USE CASES
GITHUB PROJECT LINK:
https://github.com/evanhowe03/SoftwareDesignProject

Use Case R1: Store user recipe into database

Primary Actor: User
Stakeholders and interests:
- User: wants to be able to store many recipes

Preconditions: User is created
Success guarantee: Recipe is added into database.
Main success scenario:
1. User has created a recipe
2. User clicks “Save to database”
3. Recipe is saved

Extensions:
2a. Device is not connected to the network
    1. Device signals no network connection and prompts to save to local network
    2. Recipe is saved to local storage until device reconnects to network
        2a. There is not enough storage in local storage
            1. Recipe is rejected

Special requirements:
1. The interface of saving recipes must follow accessibility guidelines and be color blind friendly
2. System should respond within 2 seconds and be compatible with different environments.

Technology and data variation list:
3a. The recipes being saved will display an image of the user’s choosing, typically the final product when placed into the library.
3b. The recipe will be able to be covered into a slideshow or pdf

Use Case R2: Create Custom Recipes

Primary Actor: User
Stakeholders and interests:
- User: wants to be able to fully customize recipes for their needs

Preconditions: User is logged in.
Success guarantee: Recipe is created
Main success scenario:
1. User clicks “create a new recipe”
2. Device prompts a variety of templates and recipes found online
3. User chooses a blank template
4. User creates a custom recipe

Extensions:
2a. Device is not connected to the network
    1. Device only prompts a blank template
    2. User chooses the blank template

3. User creates a custom recipe

3a. User chooses a shared template

1. User changes the recipe to fit their own needs

Special requirements:

1. The interface is disabled friendly where it adjust features for that specific users need
2. Each transition should happen within seconds and respond back in small amounts of time

Technology and data variation list:

2a. Online recipes will be accessible using any API of the user's choosing

Use Case R3: Identity missing ingredients from recipes

Primary Actor: User

Stakeholders and interests:

- User: wants accurate feedbacks for shopping list

Preconditions: User has recipes saved

Success guarantee: A temporary shopping list is created

Main success scenario:

1. User opens up their library of saved recipes
2. User chooses a recipe they are interested in
3. Recipe shows that a few ingredients are missing
4. Device prompts to create a list of ingredients
5. User chooses another recipe they are interested
6. Recipe shows that a few ingredients are missing
7. Device prompts to add ingredients to existing list
8. User chooses to add to existing shopping list

Extensions:

3a. Recipe shows that fridge has all necessary ingredients

1. User chooses another recipe they are interested in
2. Recipe shows that a few ingredients are missing
3. Device prompts to create a list of ingredients
4. User chooses to not create a list

Special requirements:

1. The colors indicators showing missing ingredients must be color blind friendly
2. The list creation process must complete within 3 seconds 90% of the time within normal conditions

Technology and data variation list:

4a. List should be stored as a csv file

Use Case R4: Sort recipes based on ingredients

Primary Actor: User

Stakeholders and interests:

- User: Wants to easily find recipes based off of ingredient(s)

Preconditions: User has recipes saved

Success guarantee: Recipes are sorted based on ingredient(s) selected by user

Main success scenario:
1. User opens up their library of saved recipes
2. User selects option to sort recipes by ingredient(s)
3. Device prompts user to input ingredient(s) they would like to sort by
4. User inputs the ingredients to sort by
5. Device filters recipes to show recipes that can be made with ingredient(s)
6. User selects a recipe from the filtered list

Extensions:
2a. Online recipe API is unable to be connected to
   1. Notify user that online API is unable to be connected to
   2. Provide results based off of locally available recipes

5a. No recipes contain the selected ingredients
   1. Device suggests partial matches that contain similar/some of the ingredient(s) filtered for
   2. User selects a suggested recipe or redoes search with different filters

Special requirements:
- Sorting algorithm should be able to suggest partial matches and be able to make accurate alternative substitutions for ingredients
- Sorting algorithm should be responsive within 5 seconds of adding a filter.

Technology and data variation list:
3a. Sorting algorithm should be able to handle data from both a local database alongside data from the online API.

Use Case R5: Manually Add Item(s) to Inventory

Primary Actor: User

Stakeholders and interests:
- User: Wants to accurately track their inventory of ingredients, in order for accurate representation in database

Preconditions:
- User is created
- User can access a way to update database (via smartphone, smart fridge, other devices.)

