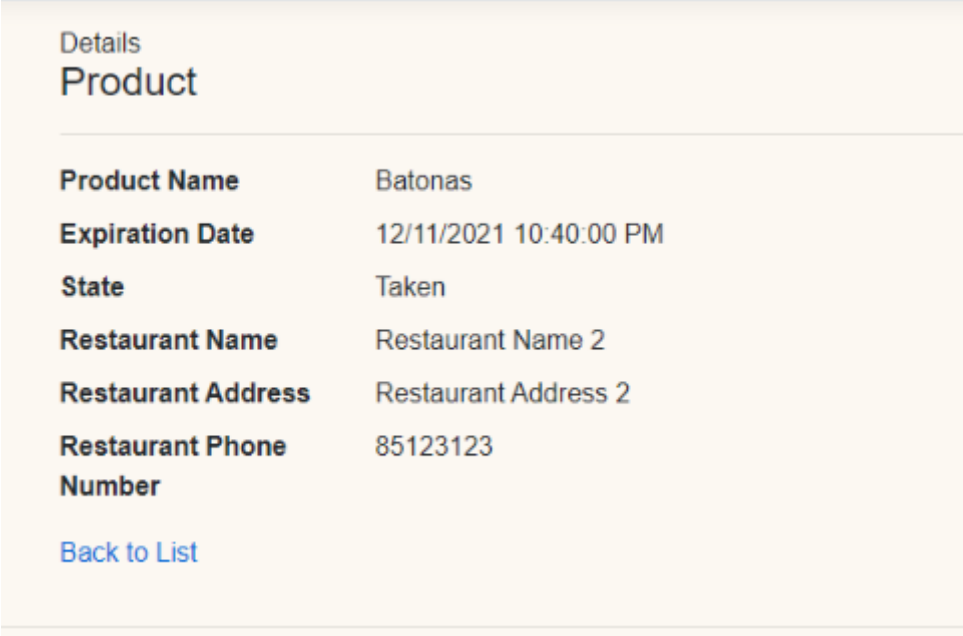

Figure 1: Food waste Product View

1.4 Development environment

Version control systems, play a major role in any modern software development project. This is especially important for us, since out team will mostly work remotely. Our version control system is Git. The source code is hosted on GitHub, because all of the members are familiar with this repository management tool. Following good coding practices, every new feature implementation will be created in a separate branch and reviewed by at least one team member.

2 Development view

The development view illustrates a system from programmer's perspective and is concerned with software management. This view contains:
1. Component diagram

2.1 Component diagram

Component diagram provides high level architecture overview of different components used for operating Food Waste

