Use Case – Terminplanung

**Use Case 1 CRUD-Kundenterminplanung**

**Primary Actor:**

Mosterei-Inhaber

**Stakeholders and Interests:**

* Mosterei-Inhaber: Termine sollen leicht in einer Kalenderübersicht zu verwalten sein (löschen, erstellen, ändern)
* Mosterei-Mitarbeiter: Übersichtlicher Terminplan, mit Kundeninformation (Name, Menge des Obstes), schnelle Abfertigung
* Kunde: schneller Service

**Preconditions:**

Chef ist im Programm angemeldet(mit ID und Passwort) um Terminplan abrufen/bearbeiten zu können, es sollte eine Kundenliste geben(ansonsten zuerst Kunde anlegen)

**Success Guarantee(Postconditions):**

Terminplanung ist abgeschlossen und ein übersichtlicher, lückenfreier Plan wurde erstellt.

**Main Success Scenario(or Basic Flow):**

1. Kunde ruft bei Mosterei an /oder kommt zur Mosterei um einen Termin zu machen/abzusagen
2. Mosterei-Inhaber öffnet die Terminplanung mit Kalenderübersicht
3. System erlaubt den Zugriff und die Bearbeitung auf die Terminplanung
4. Mosterei-Inhaber sucht mit Kunden einen freien Termin, bzw. kann über eine Suchfunktion, den bereits bestehenden Termin eines Kunden, mit Hilfe des Nachnamens suchen und gegeben falls löschen/verschieben (hierbei ist auch wichtig wie viel Obst gepresst werden soll(Zeitmanagement))
5. Über einen Speicherbutton kann der Termin/die Änderung abgespeichert werden, auch soll beim Schließen des Fensters Terminplanung, erscheint ein Fenster „Änderungen Speichern?“, so soll vermieden werden, dass die Termine verloren gehen
6. (Mosterei-Inhaber loggt sich aus)

**Extensions(or Alternativ Flows):**

\*a. Zu jeder Zeit, wenn das System ausfällt:

ERGÄNZUNG(Eventuell das unser Programm Daten zwischenspeichert von allein?)

1. Mosterei-Inhaber/Mitarbeiter startet das System neu
2. System konstruiert vorherigen Zustand
   1. System erkennt Gefahr beim Vorbereiten des vorherigen Zustandes
      1. System gibt eine Fehlerwarnung aus und vermerkt den Fehler, übergibt aber einen sauberen Zustand
      2. Mosterei-Inhaber startet die Terminplanung neu

\*b. Falsche ID: System gibt einen Fehler aus, erneute Eingabe möglich

\*c. Falsches Passwort: System gibt einen Fehler aus, erneute Eingabe möglich

\*d. Belegte Termine sollen rot, teils belegte Termine grün, freie Termine ohne Farbe hinterlegt sein

\*e. (Termin kann für anwesenden Kunden ausgedruckt werden oder als Emailbestätigung geschickt

werden)

**Special Requirements:**

* Internationalisierung der Sprache (Englisch, Deutsch)
* Lesen von ca. 1 Meter Entfernung soll möglich sein
* In 90% der Fälle soll das Programm keine lange Ladedauer haben

**Technology and Data Variations List:**

Eine Schnittstelle mit einem Drucker sollte vorhanden sein.

**Frequency of Occurrence:**

Bei jeder Terminplanung notwendig

**Open Issues:**

* Was für ein Kalender ist sinnvoll für unser Programm (Monatskalender, Tageskalender mit oder ohne Uhrzeiten)

**Use Case 2 CRUD-Kundenterminplanung**

**Primary Actor:**

Mosterei-Mitarbeiter

**Stakeholders and Interests:**

* Mosterei-Inhaber: Termine sollen leicht in einer Kalenderübersicht zu verwalten sein (löschen, erstellen, ändern)
* Mosterei-Mitarbeiter: Übersichtlicher Terminplan, mit Kundeninformation (Name, Menge des Obstes), schnelle Abfertigung
* Kunde: schneller Service

**Preconditions:**

Mitarbeiter ist im Programm(mit ID und Passwort) angemeldet um Terminplan abrufen zu können

**Success Guarantee(Postconditions):**

Terminplanung ist abgeschlossen und ein übersichtlicher, lückenfreier Plan wurde erstellt.

