Use Cases von Mosti

Inhalt

[Use Case UC1: Verkauf abwickeln 2](#_Toc448058793)

[Use Case UC2: Preis für Dienstleistung bearbeiten 4](#_Toc448058794)

[Use Case UC3: Lagerbestand bearbeiten 6](#_Toc448058795)

[Use Case UC4: Kunde bearbeiten 8](#_Toc448058796)

[Use Case UC5: Mitarbeiter bearbeiten 10](#_Toc448058797)

[Use Case UC6: Adminaccount anlegen 13](#_Toc448058798)

[Use Case UC7: Passwort ändern 14](#_Toc448058799)

[Use Case UC8: CRUD Kundentermin bearbeiten 15](#_Toc448058800)

[Use Case UC9: CRUD Mitarbeiterschicht bearbeiten 17](#_Toc448058801)

# Use Case UC1: Verkauf abwickeln

**Primary Actor:** Kassierer

**Stakeholders and Interests:**

* Kassierer: Will übersichtliche Darstellung der angebotenen Produkte. Will, dass Abwicklung eines Verkaufs kundenspezifisch erfolgt und nach Abschluss automatisch gespeichert wird. Will eine Kundenauflistung nach aktuellem Terminplan. Will einfache und schnelle Eingabe der einzelnen Kundeneinkäufe und korrekte Ermittlung der Rechnungssumme für die Kunden. Will automatische Berücksichtigung von Kunden-Besonderheiten (z.B. Rabatt) und Aktualisierung der Lagerbestände nach jedem Verkauf von Materialien.
* Kunde: Will schnelle und korrekte Ermittlung der Rechnungssumme. Will persönliche Einkaufsübersicht in Form eines Belegs direkt nach dem Einkauf (insbesondere mit Angaben über die erzeugte Literzahl, Berücksichtigung kundenspezifischer Besonderheiten wie z.B. Rabatt). Will vertrauliche Behandlung der erfassten Daten.
* Mosterei-Inhaber: Will korrekte, schnelle Verkaufsabwicklung durch schnelle und übersichtliche Eingabe. Will Speicherung der abgeschlossenen Einkäufe (mind. der verkauften Literzahl) für spätere Auswertungen oder auch Kundenrückfragen. Will Aktualisierung der Lagerbestände nach jedem Verkauf von Materialien.
* Jäger: Will korrekte Erfassung und Speicherung der an einem Tag verkauften Literzahlen (Gesamttages-Literzahl zur Berechnung des Rechnungsbetrags für den Trester).
* Mosterei-Anlagen-Firma: Will korrekte Erfassung und Speicherung der erzeugten Literzahlen in einem bestimmten Zeitraum, um Rückschlüsse auf die Funktionalität der Maschinen ziehen zu können.

**Preconditions:**

Kassierer ist eingeloggt/authentifiziert. Aktueller Tages-Terminplan liegt vor.

**Postconditions:**

Verkauf ist korrekt abgewickelt, abgeschlossen und gespeichert. Ausdruckbare Auflistung des jeweiligen Kundeneinkaufs wurde erstellt. Lagerbestände wurden aktualisiert

**Main Success Scenario**

1. Kassierer wählt einen Kunden aus der tagesaktuellen Terminliste aus.
2. System zeigt die Einkäufe-Eingabe-Maske für diesen Kunden an.
3. Kassierer trägt die Anzahl der vom Kunden verbrauchten Materialien ein (incl. Literzahl und Abfüllweise).
4. System bildet die Summe der Kosten für die aktuell eingetragenen Produkte.

*Schritte 3-4 wiederholen sich solange, bis der Kassierer alle der vom Kunden zu erwerbenden Produkte in der Eingabemaske erfasst hat.*

1. Kassierer schließt die Gesamt-Kostenermittlung endgültig ab.
2. System erstellt eine Auflistung des Kundeneinkaufs und stellt sie zum Ausdruck bereit.
3. Kassierer bestätigt die Zahlung durch den Kunden.
4. System markiert den Kunden als abgeschlossen, speichert die Verkaufsinformationen und aktualisiert die Lagerbestände.

**Extensions:**

\*a. Jederzeit, wenn das System ausfällt:

Um Wiederherstellung zu gewährleisten, sicherstellen, dass alle

Transaktionsrelevanten Zustände und Ereignisse in jedem Schritt des Szenarios

wiederhergestellt werden können.

