SmartFridge

# Kravspecifikation

## **Figur 2** Aktørdiagram over SmartFridge Aktører

**Figur 2** viser use case-diagrammet med alle aktører og deres forhold til systemet SmartFridge.

## Aktørbeskrivelse

### Bruger

|  |  |
| --- | --- |
| **Aktørnavn:** | Bruger |
| **Alternativt navn:** |  |
| **Type:** | Primær |
| **Beskrivelse:** | Bruger er den primære bruger af systemet. Bruger interagerer med systemet gennem en brugergrænseflade. |

### Ekstern Database

|  |  |
| --- | --- |
| **Aktørnavn:** | Ekstern Database |
| **Alternativt navn:** | Online Database |
| **Type:** | Sekundær |
| **Beskrivelse:** | Den Eksterne Database er en database, der ligger på en webserver. Den har kun forbindelse til det lokale system når der synkroniseres (se Use Case 7), og fungerer som direkte database for web-app’en. Databasen får tilføjet, fjernet og redigeret data ved synkronisering, eller manipulationen kan foregå fra systemets web app. |

## Termliste

### Fridge app

Fridge app er den lokale del af systemet, og dækker over den lokale brugergrænseflade, samt den lokale database.

### Web app

Web app er den eksterne del af systemet, og dækker over websitet.

### Kernefunktionalitet

Begrebet dækker over de mest basale funktionaliteter, som gør at systemet er sammenhængende og brugbart. Disse funktionaliteter er repræsenteret ved use cases 1-5.

### Standard-varer

Begrebet dækker over en række varer, som Bruger altid ønsker at have i sin varebeholdning. Varerne tilføjes til en liste på lige fod med andre lister i systemet.

Kaldes også ”standard-beholdning”.

## Use cases

For alle use cases gælder det, at i enhver undermenu, findes der altid mulighed for at vende tilbage til hovedmenuen.



**Figur 3** Use case-diagram over SmartFridge

### UC 1: Se varer

Use case 1 har til formål at lade brugeren får overblik over enten varebeholdningen, indkøbslisten eller listen over standard-varer.

|  |  |
| --- | --- |
| **Use case nr./navn** | **1 – Se varer** |
| **Mål** | At få frembragt varerne på en liste. |
| **Initialisering** | Bruger |
| **Aktører** | * Bruger |
| **Samtidige forekomster** | 1 |
| **Forudsætninger** | Applikationen er startet |
| **Resultat** | Indholdet på den valgte liste vises på skærmen. |
| **Hovedscenarie** | 1. Bruger trykker på knappen ”I køleskab”.   **[Alternativt flow 1.a: Bruger trykker på ”Indkøbsliste”]**  **[Alternativt flow 1.b: Bruger trykker på ”Standard-varer”]**   1. En liste over nuværende varer i køleskabet, samt mængden af disse, vises på skærmen. |
| **Alternativt flow** | **1.a Bruger trykker på ”Indkøbsliste”.**  1.a.1 En liste over nuværende varer på indkøbslisten, samt mængden af disse, vises på skærmen.  **1.b Bruger trykker på ”Standard-varer”.**  1.b.1 En liste over nuværende standard-varer, samt mængden af disse, vises på skærmen. |

### UC 2: Tilføj vare

Use case 2 har til formål at lade brugeren tilføje varer til enten varebeholdningen, indkøbslisten eller listen over standard-varer.

Felterne ”Antal” og ”Volumen/vægt” udfyldes automatisk med standardværdier, når varetypen er valgt, men kan ændres efter behov.

|  |  |
| --- | --- |
| **Use case nr./navn** | **2 - Tilføj vare** |
| **Mål** | At tilføje en vare til en liste. |
| **Initialisering** | Bruger |
| **Aktører** | * Bruger * GUI |
| **Samtidige forekomster** | 1 |
| **Forudsætninger** | UC1: Se varer |
| **Resultat** | Varen er tilføjet til en liste. |
| **Hovedscenarie** | 1. Bruger trykker på ”Tilføj”. 2. Varetype vælges.   **[Undtagelse 2.a: Varetypen eksisterer ikke]**   1. Antal vælges. 2. Volumen/vægt vælges 3. Enhed vælges 4. Udløbsdato vælges eventuelt. 5. Bruger trykker på ’Tilføj og afslut,’ og varen tilføjes til listen.   **[Alternativt flow 7.a: Bruger trykker på "Tilføj"]**   1. Der returneres til listen. |
| **Alternativt flow** | **7.a Bruger trykker på "Tilføj".**  7.a.1 Varen gemmes, og der fortsættes fra punkt 2. |
| **Undtagelser** | **2.a Varetypen eksisterer ikke.**  2.a.1 Bruger indtaster den ønskede vare, og fortsætter fra punkt 3. |