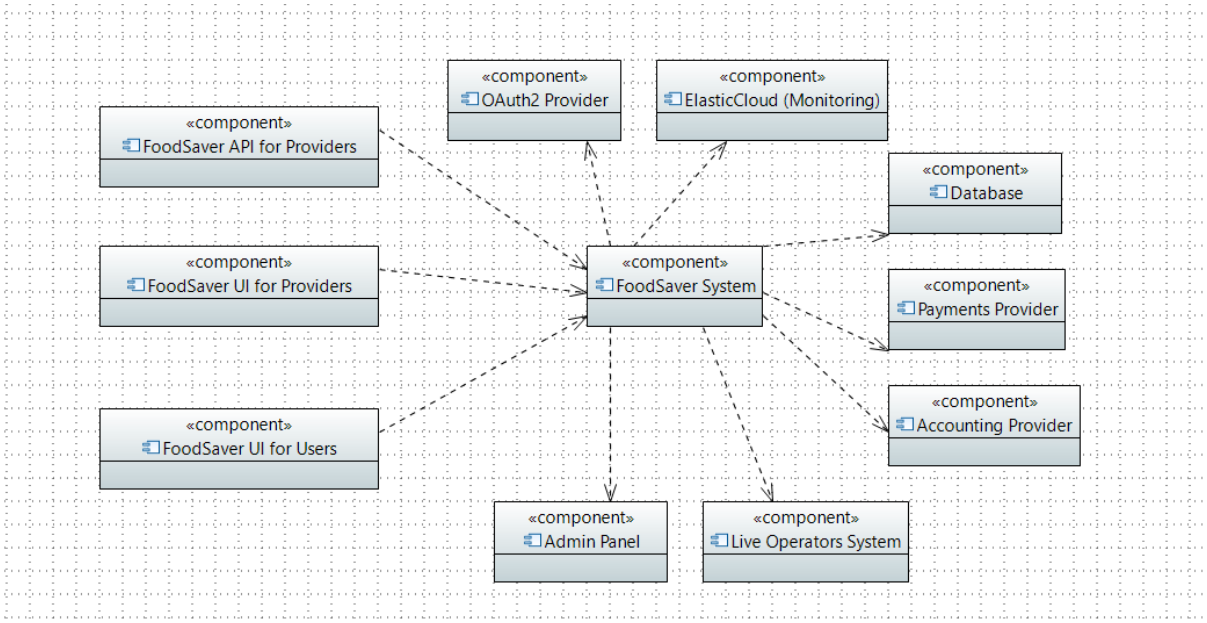

Figure 2: Food waste Component diagram

3 Logical view

Logical view is concerned with the functionality that the system provides to end-users. This will be achieved via these diagrams:
1. Class diagrams
2. State machine diagrams

Each of these diagrams has a separate section in which diagrams itself and descriptions are provided.

3.1 Class diagrams

The class diagram shown belown illustrates our application after the changes. We have included a new functionality of ordering the products and designed tech support. Also we kept in mind the necessity to manage restaurants and added some additional operations. This diagram allows us to implement the changes more easily with its structured view.

