Wasteless Assignment 2
Analysis and Design Document

Student: Chiorean Tudor Octavian
Group: 30431

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Design and implement an application that helps users manage food waste.

1.2 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.

1.3 Non-functional Requirements

The system has to be easily usable (Accessibility) and safe (Security)

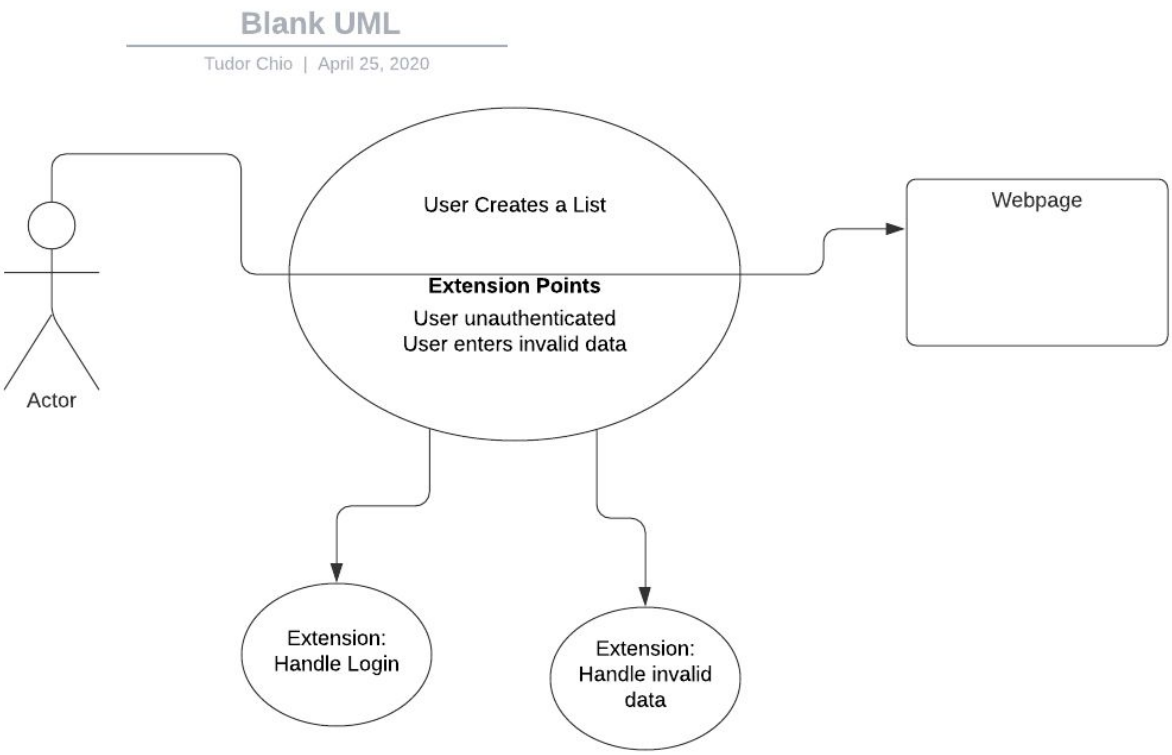
In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements.

2. Use-Case Model

Use case: User creates a List
Level: User-goal level
Primary actor: The user
Main success scenario:

1. User authenticates
2. User manages to create a list
3. User successfully adds items to the grocery list

Extensions:
User also gets the optimal burndown rate
User is unable to login
User is unable to manage to create a list



3. System Architectural Design

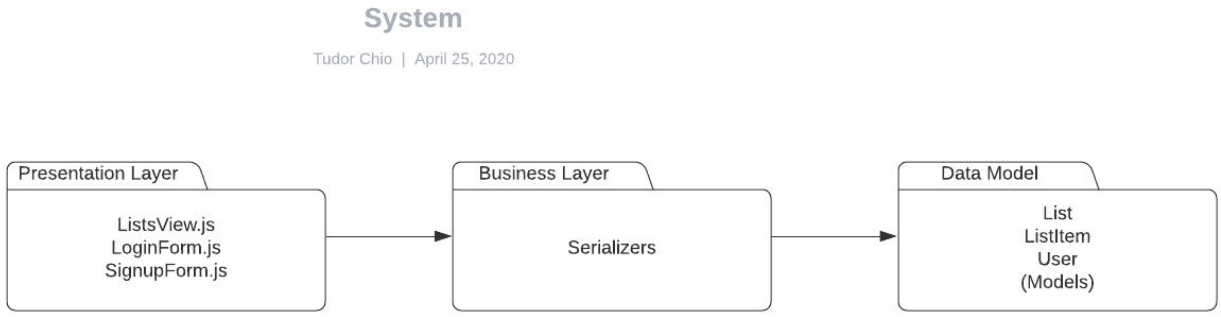
3.1 Architectural Pattern Description

A Layered Architecture, as I understand it, is the organization of the project structure into four main categories: presentation, application, domain, and infrastructure. Each of the layers contains objects related to the particular concern it represents.