### UC 3: Rediger vare

Use case 3 har til formål at lade brugeren redigere varer på enten varebeholdningen, indkøbslisten eller listen over standard-varer.

|  |  |
| --- | --- |
| **Use case nr./navn** | **3 - Rediger vare** |
| **Mål** | At redigere en vare på en den nuværende liste. |
| **Initialisering** | Bruger |
| **Aktører** | * Bruger * GUI |
| **Samtidige forekomster** | 1 |
| **Forudsætninger** | UC1: Se varer |
| **Resultat** | Varen er blevet redigeret. |
| **Hovedscenarie** | 1. Bruger trykker på ”Rediger”. 2. Bruger retter vareinformation.   **[Alternativt flow 2.a: Bruger ændrer Varetype] [Alternativt flow 2.b: Bruger ændrer Antal] [Alternativt flow 2.c: Bruger ændrer Volumen/Vægt] [Alternativt flow 2.d: Bruger ændrer Enhed] [Alternativt flow 2.e: Bruger ændrer intet]**   1. Bruger trykker på ”Gem” og ændringerne gemmes i varen.   **[Alternativt flow 3.a: Bruger trykker på "Annuller"]** |
| **Alternativt flow** | **2.a Bruger ændrer Varetype** 2.a.1 Bruger ændrer varens antal, og der returneres til punkt 2. **2.b Bruger ændrer Antal** 2.b.1 Bruger ændrer varens antal, og der returneres til punkt 2. **2.c Bruger ændrer Volumen/Vægt** 2.c.1 Bruger ændrer varens volumen/vægt, og der returneres til punkt 2. **2.d Bruger ændrer Enhed** 2.d.1 Bruger ændrer varens enhed, og der returneres til punkt 2. **2.e Bruger ændrer intet** 2.e.1 Bruger ændrer ikke noget, og der fortsættes fra punkt 3. **3.a Bruger trykker på "Annuller"** 3.a.1 Ændringer gemmes ikke. |

### UC 4: Fjern vare

Use case 4 har til formål at lade brugeren fjerne varer på enten varebeholdningen, indkøbslisten eller listen over standard-varer.

|  |  |
| --- | --- |
| **Use case nr./navn** | **4 – Fjern vare** |
| **Mål** | At fjerne en vare fra en beholdning. |
| **Initialisering** | Bruger |
| **Aktører** | * Bruger, GUI, Database |
| **Samtidige forekomster** | 1 |
| **Forudsætninger** | UC1: Se varer |
| **Resultat** | Varen er fjernet fra beholdning |
| **Hovedscenarie** | 1. Bruger trykker på ”Fjern”-Ikonet ud for en eksisterende vare, og varen fjernes fra GUI og database. |

### UC 5: Synkroniser til ekstern database

Use case 5 har til formål at lade brugeren initiere en øjeblikkelig synkronisering mellem den lokale og den eksterne database.

|  |  |
| --- | --- |
| **Use case nr./navn** | **5 – Synkroniser til ekstern database** |
| **Mål** | At synkronisere den lokale og den eksterne database. |
| **Initialisering** | Bruger |
| **Aktører** | * Bruger * GUI * Database |
| **Samtidige forekomster** | 1 |
| **Forudsætninger** | At Fridge app er forbundet til internettet. |
| **Resultat** | Den lokale og den eksterne database er synkroniseret. |
| **Hovedscenarie** | 1. Bruger trykker på ”Synkroniser”, og en øjeblikkelig synkronisering påbegyndes. |

## Udvidelser

Følgende udvidelser vil i næste afsnit blive opdelt efter MoSCoW-metoden (DSDM Consortium, u.d.). De vil blive implementeret efter prioritet, efter systemets kernefunktionaliteter er implementeret.

### Sidste type vare fjernes

Når Bruger fjerner den sidste type af en vare, kommer applikationen med en notifikation, for at gøre opmærksom på dette.

### Gammel vare fjernes

Hvis holdbarhedsdatoen er overskredet ift. den dato som Bruger har angivet for en vare, notificeres Bruger. Notifikationen fjernes når Bruger indikerer til systemet at varen er fjernet, eller holdbarhedsdatoen på varen er ændret til en fremtidig dato.

### Tilføj opbevaringssted

Bruger vælger tilføj opbevaringssted, og giver denne et navn. Bruger kan nu vælge dette sted som opbevaringsplads, når en ny vare tilføjes eller redigeres.