1. Mosterei-Inhaber startet das System neu, meldet sich an und fordert Wiederherstellung eines früheren Zustandes an.
2. System rekonstruiert früheren Zustand.

2a. System entdeckt Anomalien, die Wiederherstellung verhindern:

1. System signalisiert Fehler zum Mosterei-Inhaber, speichert den Fehler ab und geht in einen sauberen Zustand über.
2. Mosterei-Inhaber startet Programm neu.

1a. Aktueller Kunde ist trotz Termin nicht in der Terminliste vorhanden.

* 1. Kassierer fordert Zusatzmaske an.
  2. System zeigt erweiterbare Zusatzkunden-Liste an
  3. Kassierer prüft, ob Kunde in der Kundendatenbank registriert ist.

3a. Kunde ist in der Kundendatenbank registriert

3b. Kunde ist nicht in der Kundendatenbank registriert

1. UC2: CRUD Kundenverwaltung, Extension 1a. ausführen.

* 1. Kassierer fügt Kunden zur Zusatzkunden-Liste hinzu und wählt den Kunden aus.

3a. Gesuchtes Produkt ist nicht in der Eingabemaske verfügbar.

1. UC3: CRUD Lagerverwaltung, Extension 1a. ausführen.
2. System zeigt aktualisierte Eingabemaske an.

**Special Requirements:**

* Schriftgröße mindestens 12.
* Sprache: deutsch.

**Technology/Data Variations List:**

* Ausdruck der Kostenzusammenstellung/des Belegs mittels Drucker.
* Speicherung der Verkäufe in Datenbank.

**Frequency of Occurrence:**

50-80-mal am Tag, bis zu 10-mal pro Stunde.

**Open Issues:**

* Integration der Terminliste in die Kassenfunktion – wie?
* Einbezug von kundenspezifischer Begünstigungen (Verknüpfung mit Kundenverwaltung)
* Extra Abrechnung für Jäger – mit nur Literzahl und entsprechender Preisumrechnung?

# Use Case UC2: Preis für Dienstleistung bearbeiten

**Primary Actor:** Mosterei-Inhaber

**Stakeholder and Interests:**

* Mosterei-Inhaber: Will, dass maschineller Aufwand sowie unterschiedlich aufwändige Abfüllweisen berücksichtigt und ebenso berechnet werden können. Will die aktuellen Preise schnell und einfach bearbeiten können.
* Mitarbeiter: Will mit den richtigen/aktuellen Preisen abrechnen.
* Kunde: Will, dass sein Einkauf nach den korrekten/aktuellen Preisen abgerechnet wird.

**Preconditions:**

Der Mosterei-Inhaber ist im System eingeloggt/authentifiziert.

**Postconditions:**

Preis für Dienstleistung wurde aktualisiert.

**Main Success Scenario:**

1. Mosterei-Inhaber fordert Preisänderung einer Dienstleistung an.
2. System zeigt Übersicht mit den einzelnen Dienstleistungen und deren Preise.
3. Mosterei-Inhaber wählt Dienstleistung aus und nimmt gewünschte Änderungen vor.
4. Mosterei-Inhaber bestätigt Änderungen.
5. System speichert die Daten.

**Extensions:**

\*a. Jederzeit, wenn das System ausfällt:

Um Wiederherstellung zu gewährleisten, sicherstellen, dass alle

Transaktionsrelevanten Zustände und Ereignisse in jedem Schritt des Szenarios

wiederhergestellt werden können.

1. Mosterei-Inhaber startet das System neu, meldet sich an und fordert Wiederherstellung eines früheren Zustandes an.
2. System rekonstruiert früheren Zustand.
   1. System entdeckt Anomalien, die Wiederherstellung verhindern:
      1. System signalisiert Fehler zum Mosterei-Inhaber, speichert den Fehler ab und geht in einen sauberen Zustand über.
      2. Mosterei-Inhaber startet Programm neu.

4-5a. Mosterei-Inhaber bestätigt die Änderungen nicht.

1. System zeigt Übersicht mit den einzelnen Dienstleistungen und deren Preise.

**Special Requirements:**

* Schriftgröße mindestens 12.
* Sprache: deutsch.
* Lesen von ca. 1 Meter Entfernung soll möglich sein.
* In 90% der Fälle soll die Übersicht in weniger als 10s geladen sein.

**Technology/Data Variations List:**

* Speicherung der Preise in einer Datenbank
* Verbindung bzw. Zugriff auf Datenbank

**Frequency of Occurrence:**

Einmal jährlich.

**Open Issues:**

* Voreingestellte Preise?