Success guarantee: Item is added to database for system to use

Main success scenario:
1. User selects ‘add item to inventory’ on their preferred device
2. Device prompts user to scan item via barcode or manual entry
3. User selects manual entry
4. User inputs item details, such as quantity, name, expiration date (if needed)
5. User confirms entry
6. Item is added to the inventory database
7. Device displays a confirmation, prompts user if they would like to add more items

Extensions
6a. User realizes details are wrong during confirmation
   1. User inputs details incorrectly and is presented wrong details on confirmation
   2. User selects ‘edit’ and starts back at step 2.

Special requirements:
- Item should be confirmed to be added to the database within 3 seconds of selecting ‘confirm’

Technology and data variation list

6a. System should sync with the cloud database after each confirmation so that there are two points of storing data (locally and cloud based).

Use Case R6: Alert Items Getting Low

Primary Actor: User
Stakeholders and interests:
- User: Wants to be kept up to date with items inside the inventory

Preconditions: Having items inside the inventory
Success guarantee: Device outputs a warning that certain ingredients are running out.
Main success scenario:
1. User logs into the application
2. The inventory notifies the user that items A and B are running low
3. Device prompts to add both items to the shopping list
4. User accepts and a shopping list is created with items A and B

Extensions:
2a. The inventory shows that all ingredients are sufficient for the recipes stored
4a. User chooses to add item A to the shopping list and ignores item B
    1. User accepts and a shopping list with only item A is created

Special requirements:
1. System should notify user low inventory 5 seconds of logging in
2. Alert should be adapted the user so it may have different color, font size, or look depending on user

Technology and data variation list:

Use Case R7: Remove Item(s) from Inventory

Primary Actor: User
Stakeholders and interests:
- User: Wants to remove item(s) from the database

Preconditions: User has added items to the database
Success guarantee: Item(s) is/are removed from the database
Main success scenario:
1. User selects ‘delete item from inventory’
2. System prompts user to select from list of items in the inventory to remove
3. User selects item(s) and finishes selection
4. System prompts user to confirm item(s) to delete
5. User selects confirm
6. Inventory is refreshed with deleted items removed

Extensions
2a. No items in inventory
   1. System prompts user that there is no items in the inventory
   2. User acknowledges prompt and returns to main menu

4a. User realizes wrong selection
   1. User inputs details incorrectly and is presented wrong details on confirmation
   2. User selects ‘edit’ and starts back at step 2.

6a. Inventory is completely cleared
   1. User selects and confirms to delete all items from inventory

2. Inventory shows up blank with only option to add to inventory

Special requirements:

- Inventory should process remove order within 3 seconds of confirmation
- User interface should be intuitive and friendly

Technology and data variation list

6a. Inventory should sync with cloud database when updated

Use Case R8: Scan Grocery Items into Inventory

Primary Actor: User

Stakeholders and interests:

-

Preconditions:User has item(s) they want to add to inventory via barcode
User has device to scan barcode with

Success guarantee:

Main success scenario:

1. User selects ‘add item to inventory’ on their preferred device
2. Device prompts user to scan item via barcode or manual entry
3. User selects barcode entry
4. User scans barcode
5. System displays item scanned, prompts user to input quantity, expiration date (if needed)
5. User confirms entry
6. Item is added to the inventory database
7. Device displays a confirmation, prompts user if they would like to add more items

Extensions

5a. Barcode Scanner Success

1. Device scans barcode
2. Item details (such as name, category) are automatically filled in
3. User adjusts details as needed

5b. Barcode Scanner Failure

1. Device scans barcode
2. Cannot understand/read in barcode
3. Prompts user to entry details in manually

Special requirements:

- Item should be confirmed to be added to the database within 3 seconds of selecting ‘confirm’

Technology and data variation list

5a. System should allow for items to be scanned in via a barcode, which should be fetched from an API that looks up items from most grocery stores.

