**Development and Features of FreshTrack app**

**Davide Galbiati (1230025034), Giuseppe Silvestro (1230024933)**

**Introduction**

**FreshTrack** is a web application designed to help users manage their pantry and shopping list efficiently. The application aims to provide a user-friendly interface for tracking pantry items and shopping list and receiving notifications for items nearing expiration. This document outlines the development choices and key functionalities of FreshTrack.

**Planning and Requirements**

The initial phase involved understanding the requirements for a pantry and shopping list management application. The key features identified were:

1. **Pantry management**: Adding, editing, and removing items.
2. **Shopping list**: Managing a custom shopping list.
3. **Expiration notifications**: Alerts for items nearing expiration.
4. **User-friendly and responsive interface**.

The planning phase has been followed by decisions on the technology stack to be used to develop the FreshTrack application.

**Technology Stack**

**Backend: Django**

Django was chosen for the backend due to its robustness, scalability, and the following features:

* **Rapid Development**: Django's built-in features and conventions allow for rapid development of secure and maintainable web applications.
* **Admin Interface**: Django comes with a useful admin interface that simplifies the management of application data.
* **ORM (Object-Relational Mapping)**: Django’s ORM allows for easy database interactions using Python code instead of raw SQL queries.
* **Security**: Django provides built-in protection against common security threats such as SQL injection, cross-site scripting, and cross-site request forgery.

**Frontend: HTML, CSS (Tailwind CSS), JavaScript**

* **HTML**: The standard markup language for creating web pages, HTML is essential for structuring the content of the application.
* **CSS (Tailwind CSS)**: Tailwind CSS was chosen for its utility-first approach, allowing for rapid styling of the application. It provides pre-defined classes that make it easy to create a responsive design.
* **JavaScript**: Used for adding interactivity to the web application.

**Database: SQLite**

SQLite was chosen as the database for its simplicity and ease of use during development. It is a lightweight, file-based database that is easy to set up and does not require a separate server.

**Package Management: Node.js**

Node.js and npm (Node Package Manager) were used to manage frontend dependencies, ensuring that all required libraries and frameworks are included and up to date.

**Hosting: PythonAnywhere**

PythonAnywhere was used for hosting the application, making it accessible from external locations. This platform allows for easy deployment and management of Django applications.

**Key Features**

**Pantry Management**

Users can add, edit, and remove items in their pantry. Each item includes details such as name, quantity, and expiration date. The application displays all items in a list format, allowing users to easily track their pantry stock. As additional features aimed at simplifying the add of new products, users can use a barcode scanner or upload the grocery receipt to automatically detect them.

**Shopping List**

The application enables users to manage a shopping list. Users can add pantry items to these lists and organize their shopping efficiently. The products labelled as purchased can be automatically added to the pantry.

**Product Details**

Products, both in the pantry and in shopping list, have their own dedicated pages where users can view more details and modify them, such as adding notes, categorizing, and moving them between different storage locations.

**Expiration Notifications**

FreshTrack includes a feature to alert users about items nearing their expiration dates. This helps prevent food waste and ensures that users can use their pantry items efficiently.

**Responsive Interface**

The frontend of the application is built using Tailwind CSS, providing a modern and responsive user interface. This ensures a seamless experience across different devices and screen sizes.

**Testing**

Comprehensive testing was conducted to ensure the functionality and reliability of the application. The tests included:

**Unit Tests**

Unit tests were written for various components of the application to ensure that individual units of code function as expected. This included testing the models, forms, and views.

* **Model Tests**: Verified that the models correctly handle data and relationships.
* **Form Tests**: Ensured that form validations and data handling are correct.
* **View Tests**: Checked that the views render correctly, and handle user interactions as intended.

**Integration Tests**

Integration tests were used to verify that different components of the application work together seamlessly. This included testing user flows such as logging in, adding items to the pantry and shopping list and modifying them.

**Immagine che contiene testo, schermata

Descrizione generata automaticamenteAutomated Tests**

Automated tests were implemented using Django’s test framework to ensure continuous integration and deployment. These are the results (we used the coverage.py tool)

**Manual Testing**

In addition to automated tests, manual testing was conducted to ensure the user interface is intuitive and functions correctly across different devices and browsers.

**Usage**

The application can be tested by following the instructions in the README.md file in the github repository at <https://github.com/Dado-hash/PrinciplesOfSoftwareDevelopment> or by visiting the website at <https://davideg.pythonanywhere.com/.>