# Use Case UC3: Lagerbestand bearbeiten

**Primary Actor:** Mosterei-Inhaber

**Stakeholders and Interests:**

* Mosterei-Inhaber: Will klare Übersicht über aktuelle Lagerbestände. Will Produkte schnell und einfach in der Lagerübersicht hinzufügen, bearbeiten oder löschen. Will Meldung/Warnung, falls bestimmte Untergrenze eines Produkts unterschritten wird.
* Kunde: Will stets gewünschte Produkte zur Verfügung haben.

**Preconditions:**

Mosterei-Inhaber ist eingeloggt/authentifiziert.

**Postconditions:**

Lagerbestände sind produktweise vollständig und aktuell.

**Main Success Scenario**

1. Mosterei-Inhaber fordert Anzeige einer Lagerübersicht.
2. System zeigt eine Übersicht über die Lagerbestände an.

**Extensions:**

\*a. Jederzeit, wenn das System ausfällt:

Um Wiederherstellung zu gewährleisten, sicherstellen, dass alle

Transaktionsrelevanten Zustände und Ereignisse in jedem Schritt des Szenarios

wiederhergestellt werden können.

1. Mosterei-Inhaber startet das System neu, meldet sich an und fordert Wiederherstellung eines früheren Zustandes an.
2. System rekonstruiert früheren Zustand.
   1. System entdeckt Anomalien, die Wiederherstellung verhindern:
      * 1. System signalisiert Fehler zum Mosterei-Inhaber, speichert den Fehler ab und geht in einen sauberen Zustand über.
        2. Mosterei-Inhaber startet Programm neu.

1a. Mosterei-Inhaber erstellt ein neues Produkt:

1. System fordert Produktname, Einzelverkaufspreis für Produkt, aktuell zu Verfügung stehende Produktmenge und Untergrenze für Warnhinweis an.
2. Mosterei-Inhaber gibt geforderte Daten ein.
3. System legt neues Produkt an und speichert die Produktdaten.

1b. Mosterei-Inhaber ändert die Angaben zu einem Produkt:

1. System speichert die neuen Änderungen.

1c. Mosterei-Inhaber löscht ein Produkt:

1. System erfragt Bestätigung des Löschvorgangs.
2. Mosterei-Inhaber bestätigt Löschvorgang.
3. System löscht Produkt und speichert neue Änderungen.

**Special Requirements:**

* Schriftgröße mindestens 12.
* Sprache: deutsch.
* Lesen von ca. 1 Meter Entfernung soll möglich sein.
* In 90 % der Fälle soll die Übersicht in weniger als 10 Sekunden geladen sein.

**Technology/Data Variation List:**

Speicherung der Lagerbestände in einer Datenbank.

**Frequency of Occurrence:**

Ein- bis zweimal im Monat.

**Open Issues:**

* Aktualisierung in Zusammenhang mit Kassenfunktion
* Meldungen?

# Use Case UC4: Kunde bearbeiten

**Primary Actor:** Mosterei-Inhaber

**Stakeholders and Interests:**

* Mosterei-Inhaber: Will einen Kunden schnell und einfach hinzufügen, bearbeiten oder löschen. Will Übersicht über Kunden, welche seine Leistungen in Anspruch nehmen.
* Mitarbeiter: Will bestimmte Kundendaten schnell und einfach einsehen (z. B. die Telefonnummer im Falle eines Nichterscheinens trotz Termins). Will einen Kunden schnell und einfach hinzufügen, bearbeiten oder löschen.
* Kunde: Will, dass seine Daten korrekt abgespeichert sind und vertrauenswürdig behandelt werden. Will, dass auf seinen Wunsch hin die Daten geändert werden.

**Preconditions:**

Mosterei-Inhaber oder Mitarbeiter ist eingeloggt/authentifiziert.

**Postconditions:**

Kundenliste wird angezeigt.

**Main Success Scenario**

1. Mosterei-Inhaber fordert Anzeige einer Kundenübersicht.
2. System zeigt eine Liste aller Kunden an.

**Extensions:**

1a. Mosterei-Inhaber oder Mitarbeiter erstellt einen neuen Kunden:

* 1. System fordert Name, Vorname, Adresse, Telefonnummer des Kunden.
  2. Mosterei-Inhaber oder Mitarbeiter gibt geforderte Daten ein.
  3. System legt neuen Kunden an.