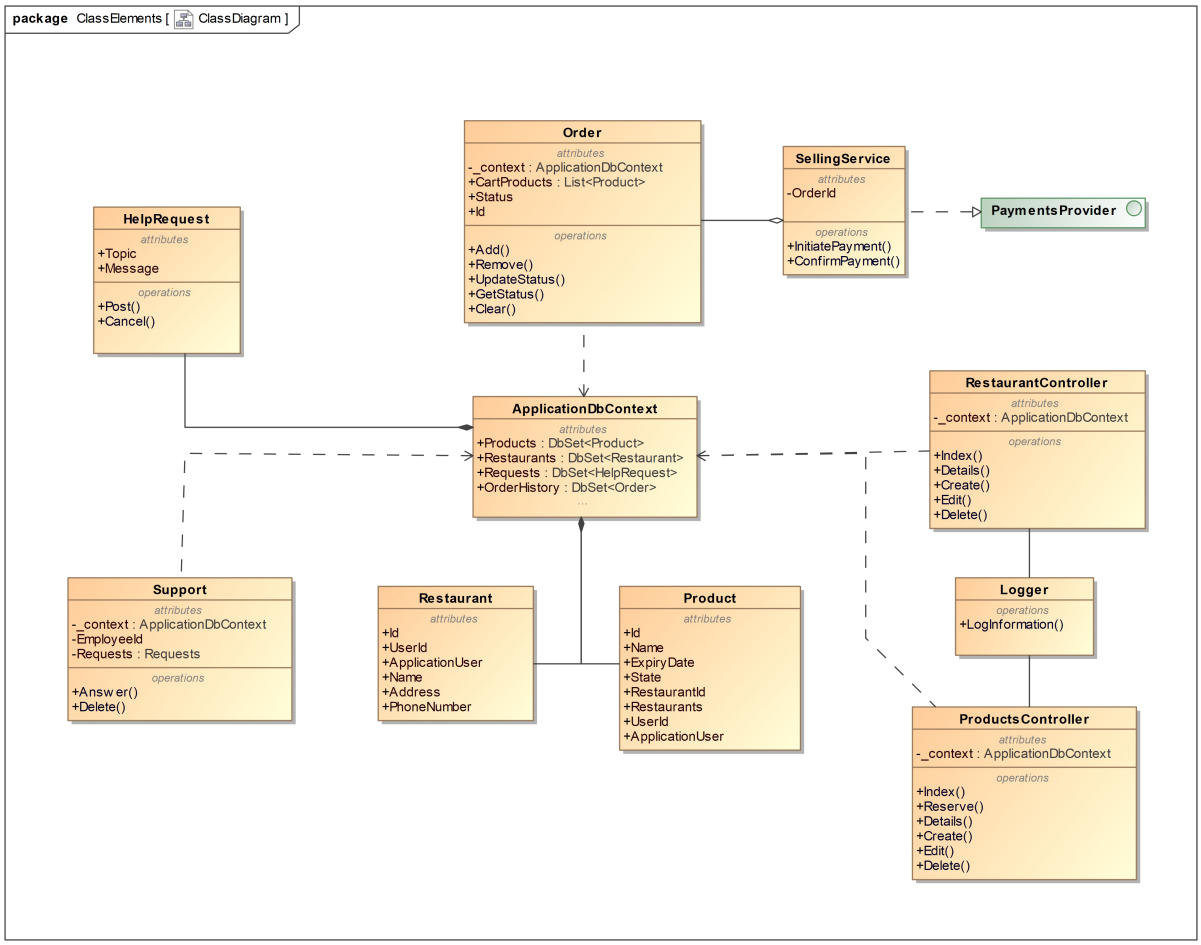

Figure 3: Food waste Class diagram

3.2 State machine diagrams

The state diagram shown below illustrates how our ordering systems works in happy day scenario. We can see that user adds or removes products to his order as he wishes and then proceeds to the payment. Once the payment is started, order status gets updated throughout the process and transaction is being verified. When the payment gets verified and customer gets his bill order is considered to be finished.

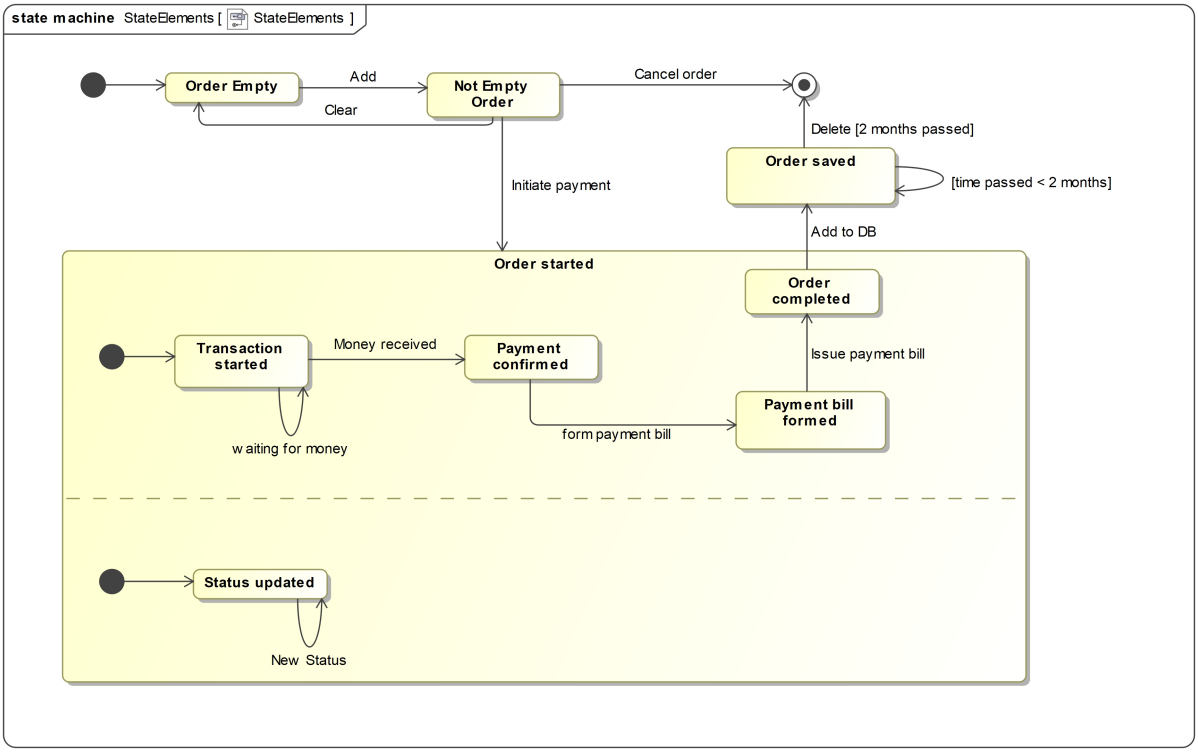

Figure 4: Food waste State diagram

4 Process view

Process view illustrates and explains the system processes. The focus is on their communication and synchronization. This view contains:
1. Communication diagrams
2. Activity diagrams

4.1 Communication diagrams

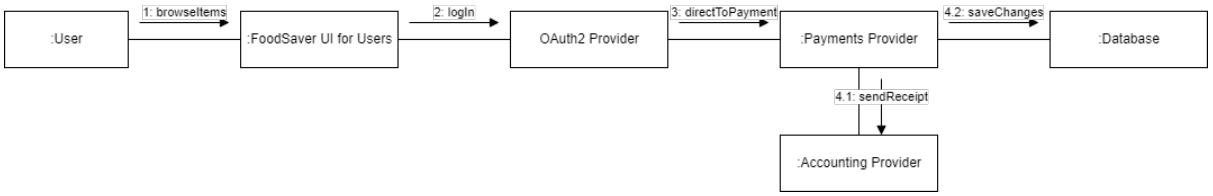

Figure 5: User purchases communication diagram

This communication diagram shows how components are supposed to communicate with one another when a user wants to make a purchase. As you can see, the user can browse available products without logging in, but when he wants to make a purchase he has to provide credentials. After that the user makes a purchase, the database is updated accordingly and the accounting is notified of the transaction.