Use Case R9: Track Quantity of Condiments and Partitioned Goods After Use

Primary Actor: User

Stakeholders and interests:

- User: Wants to track the remaining quantity of condiments and partitioned goods (i.e. portions of packaged foods, spices, jars) to maintain accuracy

Preconditions:User has added partitioned goods to the inventory

Success guarantee: Partitioned foods are updated with an accurate representation of their state

Main success scenario:

1. User selects ‘Update item’ on a partitioned item in the inventory list
2. System prompts user to update the quantity of the item (i.e. 50% full, half used)
3. User inputs quantity amount
4. User selects confirm
5. Item is updated in the inventory

Extensions

1a. Item is not a partitioned item
   1. Item user selects is not a partitioned item either due to inputting error or user error
   2. System allows user to edit item in the inventory
   3. User changes the quantity type to a partitioned item

3a. User sets quantity amount to ≤0%
   1. User sets item quantity to ≤0%
   2. System prompts user that the item will be deleted
   3. User either confirms or edits quantity to a >0% number

Special requirements:
- Item should be confirmed to be updated in the database within 3 seconds of selecting ‘confirm’

Technology and data variation list

5a. System should sync with the cloud database after each confirmation so that there are two points of storing data (locally and cloud based).

Use Case R10: Identify Untracked Item Usage

Primary Actor: User

Stakeholders and interests:
- User: Wants to track items that are not inputted into the inventory system

Preconditions:User has ingredients in inventory
User has been using items that are not tracked in the system

Success guarantee: User adds untracked items to the inventory, thus making them tracked

Main success scenario:

1. System notices that items that are not tracked are being removed (via sensors, behavior patterns, etc)
2. System prompts user with a message asking if there is untracked items that need to be added
3. User realizes item is untracked, selects prompt to add item to inventory
4. User goes through standard addition procedure (either thru R5/R8)
5. System updates inventory with now tracked item

Extensions

2a. User is not removing items
   1. User is prompted that the system thinks items that are untracked are being used
   2. User is not using untracked items, selects option to ignore suggestion

Special requirements:
- System prompts users in an accessible way, so that users with ADHD and other conditions are considered (which is the point of this requirement.)

Technology and data variation list

1a. Fridge should either be able to have sensors to detect items leaving, either through integrated components or through external devices provided with the system.
4a. System should be able to detect barcodes in compliance with requirement R8.

Use Case R11: Modify Inventory Based on Completed Recipes

Primary Actor: User
Stakeholders and interests:
- User: Wants inventory to update without having to remove each item after completing a recipe

Preconditions: User has items available for a recipe in the inventory
User starts a recipe
Success guarantee: Inventory is updated to remove a quantity amount of each item in the recipe from the inventory
Main success scenario:
1. User selects a recipe to make and receives relevant information
2. Items used to make recipe are placed on a 'hold' status to indicate they are being used in the inventory
3. After predetermined amount of time passes, system prompts user to update the inventory based off of completion of recipe
4. User confirms completion of recipe
5. Inventory removes relevant amount of each item used in the recipe

Extensions
1a. User does not have enough in inventory to make selected recipe
    1. System notifies user that the recipe cannot be made, and redirects them back to recipe page
2a. User tries to start new recipe with items that are placed on 'hold'
    1. System notifies user that items selected are currently being used in progress by a recipe
    2. Redirects user back to page where they were using the item
3a. User did not complete recipe
    1. System prompts user to update inventory based off of completion of recipe
    2. User selects 'not completed'
    3. System prompts user to update inventory
        3a. User did not use any items in the recipe, and selects cancel
        3b. User used some items, and updates the inventory as applicable

Special requirements:
1.After user notify that user is done with recipe within 2 seconds the inventory should be updated
2.System should handle multiple concurrent recipe sessions
Technology and data variation list
3a. System should have predetermined amounts of time for each recipe, which would be fetched by the external recipe API and stored locally.

Use Case R12: Modify Inventory Based Off of Grocery Receipt

Primary Actor: User
Stakeholders and interests:
- User: Wants to update inventory by scanning in grocery receipt therefore bypassing having to add each item individually

Preconditions: User has a supported* grocery receipt to add to inventory
Success guarantee: All items from grocery receipt are added to systems inventory
Main success scenario:
1. User selects 'add item' on a device capable of scanning.
2. User scans the grocery receipt with device.
3. System reads in QR code, interfaces with API to add items to inventory
4. User is presented with list of items to be added from interpreted grocery receipt.
5. User confirms list
6. Items added to system inventory

Extensions
3a. No QR Code / Unsupported receipt type
1. System cannot read in QR code / receipt
2. System relies on a OCR to read in the names of items on the receipt along with quantity
3. User manually confirms each item as they are processed

3b. Receipt already read in
1. System detects receipt is was already inputted
2. System notifies that the user may be inputting duplicate receipt
3. User can either confirm or ignore warning

Special requirements:
- System handles duplicate receipts in a way that coordinates with accessibility requirements, since forgetfulness is a part of ADHD and other conditions.