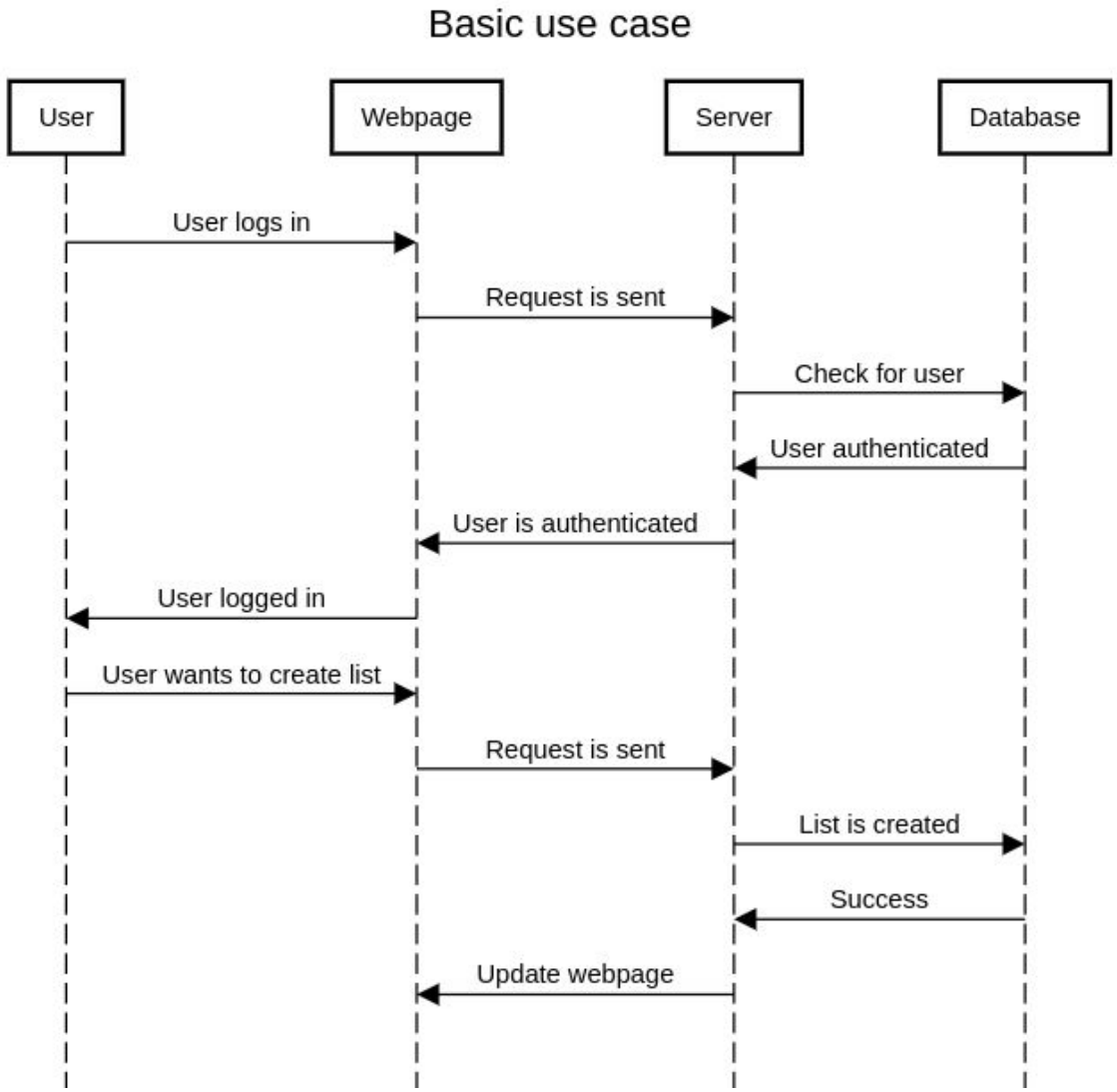
- The presentation layer contains all of the classes responsible for presenting the UI to the end-user or sending the response back to the client (in case we're operating deep in the back-end).
- The application layer contains all the logic that is required by the application to meet its functional requirements and, at the same time, is not a part of the domain rules. In most systems that I've worked with, the application layer consisted of services orchestrating the domain objects to fulfill a use case scenario.
- The domain layer represents the underlying domain, mostly consisting of domain entities and, in some cases, services. Business rules, like invariants and algorithms, should all stay in this layer.
- The infrastructure layer (also known as the persistence layer) contains all the classes responsible for doing the technical stuff, like persisting the data in the database, like DAOs, repositories, or whatever else you're using.

In this assignment we use a Client-Server architecture, consisting of a React frontend application and a Django backend, with Django Rest framework to create an API.

3.2 Diagrams



4. UML Sequence Diagrams



5. Class Design

5.1 Design Patterns Description

The MVT (Model View Template) is a software design pattern. It is a collection of three important components Model View and Template. The Model helps to handle database. It is a data access layer which handles the data.

The Template is a presentation layer which handles User Interface part completely. The View is used to execute the business logic and interact with a model to carry data and renders a template.