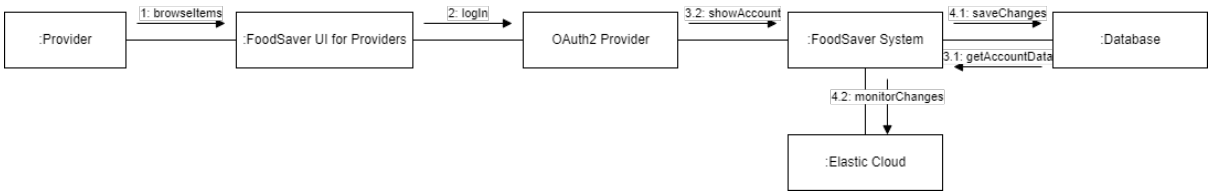

Figure 6: Provider changes communication diagram

The provider needs the possibility to update information about himself, so this communication diagram shows how the process should go. Like a regular user, the provider can freely browse the products, but if he wants to make a change he must log in. After logging in, he is provided data about his account and his supplied products. He can make changes to this information, which is later saved into the data base. All changes are monitored.

4.2 Activity diagrams

Tech support until this point was virtually non-existent, so there is not much to compare it to. The diagram describes how a user (in this case a logged in provider) should deal with a system error that does not allow him to properly continue his work. As is shown in the diagram, after an error the system logs the circumstances under which the error occurred and asks the provider if he wants to issue a ticket. If he does so, the tech support personnel review the problem, communicate with the provider and fix a problem that the person is having. After all this, the provider can return to his work.