**Main Success Scenario(or Basic Flow):**

1. Mosterei-Mitarbeiter öffnet die Terminplanung
2. System erlaubt den Zugriff auf die Terminplanung aber nicht die Bearbeitung
3. (Mosterei-Mitarbeiter loggt sich aus)

**Extensions(or Alternativ Flows):**

\*a. Zu jeder Zeit, wenn das System ausfällt:

ERGÄNZUNG(Eventuell das unser Programm Daten zwischenspeichert von allein?)

1. Mosterei-Inhaber/Mitarbeiter startet das System neu
2. System konstruiert vorherigen Zustand
   1. System erkennt Gefahr beim Vorbereiten des vorherigen Zustandes
      1. System gibt eine Fehlerwarnung aus und vermerkt den Fehler, übergibt aber einen sauberen Zustand
      2. Mosterei-Inhaber startet die Terminplanung neu

\*b. Falsche ID: System gibt einen Fehler aus, erneute Eingabe möglich

\*c. Falsches Passwort: System gibt einen Fehler aus, erneute Eingabe möglich

\*d. Belegte Termine sollen rot, teils belegte Termine grün, freie Termine ohne Farbe hinterlegt sein

\*e. Terminplan kann ausgedruckt werden

**Special Requirements:**

* Internationalisierung der Sprache (Englisch, Deutsch)
* Lesen von ca. 1 Meter Entfernung soll möglich sein
* In 90% der Fälle soll das Programm keine lange Ladedauer haben

**Technology and Data Variations List:**

Eine Schnittstelle mit einem Drucker sollte vorhanden sein.

**Frequency of Occurrence:**

Bei jeder Terminplanung notwendig

**Open Issues:**

* Was für ein Kalender ist sinnvoll für unser Programm (Monatskalender, Tageskalender mit oder ohne Uhrzeiten)

**Use Case 1 CRUD-Mitarbeiterschichtplan/Mitarbeiterterminplanung**

**Primary Actor:**

Mosterei-Inhaber

**Stakeholders and Interests:**

* Mosterei-Inhaber: Mitarbeiterschichtplan soll einfach und schnell zu handhaben sein (erstellen, ändern, löschen)
* Mosterei-Mitarbeiter: Schichtplan soll übersichtlich sein und nach gewünschten Arbeitsmöglichkeiten sein

**Preconditions:**

Chef ist Programm angemeldet(mit ID und Passwort) um Schichtplan abrufen/bearbeiten zu können, es muss eine Mitarbeiterliste geben

**Success Guarantee(Postconditions):**

Schichtplan ist abgeschlossen und ein übersichtlicher, lückenfreier Plan wurde erstellt.

**Main Success Scenario(or Basic Flow):**

1. Mitarbeiter gibt dem Mosterei-Inhaber Information über Tage an denen er arbeiten kann
2. Mosterei-Inhaber öffnet die Mitarbeiterschichtplan
3. System erlaubt den Zugriff und Bearbeitung auf den Mitarbeiterschichtplan
4. Mosterei-Inhaber erstellt durch die Information seiner Mitarbeiter einen Schichtplan
5. Über einen Speicherbutton kann der Plan/die Änderung abgespeichert werden, auch soll beim Schließen des Fensters Terminplanung, erscheint ein Fenster „Änderungen Speichern?“, so soll vermieden werden, dass die Daten verloren gehen
6. Mitarbeiter bekommt seinen Plan/hat Zugriff auf seinen Plan
7. (Mosterei-Inhaber loggt sich aus)

**Extensions(or Alternativ Flows):**

\*a. Zu jeder Zeit, wenn das System ausfällt:

ERGÄNZUNG(Eventuell das unser Programm Daten zwischenspeichert von allein?)

