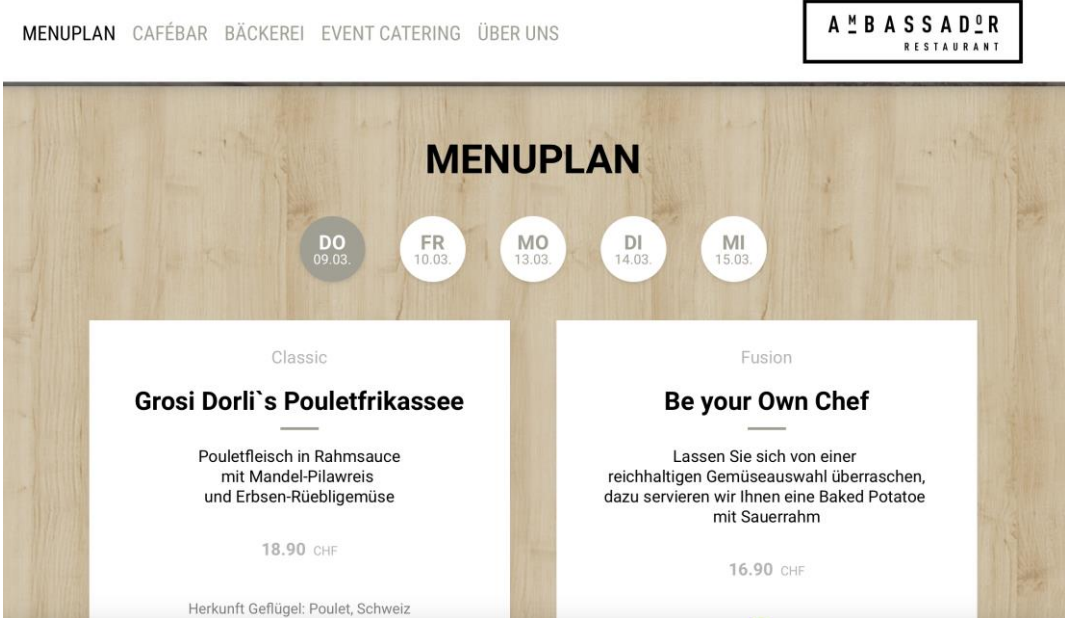


## 1 Teilnehmer/innen des Teams:

Klasse: WUP22	Team: Luca Binder, Carl Otto Strömstedt
------------------	--

## 2 Anforderungsdefinition (Meilenstein A)

Sunrise-Mensa	
<b>Fachlicher Inhalt:</b> (Allgemeine Beschreibung)	<p><b>Skript das jeden Tag des Tagesmenü der Sunrise-Mensa verschickt</b></p> <p>Die Homepage von Sunrise wird jeden Tag um 8 Uhr Angefragt.</p> <p>Homepage:</p>  <p>Die Antwort wird nach den einzelnen Menüs durchsucht. Die Menüs werden dann per Mail durch Outlook versendet.</p> <p>Das Email sollte in einzelnen Menüs strukturiert sein und so aussehen:</p>

	<p>Küche: Rice &amp; Noodles Titel: Chana Masala -----</p> <p>Indisches Curry mit Kichererbsen, Tomaten, Auberginenwürfel, Kokosmilch, dazu Basmatireis und Gurken Raita -----</p> <p>Küche: Daily Bowl Titel: Burrito Bowl -----</p> <p>Im Ofen gebackene Süsskartoffeln mit Limettenreis, Peperoni, Avocado, Kidneybohnen, Lattich und Koriander -----</p> <p>Küche: Topping 1 Titel: Topping 1 -----</p> <p>Grillierte Pouletbruststreifen mit Cayenne, Limette und Crème fraîche -----</p> <p>Küche: Topping 2 Titel: Topping 2 -----</p> <p>Geräucherte Tofuwürfel mit Paprika und Crème fraîche -----</p>
<p><b>MUSS</b> <b>Kriterien:</b> (Konkrete Features, die umzusetzen sind)</p>	<p><b>Ablauf des Scripts:</b></p> <ul style="list-style-type: none"><li>• Web-Request</li><li>• Response einlesen</li><li>• For-Loop bei dem für jedes Menü ein Text erstellt wird</li><li>• Outlook-Objekt wird definiert</li><li>• Emailangaben werden konfiguriert</li><li>• E-Mail wird versendet</li></ul>

## Dokumentation Projekt Sunrise-Mensa

<b>KANN</b> <b>Kriterien:</b> (Konkrete Features, die optional sind)	<b>Zusatz:</b> <ul style="list-style-type: none"> <li>• Einstellen welche Menüs man haben möchte</li> <li>• Email angeben an die man das Menü versendet möchte</li> </ul>
--	---