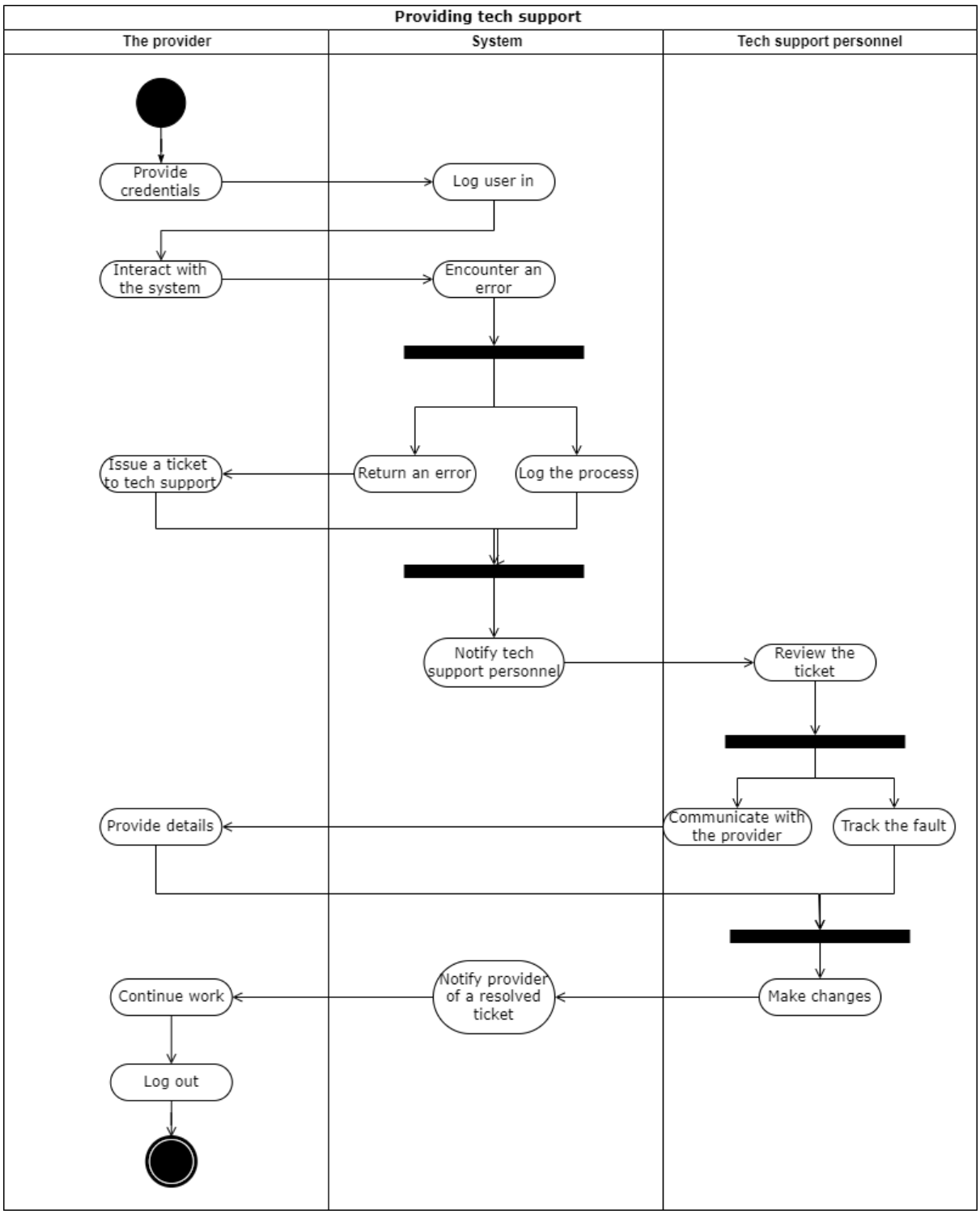

Figure 7: Food waste activity diagram

5 Physical view

In this part we analysed the topology of software components on the physical layer as well as physical connections between these components. This view contains:

1. Deployment diagram

5.1 Deployment diagram

The deployment diagram below shows the relationships between the software and hardware components in the system and the physical distribution of the processing. Food waste system could be accessed using an ordinary web browser, no additional installations should not be needed. When the user opens the page, it sends request to the linux FoodWasteSystem server. The System communicates with the PostgreSQL Database in the same linux server to retrieve the data.

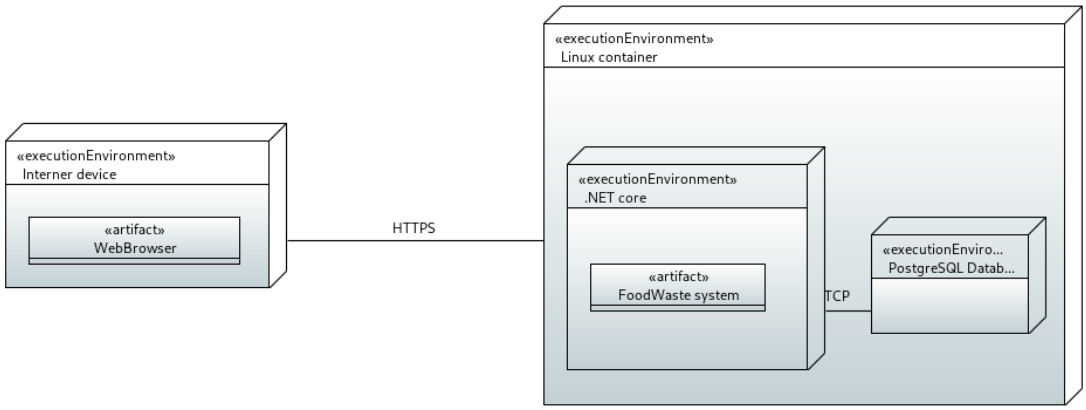

Figure 8: Food waste deployment diagram

6 CI/CD

Continuous integration, delivery and deployment (CI/CD) enables FoodWaste developer team to release and test software on a more frequent basis without compromising on quality. Each new version of software is ensured to be without compilation errors and passing all of the predefined tests.

1. Each developer tests their own features in local environment
2. Automatic pipelines ensure that changed system compiles without errors and passes all of the tests
3. Other developers review the changes
4. Changes are merged to test environment (automatically deployed on a virtual machine)
5. Once a batch of changes is collected manual testing is used to make sure no faults are present
6. Changes are delivered to production system by manually starting deployment process
7. Clients are served by the new system