1. Mosterei-Inhaber/Mitarbeiter startet das System neu
2. System konstruiert vorherigen Zustand
   1. System erkennt Gefahr beim Vorbereiten des vorherigen Zustandes
      1. System gibt eine Fehlerwarnung aus und vermerkt den Fehler, übergibt aber einen sauberen Zustand
      2. Mosterei-Inhaber startet die Terminplanung neu

\*b. Falsche ID: System gibt einen Fehler aus, erneute Eingabe möglich

\*c. Falsches Passwort: System gibt einen Fehler aus, erneute Eingabe möglich

\*d. (Plan kann für Mitarbeiter ausgedruckt werden)

**Special Requirements:**

* Internationalisierung der Sprache (Englisch, Deutsch)
* Lesen von ca. 1 Meter Entfernung soll möglich sein
* In 90% der Fälle soll das Programm keine lange Ladedauer haben

**Technology and Data Variations List:**

Eine Schnittstelle mit einem Drucker sollte vorhanden sein.

**Frequency of Occurrence:**

Bei jedem Schichtplan nötig

**Open Issues:**

* Urlaubstage, kann der Mitarbeiter selber eintragen? (Zumindest wo er gern hätte)
* Krankheitsausfälle?

**Use Case 2 CRUD-Mitarbeiterschichtplan/Mitarbeiterterminplanung**

**Primary Actor:**

Mosterei-Mitarbeiter

**Stakeholders and Interests:**

* Mosterei-Inhaber: Mitarbeiterschichtplan soll einfach und schnell zu handhaben sein (erstellen, ändern, löschen)
* Mosterei-Mitarbeiter: Schichtplan soll übersichtlich sein und nach gewünschten Arbeitsmöglichkeiten sein

**Preconditions:**

Mitarbeiter ist im Programm angemeldet(mit ID und Passwort) um Schichtplan abrufen zu können

**Success Guarantee(Postconditions):**

Schichtplan ist abgeschlossen und ein übersichtlicher, lückenfreier Plan wurde erstellt.

**Main Success Scenario(or Basic Flow):**

1. Mosterei-Mitarbeiter öffnet die Mitarbeiterschichtplan
2. System erlaubt den Zugriff auf den Mitarbeiterschichtplan, aber nicht die Bearbeitung des Plans
3. (Mosterei-Mitarbeiter loggt sich aus)

**Extensions(or Alternativ Flows):**

\*a. Zu jeder Zeit, wenn das System ausfällt:

ERGÄNZUNG(Eventuell das unser Programm Daten zwischenspeichert von allein?)

1. Mitarbeiter startet das System neu
2. System konstruiert vorherigen Zustand
   1. System erkennt Gefahr beim Vorbereiten des vorherigen Zustandes
      1. System gibt eine Fehlerwarnung aus und vermerkt den Fehler, übergibt aber einen sauberen Zustand
      2. Mosterei-Inhaber startet die Terminplanung neu

\*b. Falsche ID: System gibt einen Fehler aus, erneute Eingabe möglich

\*c. Falsches Passwort: System gibt einen Fehler aus, erneute Eingabe möglich

\*d. Plan kann von Mitarbeiter ausgedruckt werden

**Special Requirements:**

* Internationalisierung der Sprache (Englisch, Deutsch)
* Lesen von ca. 1 Meter Entfernung soll möglich sein
* In 90% der Fälle soll das Programm keine lange Ladedauer haben

**Technology and Data Variations List:**

Eine Schnittstelle mit einem Drucker sollte vorhanden sein.

**Frequency of Occurrence:**

Bei jedem Schichtplan nötig

**Open Issues:**

* Urlaubstage, kann der Mitarbeiter selber eintragen? (Zumindest wo er gern hätte)
* Krankheitsausfälle?