### Log in

Når Bruger vil tilgå web-app'en, skal der først logges ind. Når bruger er logget ind, kan egen del af den eksterne database tilgås, og de sædvanlige funktioner vil være tilgængelige.

### Importer indkøbsliste

Når Bruger har handlet ind, kan alle varer på indskøbslisten med ét tryk overføres til varebeholdningen.

### Scan vare

En stregkodescanner tilsluttes systemet, og varer kan scannes. Varerne tilføjes til en valgfri varebeholdningsliste.

### Vis ernæringsværdier

Bruger kan få oplyst ernæringsværdier for de enkelte varer.

### Find opskrift

Bruger kan finde opskrifter baseret på de tilgængelige varer. Applikationen kan ud fra den valgte opskrift danne en indkøbsliste, med evt. manglende varer.

### Valg af tema

Bruger får mulighed for at skifte grafisk tema på applikationen.

### Juster temperaturalarm

Et termometer, som kan kommunikere med Fridge app’en, lægges i køleskabet, og Bruger sætter en max.- og en min.-temperatur. Kommer temperaturen uden for de satte værdier, advares Bruger.

### Find tilbud

Bruger vælger ”Find tilbud” på den færdige indkøbsliste. Applikationen finder nu selv de supermarkeder hvor der er tilbud på de varer, der er på indkøbslisten. De butikker hvor alle nødvendige varer sammenlagt kan købes billigst, kommer først.

## MoSCow

### Must

Disse krav skal implementeres i det endelige produkt for at det er acceptabelt:

* Muligheden for at tilføje en vare.
* Muligheden for at fjerne en vare.
* Muligheden for at redigere en vares information.
* Muligheden for at se en liste over varer.
* Mulighed for at synkronisere med en ekstern database.
* Varer der ikke findes i ”Køleskab”, og som er tilføjet på ”Standard-listen”, tilføjes automatisk til indkøbslisten.

### Should

Disse krav har, ligesom i must-sektionen, høj prioritet. Men kravene er ikke essentielle for at systemet fungerer og kan benyttes.

* En påmindelse af manglende vare på en af listerne.
* En notifikation for at en vare har overskredet dens holdbarhedsdato.
* Muligheden for at tilføjelse flere skabe.
* Et log-in-system, så må kan være flere brugere om samme system, samt af sikkerhedsmæssige årsager.

### Could

Disse funktioner kunne være en del af systemet. Det er alle krav, hvor implementeringen er ret tidskrævende, og de er heller ikke essentielle for at systemet virker. De hører derfor ind under ”Nice-To-Have”-kategorien.

* Mulighed for at se ernæringsværdier for fødevarerne på listerne.
* Mulighed for at se opskrifter, baseret på de varer, der befinder sig på køleskabs-listen.
* En pænere og mere interaktivt grafisk brugergrænseflade; f.eks. at køleskabslisten ses som ”hylde” med drag-n-drop items.
* Indhentning af tilbud på de varer, der findes på indkøbslisten.

### Would/Won’t

Disse funktioner bliver ikke tilføjet til systemet pga. tid, penge og relevans. Det ville tage lang tid programmere drivere til enhederne, og koste penge at købe selve enhederne.

* En scanner, så der er mulighed for at scanne stregkoden på de nyindkøbte vare, og dermed tilføje dem listerne.
* En temperatursensor, der evt. gør brug af Bluetooth, så temperaturen i køleskabet kan overvåges.

## Ikke-funktionelle krav

Alle krav er specificeret ud fra Lenovo Yoga 2 Pro (Wupti.com, 2014) som platform.

1. **System**
   1. Kernefunktionaliteterne skal kunne udføres i både Web app og Fridge app.
2. **Databaser**
   1. Den lokale og den eksterne database skal automatisk synkroniseres hvert 10. minut.
   2. I tilfælde af konflikter ved synkronisering, overskriver de nyest tilføjede data de ældste.
3. **Fridge app**
   1. Ved opstart og nedluk, forsøges synkronisering mellem den lokale og den eksterne database.
   2. Ændringer af data lagres straks i den lokale database.
   3. En knap/et ikon på skærmen skal indikere status for synkronisering.
      1. Synkroniseret
      2. Ikke synkroniseret
      3. Ingen forbindelse
   4. Responstiden for navigation må maksimalt være to sekunder.
   5. Skal kunne anvendes uden internetforbindelse.
4. **Web app**
   1. Ændringer af data lagres straks i den eksterne database.