### 2.1 Planung

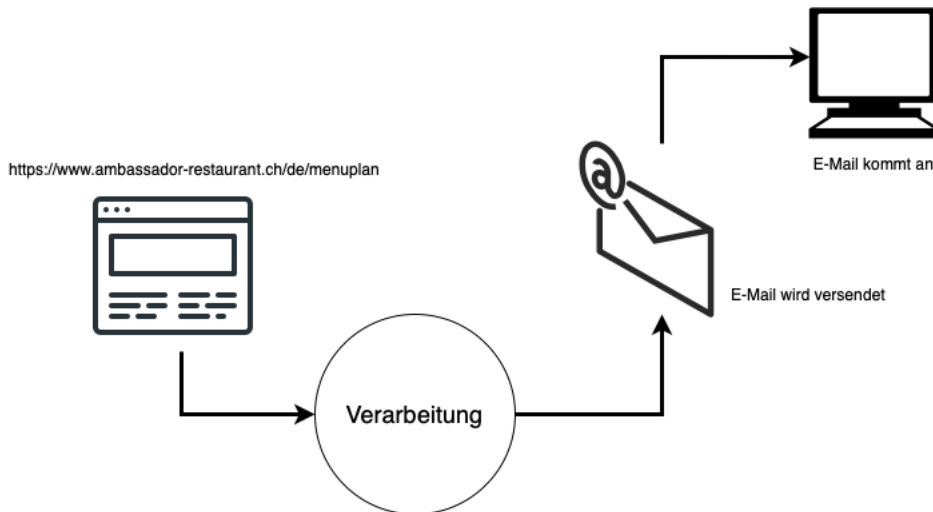
<i><b>MS</b></i>	<i><b>Tätigkeit / Abgabe</b></i>	<i><b>Soll-Datum</b></i>	<i><b>Ist-Datum</b></i>
A	<b>Projektstart</b> <ul style="list-style-type: none"> <li>➤ Team Bildung</li> <li>➤ <b>Wahl / Ausarbeitung der Anforderungsdefinition (Kap. 2)</b></li> </ul> Abnahme Anforderungsdefinition durch Lehrperson	02.03.23	
B	<b>Teamaufgabe 1:</b> <ul style="list-style-type: none"> <li>➤ <b>Abgabe: Lösungsdesign</b> (Funktionsmodell / GUI / PAP / Struktogramm / UML AD / Storyboard)</li> </ul>	09.03.23	
C	<b>Einzelaufgabe 2:</b> <ul style="list-style-type: none"> <li>➤ Abgabe Programmcode und Dokumentation</li> <li>➤ <b>Fachgespräch Projektabnahme</b></li> </ul>	30.03.23	

### 3 Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

#### 3.1 Struktur

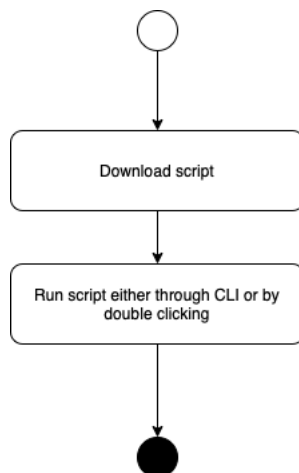
Im Folgenden sind die erwarteten Eingaben und Ausgaben beschrieben / dargestellt:



(Funktionsmodell / GUI)

#### 3.2 Ablauf

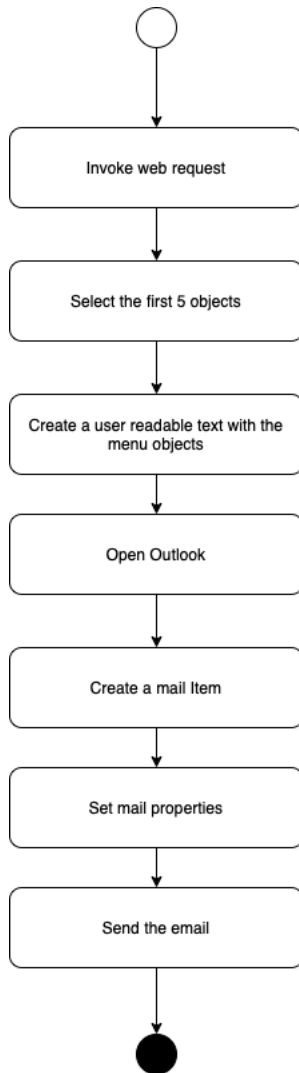
Aus Benutzersicht ist folgender Ablauf des Programms zu erwarten:



(PAP / UML AD / Storyboard)

### 3.3 Funktionen

Folgende Funktionen sind hier detailliert dokumentiert:



(Detaillierte Beschreibung der Funktionen / Struktogramm / UML AD)

## **4 Systemdokumentation (Meilenstein C: individuelle Aufgabe 3)**

Die erstellten Projekt-WPS-Scripts sind hier detailliert abgelegt:

[M122\\_Aufgabe\\_3\\_IhrName.zip](#)

### **4.1 Umfang / Abgrenzung / Änderungen gegenüber Design**

Aufgrund unten beschriebener Umstände sind Anpassungen des ursprünglichen Lösungsdesigns gemacht worden:

In unserem ursprünglichen Design haben wir die ersten 5 Objekte ausgewählt, um nur die Menüs für den Heutigen Tag anzuzeigen. Wir haben während der Implementation aber eingesehen, dass die Menüs in ihrer Anzahl von Tag zu Tag variieren und haben daher einen anderen Lösungsweg gewählt.