1b. Mosterei-Inhaber oder Mitarbeiter ändert die Daten eines Kunden:

1. System speichert die neuen Änderungen.

1c. Mosterei-Inhaber oder Mitarbeiter löscht die Daten eines Kunden:

1. System erfragt Bestätigung des Löschvorgangs.
2. Mosterei-Inhaber oder Mitarbeiter bestätigt Löschvorgang.
3. System speichert die neuen Änderungen.

**Special Requirements:**

* Schriftgröße mindestens 12.
* Sprache: deutsch.
* Lesen von ca. 1 Meter Entfernung soll möglich sein.
* In 90 % der Fälle soll die Übersicht in weniger als 10 Sekunden geladen sein.

**Technology and Data Variations List:**

* Speicherung der Kundendaten in Datenbank.
* Verbindung zur Datenbank, um geänderte Kundendaten abzuspeichern.

**Frequency of Occurence:**

Mehrmals pro Stunde.

**Open Issues:**

* Sollen Kundendaten nach bestimmter Zeit gelöscht werden?
* Müssen Kunden immer angelegt sein, um einen Einkauf zu tätigen?

# Use Case UC5: Mitarbeiter bearbeiten

**Primary Actor:** Mosterei-Inhaber

**Stakeholders and Interests:**

* Mosterei-Inhaber: Will neue Mitarbeiter schnell und einfach erstellen oder bearbeiten. Will eine Übersicht über seine Angestellten samt seiner Kontaktdaten, um ihn bei Bedarf kontaktieren zu können.
* Mitarbeiter: Will, dass seine Daten korrekt abgespeichert werden. Will Zugriff auf seine Daten, um sie bei Bedarf ändern zu können.

**Preconditions:**

Mosterei-Inhaber ist im System eingeloggt/authentifiziert.

**Postconditions:**

Mitarbeiterliste wird angezeigt.

**Main Success Scenario:**

1. Mosterei-Inhaber fordert Anzeige einer Mitarbeiterliste an.
2. System zeigt eine Liste aller Mitarbeiter an.

**Extensions:**

\*a. Jederzeit, wenn das System ausfällt:

Um Wiederherstellung zu gewährleisten, sicherstellen, dass alle

Transaktionsrelevanten Zustände und Ereignisse in jedem Schritt des Szenarios

wiederhergestellt werden können.

1. Mosterei-Inhaber startet das System neu, meldet sich an und fordert Wiederherstellung eines früheren Zustandes an.
2. System rekonstruiert früheren Zustand.
   1. System entdeckt Anomalien, die Wiederherstellung verhindern:
      * 1. System signalisiert Fehler zum Mosterei-Inhaber, speichert den Fehler ab und geht in einen sauberen Zustand über.
        2. Mosterei-Inhaber startet Programm neu.

1a. Mosterei-Inhaber erstellt einen neuen Mitarbeiter:

1. System fordert Vor- und Nachname, Geburtsdatum, Adresse und Telefon- bzw. Handynummer des Mitarbeiters.
2. Mosterei-Inhaber gibt geforderte Daten ein.
3. System legt neuen Mitarbeiter an.
4. System erstellt einen Account mit Benutzernamen und Passwort für den neuen Mitarbeiter und präsentiert die gespeicherten Daten.

1b. Mosterei-Inhaber ändert die Daten eines Mitarbeiters:

1. System speichert die neuen Änderungen.

1c. Mosterei-Inhaber wählt Mitarbeiter und fordert Löschung an:

1. System fordert Bestätigung des Löschvorgangs.
2. Mosterei-Inhaber bestätigt Löschvorgang.
3. System löscht alle Daten und den Account des Mitarbeiters.

**Special Requirements:**

* Schriftgröße mindestens 12.
* Sprache: deutsch.
* Lesen von ca. 1 Meter Entfernung soll möglich sein.
* In 90% der Fälle soll die Übersicht in weniger als 10s geladen sein.

**Technology and Data Variations List:**

* Speicherung der Mitarbeiter in einer Datenbank.
* Verbindung zur Datenbank, um geänderte Mitarbeiterdaten abzuspeichern.

**Frequency of Occurence:**

Ein- bis zweimal im Monat.

**Open Issus:**

# Use Case UC6: Adminaccount anlegen

**Primary Actor:** Mosterei-Inhaber

**Stakeholder and Interests:**

* Mosterei-Inhaber: Will als Einziger Administratorrechte für das System erhalten.

**Preconditions:**

Software wurde noch nie auf diesem Computer ausgeführt.