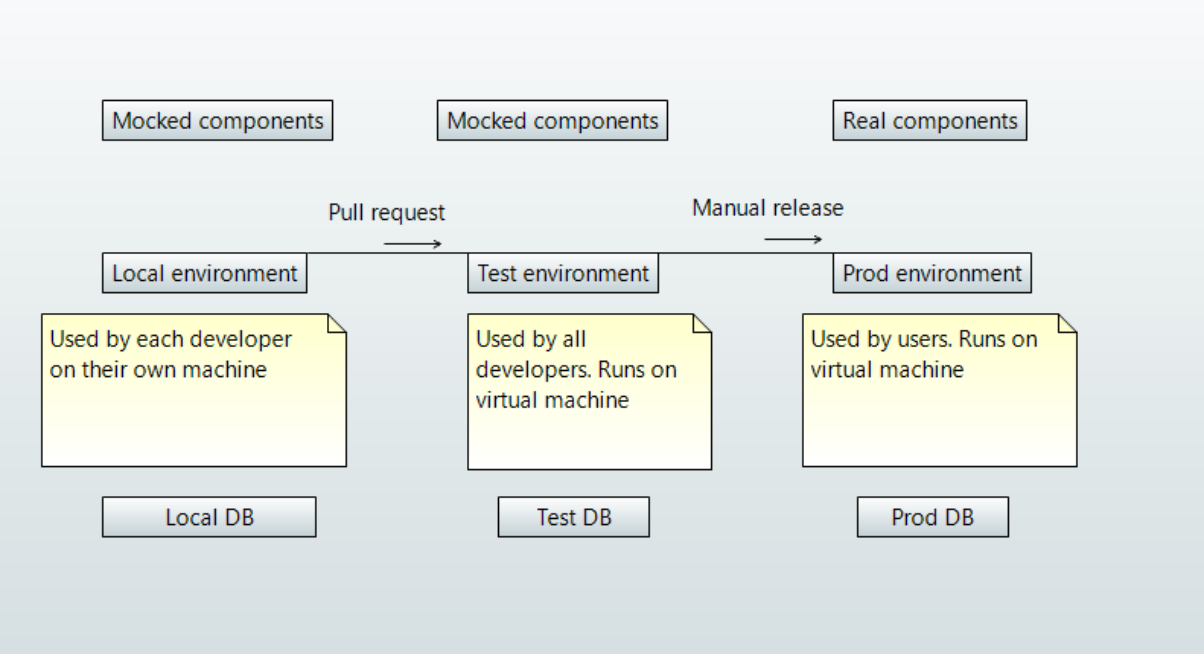

Figure 9: Food waste CI diagram

7 Use Case View

For our use cases we decided to provided diagrams of how the implemented changes should be used by system administrators, buyers and providers. The following diagrams show how providers can update their data, how buyers and providers can update the state of products, and how tech support should be handled.

