The Fridge Manager

Jarrod Amyx

Joakim Eckerman

Dhruv Gorasiya

Zoe White

Bao Lam

Keshav Mehta

Ayush Patel

Preface

This fridge management system is intended for modern households looking to streamline their daily routines and provide fresh and nutritious food alternatives to their families and people with busy lives. The system is designed for anyone who wishes to easily and efficiently manage the contents of their refrigerator.

version: 1.0 of the fridge management system.

​​We think that everyone should have access to fresh and nutritious food alternatives. We designed this fridge management software solution to assist families in accomplishing that aim by easing the process of controlling fridge contents and reducing food waste. We realize how stressful contemporary life can be, therefore we decided to provide a solution that would make fridge management simple and effective for everyone. With this system, we hope to improve the lives of busy families everywhere.

Introduction

Introducing our innovative fridge management system, designed to make the daily task of managing the contents of a fridge effortless and efficient for every member of a household. With our system, users can easily manage one fridge and its several sections, including shelves, doors, produce sections, and an integrated freezer.

Users can create accounts with their names, pictures, contact email, and customize their preferences by setting the amount of space they are allowed to use in each fridge section. Our system includes a comprehensive database that has the physical dimensions and food types of at least 10 common food items, making it easy for users to add new items with expiration dates and handling recommendations.

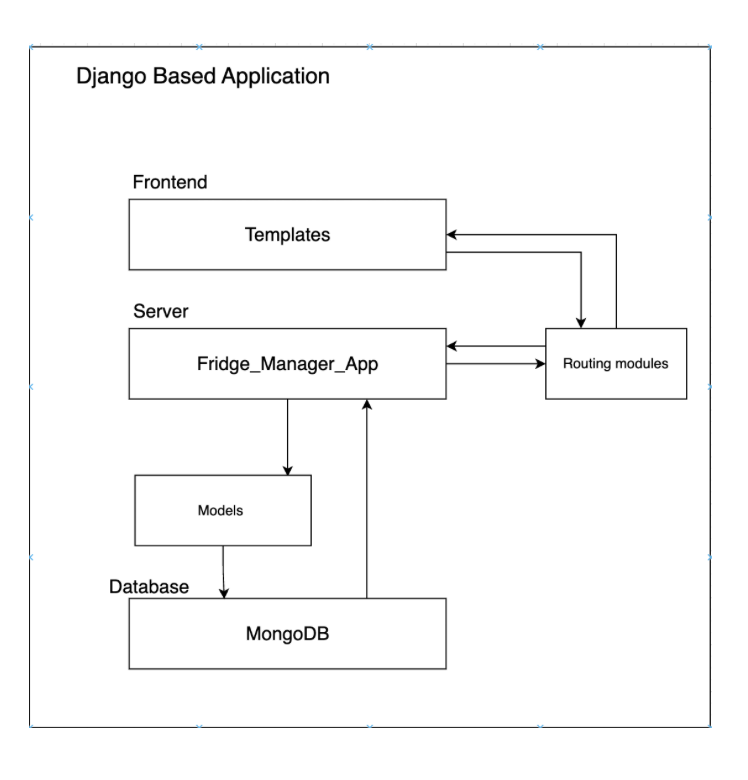
Our system also has several features to manage space, expiration dates, and other attributes. For instance, users can receive reminders when the expiration date of an item is near, reached, or exceeded. The system can also send warnings to users when the free space in the fridge is running low, prompting them to remove an item first when they try to deposit another one.

We believe that our fridge management system is an essential tool for modern households, streamlining daily routines, and ensuring that all members have access to fresh and healthy food options. With our system, managing fridge contents has never been easier, saving time and ensuring that every household member benefits from a well-stocked and organized fridge.

Architectural Change

a. What architectural changes did you make during implementation?

* We dropped the class based architecture and followed the django based frontend-backend and database integration. We followed a component based architecture where we implemented our architectural model based on the django application development architecture.
* To allow seamless communication between frontend, backend and database we implemented the following diagram.



* The templates in the frontend component represent HTML files that allow the users to interact with the application to enter and retrieve data about their refrigerator.
* Routing modules are the inbuilt django files where we mentioned navigation paths which allowed us to create a seamless follow of web pages on user’s demand with the help of backend processing.
* The server component contains actual logic of the fridge manager application which processes the data entered by the user to allow them to create an account, add refrigerator and remove items along with the ability to give access to other existing users. Thus this component does the job of processing data coming from the front end along with adding and retrieving data from the database.
* Modules component consists of frameworks for user accounts, refrigerators, shelves and items in the form of classes to allow systematic addition of documents to the database and logical retrieval of data from the database.
* The database component essentially is a storage base used in the application. It stores user accounts, refrigerators created by users, shelves of the refrigerators and items.

b. What’s the rationale behind the decision change?

* Since we chose the Django framework for creating the fridge manager application which follows component based application creation. Where there are essentially three components in the framework: fontend, backend and database. Thus we had to change our previous class based architecture to components based.
* The templates represent the frontend component, Fridge\_manager\_app represents the server component and MongoDB represents the backend Component.

Detailed Design Change

a. What detailed design changes did you make during implementation?

* We changed the frontend to be more simplistic than our original design process.
* We changed the format from an app based design to a website based design.
* We removed the ability to automatically sort items in the fridge.

b. What’s the rationale behind the decision change?

* We changed the frontend to be more simplistic to allow for us to focus more on our functionality. This will also be changed into the future to look more appealing.
* We changed from app to website to allow for the user to check on their fridge without the need to download an app.
* We removed this as our system would not know what is the priority for the users and what should go in the front or the back. We still provide the remaining volume to give the user an idea what they can fit in there.

Requirement Change

* We were going to use React for the Frontend but now we just have pure HTML for the front end. The reason is we want to focus more on the functionalities of the web app for now.
  + Does it change our design?
    - No longer using React for the frontend and instead using pure HTML, it is likely that the design will need to change. React is a JavaScript library that allows us to create dynamic user interfaces, while pure HTML is a markup language that is less flexible. Therefore, ourdesign may need to be simplified or adjusted to work within the limitations of pure HTML.
  + If so, what should be changed and how?
    - We should identify the specific areas that need to be updated. This may involve simplifying the design, removing elements that cannot be easily implemented in pure HTML, or finding alternative ways to achieve the same functionality.
  + This new requirement for using pure HTML has been implemented.
* We said that we would have password validation and requirements. Hence, we haven’t implemented it due to the time shortage and it is important to prioritize the core functionalities of our software first..
  + Does it change our design?
    - We have not yet implemented password validation and requirements but plan to do so in the future, our design does not need to change. Since it doesn’t affect how the software works or intervene with any functionalities. It's just a security aspect.
* We announced we would have: Pop up reminders for nearby or passed expirations / Editing account / Rearrange Items but haven't had them for this time release, they will be implemented in the future.
  + Does it change our design?
    - Does not affect the design, just missing functionalities.
    - Will be implemented in the future now that we have all our core functions working.