**Postconditions:**

Neuer Admin-Account wurde erstellt.

**Main Success Scenario:**

1. Mosterei-Inhaber startet zum ersten Mal das System.
2. System fordert Name, Adresse, Emailadresse und Telefonnummer.
3. Mosterei-Inhaber gibt geforderte Daten an.
4. System fordert Passwort mit Wiederholung.
5. Mosterei-Inhaber gibt Passwort zweimal ein.
6. System zeigt Zusammenfassung aller eingegebenen Daten (außer Passwort).
7. Mosterei-Inhaber bestätigt Daten.
8. System speichert Daten und legt neuen Admin-Account an.

**Extensions:**

5a. Passwörter stimmen nicht überein

1. System signalisiert Fehler und fordert neue Eingabe.

2. Mosterei-Inhaber gibt die Passwörter erneut ein.

7a. Mosterei-Inhaber entdeckt Fehler in seinen Angaben

1. Mosterei-Inhaber fordert Bearbeitung der Daten an.

2. System kehrt zur Bearbeitung der Daten zurück.

3. Mosterei-Inhaber korrigiert fälschliche Eingaben und bestätigt.

**Special Requirements:**

* Schriftgröße von mindestens 12.
* Sprache: deutsch.

**Technology/Data Variations List**

Verbindung zur Datenbank, um neuen Account zu speichern.

**Frequency of Occurence:**

Nur einmal.

**Open Issues:**

* Wie wird sichergestellt, dass Admin-Account nur beim ersten Mal erstellt wird?

# Use Case UC7: Passwort ändern

**Primary Actor:** Mitarbeiter

**Stakeholder & Interests:**

* Mitarbeiter: Möchte auf als Einziger auf seine Daten zugreifen können, möchte sicheres Passwort.
* Mosterei-Inhaber: Möchte, dass sich seine Mitarbeiter jederzeit im System anmelden können.

Preconditions:

Mitarbeiter besitzt gültigen Account.

Postconditions:

Neues Passwort wurde abgespeichert und mit dem Mitarbeiterkonto verknüpft.

Main Success Scenario:

1. Mitarbeiter loggt sich im System ein.
2. Mitarbeiter fordert Änderung des Passworts an.
3. System fordert altes und neues Passwort mit Wiederholung an.
4. Mitarbeiter gibt altes einmal und neues Passwort zweimal ein und bestätigt Eingabe.
5. System prüft Passwörter auf Übereinstimmung, speichert das neue Passwort ab und bestätigt den abgeschlossenen Vorgang an den Mitarbeiter.

Extensions:

1.a Mitarbeiter kann sich wegen vergessenen Passwort nicht einloggen

1. Mitarbeiter fordert komplett neues Passwort an.

2. System erfragt Benutzernamen des Mitarbeiters.

3. Mitarbeiter gibt Benutzernamen ein.

4. System erstellt neues Passwort, verknüpft es mit diesem Account und stellt dem Mosterei-Inhaber das neue Passwort zur Verfügung.

5.a Passwörter stimmen nicht überein.

1. System signalisiert Fehler und fordert neue Eingabe.

2. Mitarbeiter gibt Passwörter erneut ein und bestätigt.

Special Requirements:

* Schriftgröße von mindestens 12.
* Sprache: deutsch.

Technology/Data Variations List:

Zugriff auf Datenbank, um dort das neue Passwort zu speichern.

Frequency of Occurence:

Eher selten.

Open Issues:

* Wie wird der Chef über das neue Passwort informiert?

# Use Case UC8: CRUD Kundentermin bearbeiten

Primary Actor: Mosterei-Inhaber

Stakeholders and Interests:

* Mosterei-Inhaber: Will Termine leicht in einer Kalenderübersicht verwalten können (erstellen, ändern, löschen).Will schnelle Übersicht über freie bzw. mögliche Termine.
* Mosterei-Mitarbeiter: Will übersichtlichen Terminplan, mit Kundeninformation (Name, Menge des Obstes).
* Kunde: Will schnellen Service.

Preconditions:

Mosterei-Inhaber ist im System eingeloggt/authentifiziert. Kalender mit Eingabemaske ist vorhanden.

Postconditions:

Kunden-Terminplan wird angezeigt.

Main Success Scenario:

1. Mosterei-Inhaber fordert Anzeige des Kundenterminplans für einen bestimmten Tag an.
2. System zeigt eine Liste aller Mitarbeiter an.