7.1 Supplier data update use case – diagram and description

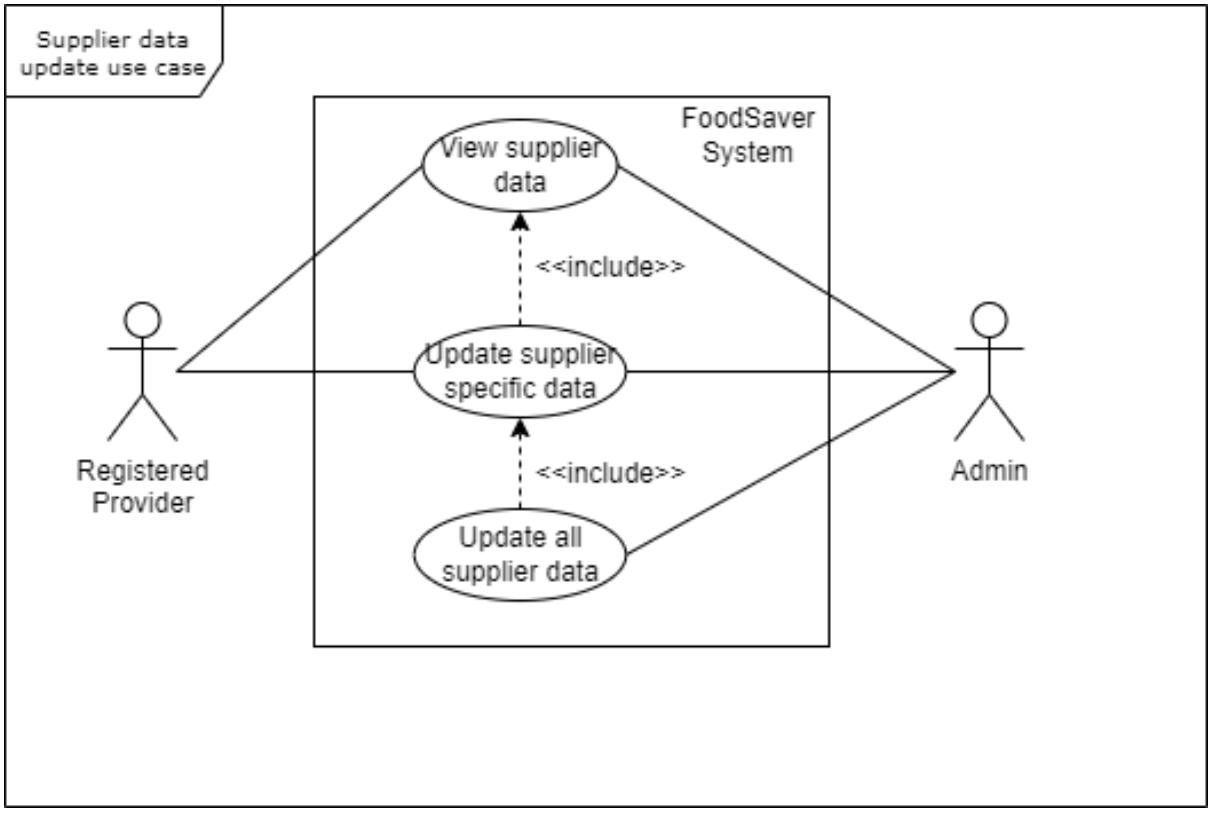

Figure 10: Supplier data update use case diagram

Here we can see a simple use case diagram of providers and how they can interact with their data. If they are logged in, they can view and change their data. A system administrator can also manage this date for the whole platform.