Technology and data variation list
3a. Ability to read in receipts using QR codes, which would require coordination with grocery stores and/or an API to read them in
3b. Implementation of an OCR so that the receipt can still be read in, granted with less accuracy.

Use Case R13: Track and Alert User on Upcoming Expiration Dates

Primary Actor: User
Stakeholders and interests:
- User: Wants to know when perishables/foods will expire without having to keep track of each individual one.

Preconditions: User has items with expiration dates in the inventory
Success guarantee: User is notified and aware of items that will be expiring within a predetermined deadline
Main success scenario:
1. User is notified on a device (i.e. smart fridge display, smartphone push notification) that item(s) are going to expire within a preset deadline (i.e. one week, 3 days).
2. User acknowledges notification
3. System prompts user if item(s) that are going to expire should be added to grocery list
4. User confirms addition to grocery list
5. System adds item(s) that are going to expire to list

Extensions

3a. User does not want to add items to grocery list
1. User rejects adding item that is going to expire to grocery list
2. System labels the item as ‘ignored’ in inventory list

Special requirements:

- System notifies the user as an accessibility feature, as forgetfulness is a part of ADHD and other conditions.

Technology and data variation list

Use Case R14: Allow Shopping List Substitutions

Primary Actor: User

Stakeholders and interests:

- User: Wants flexible shopping list that does not limit user to one brand/product

Preconditions: User has a shopping list with specific brands/products to purchase
User has specified a grocery store location to shop at.

Success guarantee: User is presented with alternative selections that are a satisfactory substitute in the case that a product is unavailable.

Main success scenario:

1. User views shopping list on device before trip
2. System notifies that item(s) on list are not available at predetermined grocery location.
3. System presents reasonable substitutions to unavailable products
4. System asks user if these substitutions should be committed on the shopping list.
5. User accepts substitutions.
6. Shopping list is updated to remove old product and replace with new for this trip only.

Extensions

4a. User denies substitutions
1. User denies substitution option
2. System ignores the item on the shopping list for that trip

Special requirements:

Technology and data variation list

2a. System should be able to fetch available products at a specific grocery store through the use of an API or similar method
3a. System should be able to make reasonable choices for substitutions.

Use Case R15: Add items to shopping list

Primary Actor: User

Stakeholders and interests:

- User: Wants to add an item to the shopping list

Preconditions: User has a shopping list

Success guarantee: Item is added to shopping list for user to track

Main success scenario:

1. User selects shopping list to modify
2. User selects ‘add item’
3. System asks user to input item information, such as name, quantity
4. User inputs name and quantity,
5. System fetches products with user inputted name and displays them for user to select
6. User selects desired item

7. Item is added to shopping list

Extensions

5a. Item is not found

1. System cannot fetch item due to failure to search, or online database is inaccessible
2. System notifies user to either reinput name, or add generic line item to shopping list
   2a. User reinputs name to repeat process, hopefully successfully
   2b. User selects to add it as a generic line item, which acts as just a checkbox on the list

Special requirements:

1.The added item should be in the shopping list within 2 seconds

2.The search feature for these items shall be easy to use and adapt to the user

Technology and data variation list

5a. System should be able to fetch available items that match the item name. Would require access to an API or other means of communicating with store.

Use Case R16: Remove items from shopping list

Primary Actor: User

Stakeholders and interests:

- User: Wants to remove a item from the shopping list

Preconditions: User has a shopping list with items in it

Success guarantee: Item is removed from shopping list

Main success scenario:

1. User selects an item(s) to be removed from shopping list
2. System asks for confirmation with list of items to be removed
3. User confirms selection
4. Items are removed from shopping list

Extensions

2a. User cancels confirmation

1. User selects ‘cancel’ on remove operation
2. Items are not removed
3. User is returned to shopping list screen

Special requirements:

Technology and data variation list

Use Case R17: Display Shopping List Amount

Primary Actor: User

Stakeholders and interests:

- User: Wants to see amount of items on shopping list to make decision on when to visit store

Preconditions: User has a shopping list with items on it

Success guarantee: User is presented with information to make an informed decision on when to go store

Main success scenario:

1. User selects a shopping list
2. System presents user with amount of items in shopping list, alongside a list of the items in the list