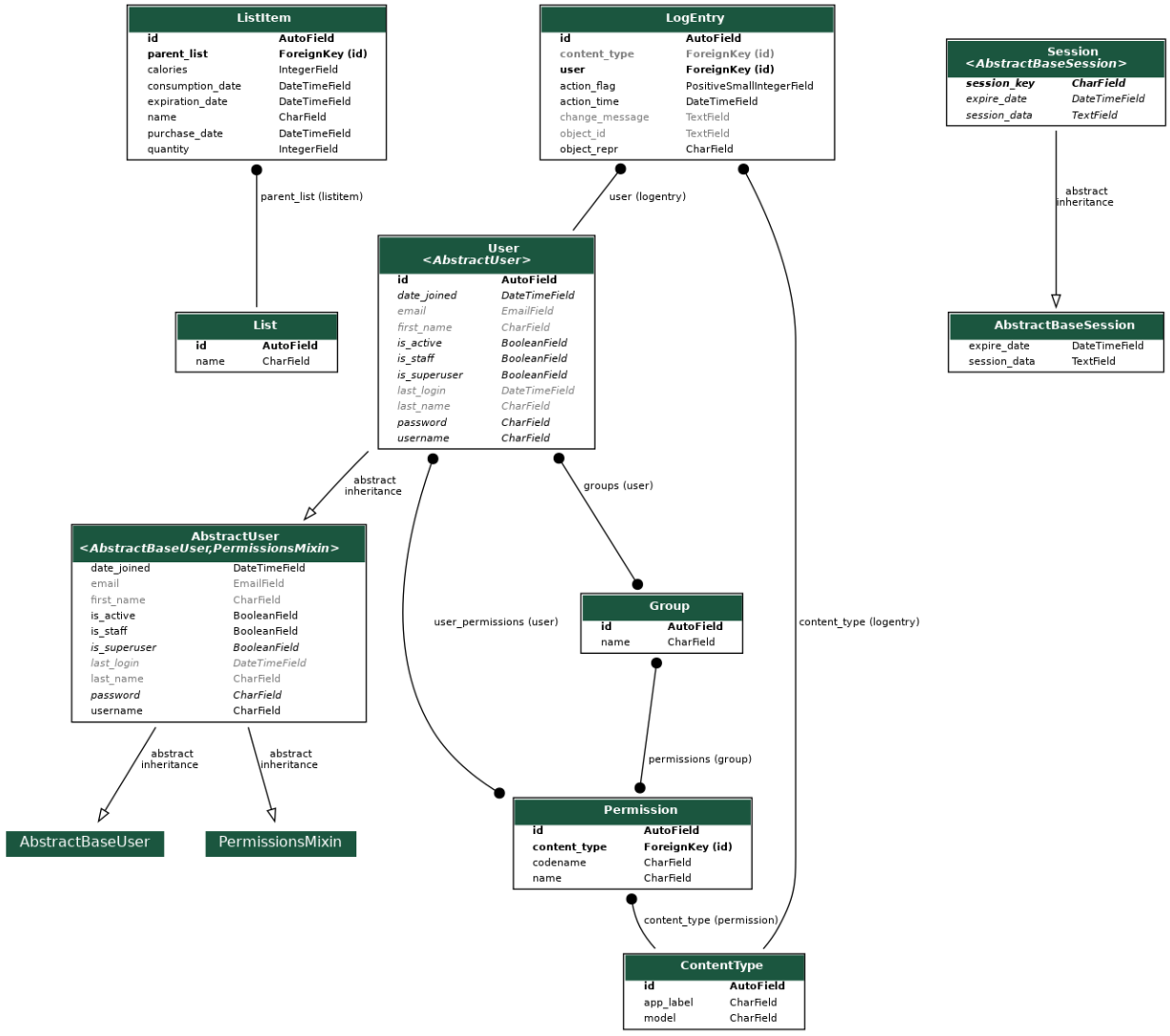
Although Django follows MVC pattern but maintains its own conventions. So, control is handled by the framework itself.

There is no separate controller and complete application is based on Model View and Template. Thats why it is called MVT application.

However, in this app we do not really use the template part of Django, opting for a rest API instead.

We used an observer in React to notify the user when an item is about to expire and provide them an opportunity to donate the respective items.

5.2 UML Class Diagram



6. Data Model

A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities. For instance, a data model may specify that the data element representing a car be composed of a number of other elements which, in turn, represent the color and size of the car and define its owner.

The data models used in this project are in the Django application.

7. System Testing

Coverage.py is a tool for measuring code coverage of Python programs. It monitors your program, noting which parts of the code have been executed, then analyzes the source to identify code that could have been executed but was not.

Coverage measurement is typically used to gauge the effectiveness of tests. It can show which parts of your code are being exercised by tests, and which are not.

8. Bibliography

https://coverage.readthedocs.io/en/coverage-5.1/
https://www.djangoproject.com/
https://www.django-rest-framework.org/
https://reactjs.org/
https://yarnpkg.com/
https://github.com/jquense/yup
https://react-bootstrap.github.io/
https://github.com/reflux/refluxjs