Extensions:

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

1. Mosterei-Inhaber/Mitarbeiter startet das System neu
2. System konstruiert vorherigen Zustand

2a.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

1a. Kunde fordert eine Terminvereinbarung:

1. Mosterei-Inhaber gibt geschätzte Abfüllmenge in das System ein.
2. System berechnet voraussichtliche Terminlänge.
3. Mosterei-Inhaber öffnet die Terminplanung mit Kalenderübersicht.
4. System zeigt momentanen Terminplan an.
5. Mosterei-Inhaber wählt mit Kunden einen freien Termin.
6. Mosterei-Inhaber sucht Kunden in der Datenbank und verknüpft ihn mit Termin.
7. Mosterei-Inhaber trägt Termin in das System ein und bestätigt.
8. System speichert die Änderungen.

1b. Mosterei-Inhaber ändert einen Termin.

1. System speichert die neuen Änderungen.

1c. Kunde fordert Terminstornierung:

1. Mosterei-Inhaber wählt entsprechenden Termin aus und fordert Löschvorgang an.
2. System fordert Bestätigung des Löschvorgangs.
3. Mosterei-Inhaber bestätigt Löschvorgang.
4. System löscht gewählten Termin.

Special Requirements:

* Sprache: deutsch
* Lesen von ca. 1 Meter Entfernung soll möglich sein
* In 90% der Fälle soll das Programm in weniger als 10s geladen sein
* Belegte Tage sollen rot, teils belegte Tage grün, freie Tage ohne Farbe hinterlegt sein

Technology and Data Variations List:

Eine Schnittstelle mit einem Drucker sollte vorhanden sein.

Frequency of Occurrence:

Mehrmals täglich.

Open Issues:

* Was für ein Kalender ist sinnvoll für unser Programm (Monatskalender, Tageskalender bzw. mit oder ohne Uhrzeiten)
* Sind nach Auswahl eines Kunden dessen vereinbarte bzw. vergangene Termine einsehbar?
* In welcher Form werden Terminlisten angezeigt – Tagesübersicht/Wochenübersicht?
* Wo wird Jäger eingetragen?

# Use Case UC9: CRUD Mitarbeiterschicht bearbeiten

**Primary Actor:** Mosterei-Inhaber

**Stakeholder and Interests:**

* Mosterei-Inhaber: Will Mitarbeiterschichten schnell und einfach verwalten können (erstellen, ändern).
* Mitarbeiter: Will auf seinen individuellen Schichtplan schnell und einfach zugreifen können. Will eine übersichtliche Darstellung seiner persönlichen Schichten.
* Kunde: Will, dass immer ausreichend Mitarbeiter zur Verfügung stehen

**Preconditions:**

Mosterei-Inhaber ist eingeloggt/authentifiziert. Gepflegte Mitarbeiterliste liegt vor.

**Postconditions:**

Schichtplan wird angezeigt.

**Main Success Scenario:**

1. Mosterei-Inhaber fordert Anzeige des Schichtplans.
2. System präsentiert den Schichtplan.

**Extensions:**

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

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

1a. Mosterei-Inhaber fordert Erstellen eines neuen Schichtplanes an:

1. System zeigt vorgefertigte Schichtplanmaske.

2. Mosterei-Inhaber trägt Mitarbeiter in entsprechende Schichten ein.

3. Mosterei-Inhaber bestätigt Eingabe.

4. System speichert Änderungen.

1b. Mosterei-Inhaber fordert Bearbeiten eines aktuellen Schichtplanes an:

1. System zeigt aktuellen Schichtplan an.

2. Mosterei-Inhaber ändert die gewünschten Daten und bestätigt.

3. System speichert Änderungen.

1-2a. Mitarbeiter fordert Anzeige des Schichtplans:

1. System präsentiert den kompletten Schichtplan.

2a. Mitarbeiter fordert gefilterte Einzelansicht seines Schichtplanes an.

1. System präsentiert nach entsprechendem Mitarbeiter gefilterten Schichtplan.

**Special Requirements:**

* Sprache: deutsch.
* Schriftgröße von mindestens 12.
* Lesen von ca. 1 Meter Entfernung soll möglich sein.
* In 90% der Fälle soll die Übersicht in weniger als 10s geladen sein.

**Technology/Data Variations List:**

Eine Schnittstelle mit einem Drucker sollte vorhanden sein.

**Frequency of Occurrence:**

Mehrmals pro Woche.

**Open Issues:**

* Sollen Schichtpläne vergangener Arbeitstage für dauerhaft gespeichert sein?