A1 WasteLess

Student: Andrei Tosa

**Group: 30431**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

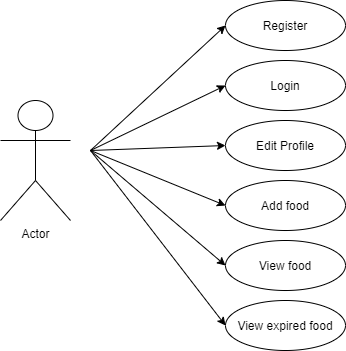
For this assignment we have to design and implement an application that helps users manage food waste. The user should be able to track all his food and see if he has an excess or if something went bad. He is presented with options to donate the food.

# Functional Requirements

Once a user is authenticated he can input grocery lists and see reports of how much food is wasted weekly and monthly. A grocery list item has a name and a quantity as well as a calorie value, purchase date, expiration date and consumption date. The system also allows users to track goals and minimize waste by sending reminders if waste levels are too high based on ideal burndown rates. The ideal burndown rate for 100 calories worth of groceries due to expire in 5 days is 20 calories worth of groceries per day. The system should provide you with options to donate excess food to various local food charities and soup kitchens and notify you of them prior to item expiration.

# Non-functional Requirements

* Implement and test the application
* Use an ORM and a DI Container
* Commit the work you do on your Git repository. Do it iteratively as you progress, not all at once (this will incur a penalty on your final mark)
* Use any OOP language you like. Non-exhaustive: Python, C#, Java, Ruby, C/C++, JS+Typescript
* Use a layered architecture
* Use the abstract factory pattern for creating weekly/monthly reports
* The data will be stored in a database
* All the inputs of the application will be validated against invalid data before submitting the data and saving it in the database.



2. Use-Case Model

*Use case: <use case goal>*

*Level: <one of: summary level, user-goal level, sub-function>*

*Primary actor: <a role name for the actor who initiates the use case>*

*Main success scenario: <the steps of the main success scenario from trigger to goal delivery>*

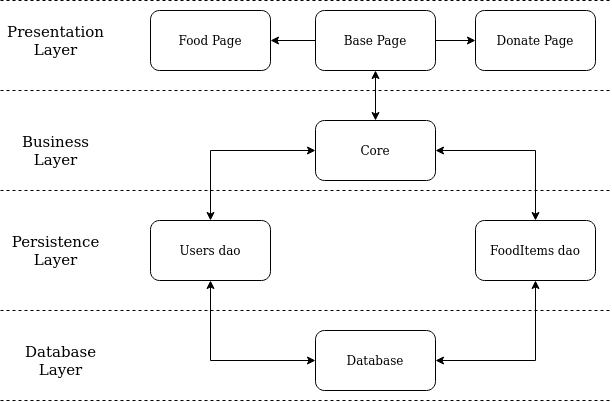
*Extensions: <alternate scenarios of success or failure>*

*]*

3. System Architectural Design

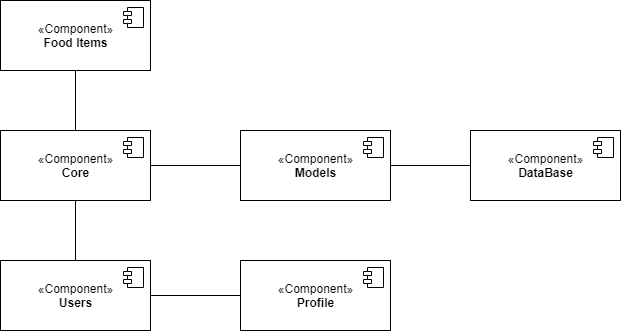
**3.1 Architectural Pattern Description**

For this project I was required to use the Layered Architectural Pattern. Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic). Although the layered architecture pattern does not specify the number and types of layers that must exist in the pattern, most layered architectures consist of four standard layers: presentation, business, persistence, and database.