7.2 Product status change use case – diagram and description

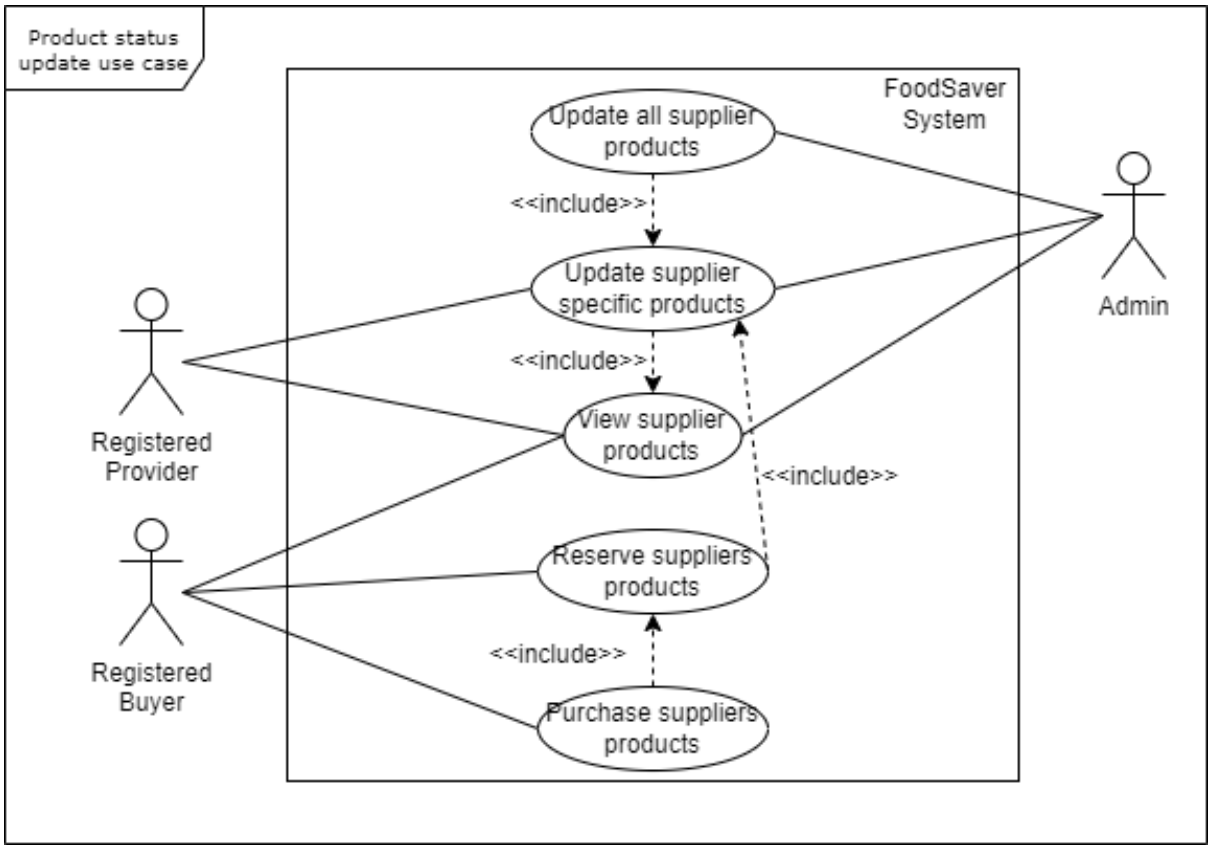

Figure 11: Product status change use case diagram

Here we can see how we expect our users to interact with our system in ways that would change the status of supplied products. To change a products status a buyer can purchase it or reserve it. Providers and administrators have more options - providers can see what they are selling on this platform and they allways can change the status of their products. The system administrator has the same possibility, but he can change the status of every single sold product on this platform.

7.3 Tech support use case – diagram and description

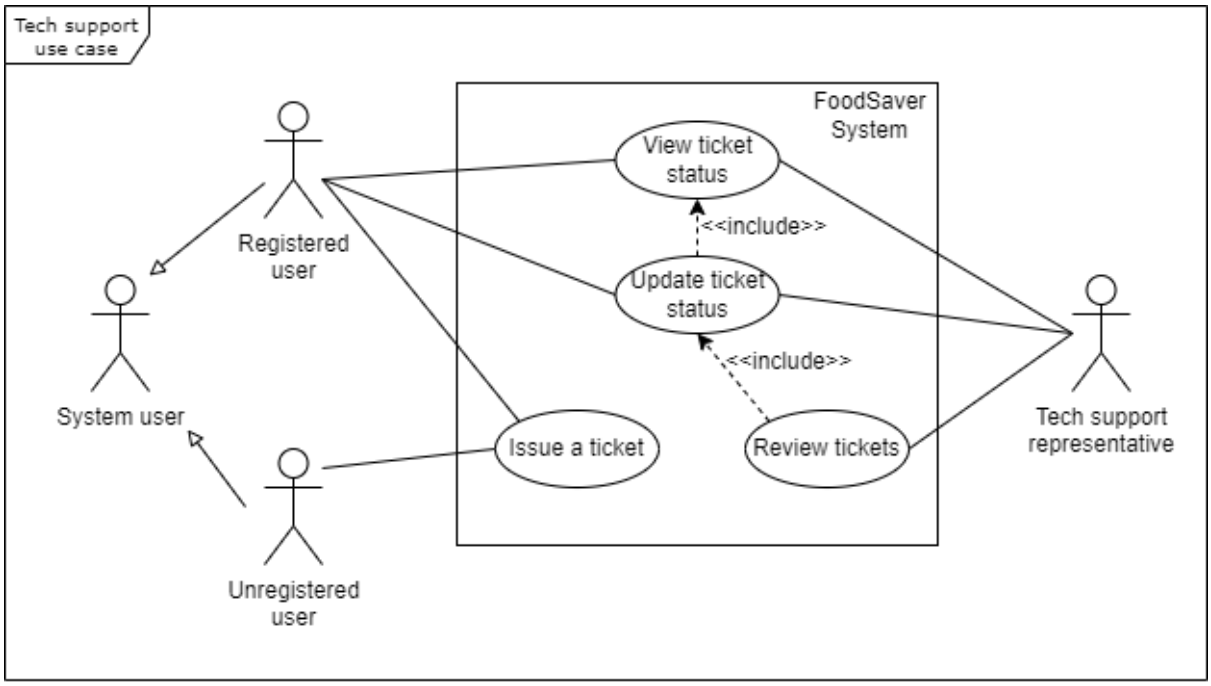

Figure 12: Tech support use case diagram

In this use case diagram we can see how we expect our users to interact with our system when faced with technical difficulties. As you can see, an unregistered user can issue a ticket if he encounters a problem. But only a register user can see if the ticket has recieved any atttention. The registered user can also provide updates to the ticket if he finds something that could help resolve the ticket. Tech support representative is constantly reviewing and updating tickets, hopefully with news about a successfull restoration of expected functionality.

8 Traceability

R1: Product status implementation, when it is bought etc.
R2: Restaurant accounts should be able edit their data.
R3: Add tech support.

	Development view	Logical View	Process View	Physical view	Use case view
R1					
R2					
R3					

Figure 13: Traceability matrix