Extensions

2a. Shopping list is empty

1. System notifies user that the shopping list is empty

2. User acknowledges notification, goes back to main screen

Special requirements:

Technology and data variation list

Use Case R18: Sort Shopping List by Department

Primary Actor: User

Stakeholders and interests:

- User: Wants to see where shopping list items are by department in order from streamlined shopping

Preconditions: User has shopping list with items in it

Success guarantee: User is presented with department of each item in the shopping list

Main success scenario:

1. User selects shopping list to sort by department on.
2. User selects 'sort by department' on shopping list
3. System presents list sorted by department

Extensions

3a. Department is not listed for item

1. Items in shopping list are unassigned to a department (generic, or by some other means)
2. Items are prioritized to the top of the list to remind user about unsorted items.

Special requirements:

- Requirement is part of the software being accessible, as presenting the user with this information helps remind them of items.

Technology and data variation list

3a. System should be able to fetch what department items are in, either via a API or some other means of communication.

Use Case R19: Track Shopping Metrics

Primary Actor: User

Stakeholders and interests:

- User: Wants to see shopping metrics to coordinate with budget

Preconditions: User has a shopping list with items
User has scanned in receipts with transactions

Success guarantee: User is presented with shopping metrics

Main success scenario:

1. User selects 'See Shopping Metrics'
2. System calculates shopping metrics based off of weekly spending from previous receipts, and shopping lists
3. User is presented with information in numeric and graphical ways

Extensions

2a. Shopping list trips and scanned receipts do not line up

1. User scanned more/less receipts than shopping trips recorded
2. System only displays numeric value for weekly amount and overall amount (whereas the normal case would display per trip amounts)

Special requirements:

1.The system should do calculations within seconds so the user has a good and fast response experience

Technology and data variation list

Use Case R20: Recommend Frequently Bought Items

Primary Actor: User
Stakeholders and interests:
- User: Receive suggestions on what they frequently buy so they can add to shopping list

Preconditions: User has a shopping list
User has scanned in several receipts
Success guarantee: User is aware of items that they frequently buy
Main success scenario:
1. User views shopping list
2. System suggests to add frequently bought items to shopping list
3. User adds frequently bought items to shopping list
4. System updates shopping list with item(s)

Extensions
3a. User ignores suggestions
1. User ignores suggestions when viewing shopping list (suggestion is not a notification - it is part of the list)
2. Suggestion remains on list

3b. User dismisses suggestions
1. User selects 'ignore' on suggestions
2. Suggestion is removed from list

Special requirements:
1.The reminder should be a safe suggestion and stand out to the user and not force the user to do anything
Technology and data variation list
2a. System should be able to calculate and interpret shopping lists for frequently bought items.

Use Case R21: Setup Meal Plan

Primary Actor: User
Stakeholders and interests:
- User: Wants to queue recipes in a plan format to have structure on what to cook over a time period

Preconditions: User has recipes they would like to make
Success guarantee: A 'meal plan' is created with queued recipes over a specified time period
Main success scenario:
1. User selects multiple recipes they would like to make
2. User selects 'create meal plan'
3. System prompts user to input over how many days they would like to spread the recipes out
4. User inputs time period
5. System prompts user to confirm potential schedule, alongside recipes to provide
6. User selects confirm
7. Meal plan is created, user is notified during time period when recipe should be made

Extensions
3a. User puts invalid time amount
1. User inputs a invalid time, such as ≤0.

2. System throws error stating invalid time amount, prompts user to reinput time

5a. User rejects schedule

1. User rejects schedule period
2. System prompts user to reinput time period to use recipes

7a. User ignores meal plan notification during allotted time

1. User ignores notification to make recipe
2. Upon opening the software again, user is notified that a meal plan was missed
   2a. User chooses to repeat recipe, extending meal plan time by period amount
   3a. User chooses to ignore recipe, meal plan time remains the same

Special requirements:

1.Setting up should be adaptable base on user and responses should be within seconds

2.Plans should be made logically with the help with the system

Technology and data variation list

5a. System should be able to space recipes accordingly, alongside understanding periods of time where a user won't cook a meal (i.e. work, sleep)

7a. System should create thoughtful notifications during time period to remind user of created meal plan.