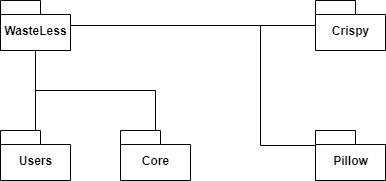
**

**3.2 Diagrams**

**Component Diagram**



**Package Diagram**



**Deployment Diagram**



4. UML Sequence Diagrams

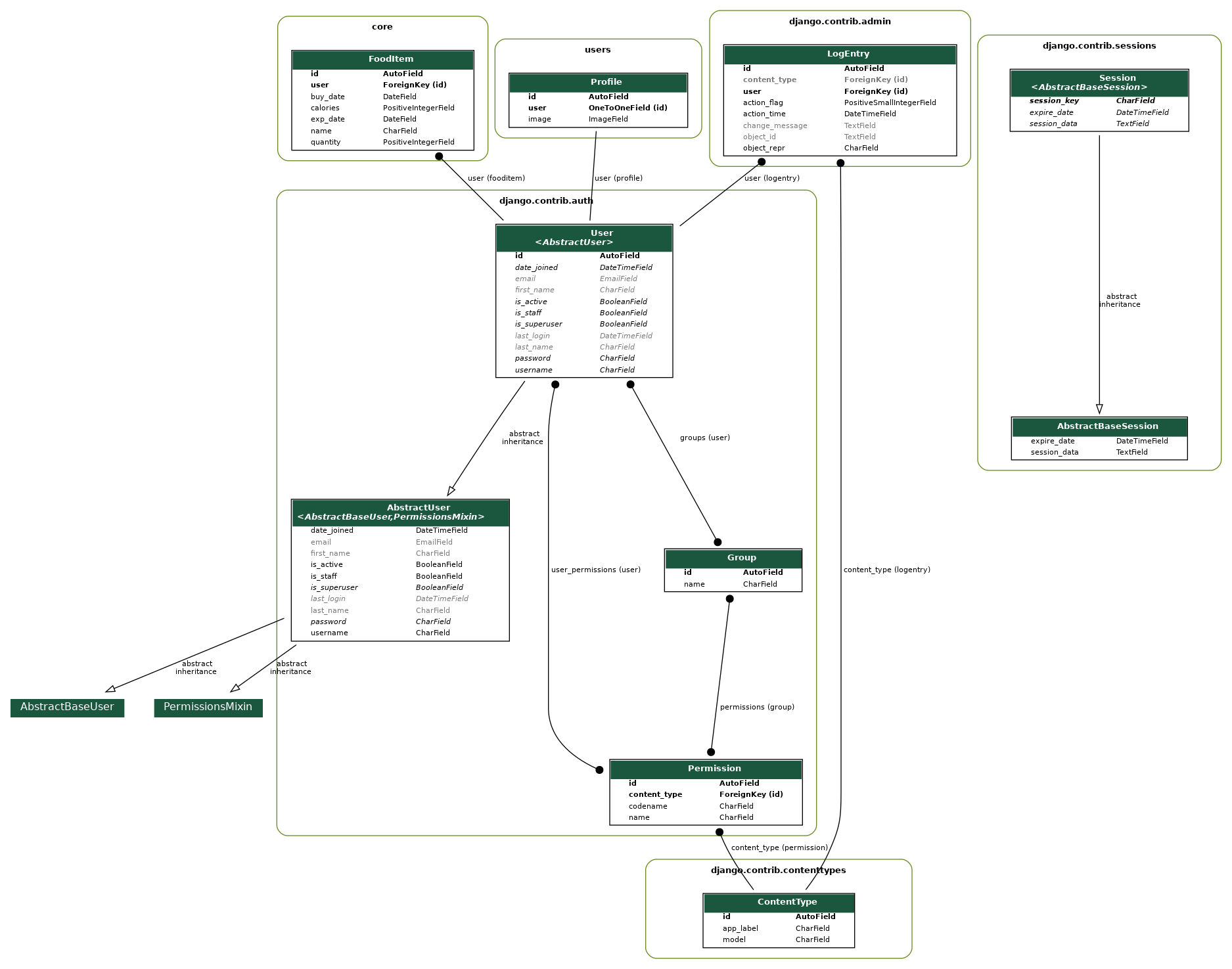
*[Create a sequence diagram for a relevant scenario.]*

5. Class Design

**5.1 Design Patterns Description**

The abstract factory pattern provides a way to encapsulate a group of individual factories that have a common theme without specifying their concrete classes. A factory is the location of a concrete class in the code at which objects are constructed. The intent in employing the pattern is to insulate the creation of objects from their usage and to create families of related objects without having to depend on their concrete classes.

**5.2 UML Class Diagram**



6. Data Model

**Profile:**

User: username, password, email

Image

**FoodItem**:

name = String

quantity = PositiveInteger

calories = PositiveInteger

buy\_date = Date

exp\_date = Date

user = ForeignKey

7. System Testing

Each component has been tested individually and after assuring functionality they have been put together. After that, data-flow testing was done to ensure that all components function as expected.

8. Bibliography

<https://www.youtube.com/user/schafer5>

<https://simpleit.rocks/python/django/generate-uml-class-diagrams-from-django-models/>