Use Case R22: Append to shopping list based on meal queue

Primary Actor: User

Stakeholders and interests:

- User: Wants to add items needed for meal plan to shopping list if meal plan is planned for the future

Preconditions: User has a shopping list
User has a meal plan set in the future/before next shopping trip

Success guarantee: User's shopping list is automatically updated with items needed for meal plan

Main success scenario:

1. User creates a meal plan (As described in R21), and sets time period to start in future
2. Shopping list is updated with items needed to make recipes in meal plan

Extensions

Special requirements:

- System should provide notification to ADHD users in case they have forgotten

Technology and data variation list

2a. System should communicate between meal plan schedule and shopping list.

Use Case R23: Sort Recipes Based On Criteria Tags

Primary Actor: User

Stakeholders and interests:

- User: Wants to see recipes based off of specific tags

Preconditions:

Success guarantee: User is presented with recipes that correspond with requested tags

Main success scenario:

1. User opens up recipe browser
2. User filters by specific tag(s)
3. System automatically fetches recipes that match specified tag(s)
4. Recipe list is updated to show filtered recipes

Extensions
3a. No recipes match tag
1. User specifies tag(s)
2. System notifies user that specified tag(s) resulted in 0 matches
3. System prompts user to change filtered tags

Special requirements:
1.The filters should be adjusted to the user ability and preferences and must have visual indicators

Technology and data variation list
3a. System should feature robust system of tags, so that the user can easily search through recipes

Use Case R24: Allow Recipe Substitution on Meal Plans

Primary Actor: User

Stakeholders and interests:
- User: Wants Meal Plan to always have meals even if requisite ingredients run out before recipe takes place

Preconditions: User has items in inventory
User has set up a Meal Plan
User uses requisite ingredients before recipe in meal plan occurs
User has updated inventory with used items

Success guarantee: User's meal plan is consistently has recipe prepared similar to user specification

Main success scenario:
1. User realizes they used requisite items for upcoming meal plan recipe
2. System prompts user upon opening the software next that a meal plan recipe has been substituted accordingly
3. System prompts user to confirm or deny substitution
4. User accept substitution
5. System updates meal plan with substituted meal

Extensions
2a. System cannot find viable substitution
1. System cannot create a reasonable substitution for meal plan recipe
2. System notifies user to select a new recipe for meal plan
2a. User ignores request for new meal plan recipe, skips that meal plan recipe entirely
2b. User selects a new meal plan recipe, meal plan is updated accordingly

3a. User denies substitution
1. User denies the substitution.
2. System prompts user to skip specific meal plan recipe, or choose a new recipe
2a. User selects a new meal plan recipe, meal plan is updated
2b. User ignores request, meal plan recipe is skipped

Special requirements:
1.Subsitution must be compatible with users preferences and accessibility
2.sunisitutions must be logical for user-defined preferences

Technology and data variation list
2a. System should be able to make reasonable substitutions for meal plan recipes based off of current inventory, user preferences, and other factors.

Supplementary Specifications:

1. Color-blind friendly palettes:

- **Description**: The system must provide color palettes that are accessible to users with color vision deficiencies (e.g., red-green color blindness). This will ensure that the user interface is inclusive and accessible to a diverse audience.
- **Example**: Use high-contrast and distinguishable colors in texts, backgrounds, and graphs to ensure important visual elements are visible to all users.

2. Extensible recipe list:

- **Description**: The system must allow easy addition of new recipes without requiring major changes to the core code or design. It should support future expansion to accommodate additional recipe types.
- **Example**: Recipe data should be stored in a flexible format (e.g., CSV or a database) that allows the application to add or modify recipes dynamically.

3. User-friendly interface:

- **Description**: The interface must be intuitive, clutter-free, and easy to navigate, allowing users of all skill levels to interact with the system without extensive training or instruction.
- **Example**: Use clear labeling, logical menu structure, and simple navigation paths to guide users through their tasks.

4. Compatibility:

- **Description**: The application must run smoothly across different smart fridge platforms. It should support various screen sizes and input methods.
- **Example**: Ensure compatibility with the latest versions of smart fridge operating systems.

5. Responsiveness:

- **Description**: The system should maintain its performance and usability, ensuring a smooth user experience and minimal lag.
- **Example**: Implement responsive design techniques to ensure the interface works fast and smooth.

6. Accessibility:

- **Description**: The system must comply with accessibility standards to ensure that users with disabilities, such as visual or motor impairments, can effectively use the application.

- **Example**: Provide assistance in navigation, screen reader compatibility, and alternative text for images to assist users with disabilities.