Die Menüs vom ersten Tag sind in einem eigenen DIV mit der ID «menu-tab-plan1» verpackt. Wir haben somit einfach nach dieser ID gefiltert und somit haben wir eine dynamischere Lösung, die immer alle Menüs für den Tag holt unabhängig von der Anzahl.

(Umstände / Anpassungen / Veränderungen)

### **4.2 Funktionalität der Implementation.**

Zusätzlich zu der Inline-Dokumentation sind hier folgende Funktionen detailliert beschrieben:

1. Ziel und Funktion des Scripts:  
Dieses PowerShell-Script ruft den aktuellen Menüplan von der Ambassador-Restaurant-Website ab und sendet diesen per E-Mail an eine definierte E-Mail-Adresse.
2. Code-Übersicht:  
Der Code besteht aus mehreren Abschnitten, die für das Abrufen des Menüs, die Formatierung des E-Mail-Texts und das Senden der E-Mail verantwortlich sind.

## 4.3 Code-Abschnitte

### 4.3.1 Variableninitialisierung

```
$email_text = ""  
$url = "https://www.ambassador-restaurant.ch/de/menuplan/"
```

Hier wird die Variable `$email_text` initialisiert, um den Text für die E-Mail zu speichern. `$url` enthält die URL der Ambassador-Restaurant-Website, von der der Menüplan abgerufen wird.

### 4.3.2 Abrufen der Webseite

```
$response = Invoke-WebRequest -Uri $url
```

Der Befehl `Invoke-WebRequest` wird verwendet, um die Webseite abzurufen und in der Variable `$response` zu speichern.

### 4.3.3 Extrahieren der Menüinformationen

```
$menu_item = ($menu_tab.getElementsByTagName('h2') | where {$_.ClassName -eq "menu-title"}).InnerText  
$menu_line = ($menu_tab.getElementsByTagName('span') | where {$_.ClassName -eq "menu-line"}).InnerText  
$menu_text = ($menu_tab.getElementsByTagName('p') | where {$_.ClassName -eq "menu-description"}).InnerText
```

In diesen Zeilen wird der Menüplan aus dem HTML der Webseite extrahiert. Dabei werden die relevanten Elemente wie Menütitel, Küchenstation und Menübeschreibung gesucht und in entsprechenden Variablen gespeichert.

### 4.3.4 Formatieren des E-Mail-Texts

```
for ($i = 0; $i -lt $menu_line.Length; $i++){  
    $email_text += @"  
    -----  
    Küche: $($menu_line[$i])  
    Titel: $($menu_item[$i])  
    -----  
    $($menu_text[$i])  
    -----  
    "@  
}
```

## Dokumentation Projekt Sunrise-Mensa

In dieser Schleife wird der E-Mail-Text formatiert, indem die extrahierten Menüinformationen in einer übersichtlichen Struktur zusammengesetzt werden.

### 4.3.5 Erstellen und Senden der E-Mail

```
$ol = New-Object -ComObject outlook.Application
# Create an email object
$mail = $ol.CreateItem(0)
# Set email properties
$mail.To = "luca.binder.privat@gmail.com"
$date = Get-Date -Format "dd/MM/yyyy"
$mail.Subject = "Menu from " + $date
$mail.Body = $email_text
$mail.Importance = 1
# send the email
$mail.Send()
```

In diesem Abschnitt wird ein neues Outlook-Objekt erstellt und ein E-Mail-Objekt mit den formatierten Menüinformationen als Inhalt erzeugt. Die E-Mail wird an die definierte E-Mail-Adresse gesendet, und der Betreff enthält das aktuelle Datum.

## 4.4 Anforderungen und Voraussetzungen

### 4.4.1 Betriebssystem und PowerShell-Version

Dieses Script wurde für die Ausführung in einer Windows-Umgebung entwickelt und erfordert eine PowerShell-Version 5.1 oder höher.

### 4.4.2 Outlook-Installation

Das Script verwendet die Outlook-Desktopanwendung, um die E-Mail zu erstellen und zu senden. Daher ist es erforderlich, dass die Outlook-Anwendung auf dem Computer installiert ist und korrekt eingerichtet wurde.



## 4.5 Anpassungen und Erweiterungen

### 4.5.1 Ändern der E-Mail-Adresse

Um die E-Mail an eine andere Adresse zu senden, ändern Sie den Wert der Variable `$mail.To`:

```
$mail.To = "beispiel@email.com"
```

### 4.5.2 Anpassen der E-Mail-Betreffzeile

Um den Betreff der E-Mail anzupassen, ändern Sie den Wert der Variable `$mail.Subject`:

```
$mail.Subject = "Neuer Betreff: " + $date
```

### 4.5.3 Erweitern des Scripts um weitere Funktionen

Das Script kann nach Bedarf angepasst oder erweitert werden, z. B. durch das Hinzufügen von:

- Cc- oder Bcc-Empfängern: Fügen Sie die Zeilen `$mail.CC = "beispiel_cc@email.com"` und/oder `$mail.BCC = "beispiel_bcc@email.com"` ein.
- E-Mail-Anhänge: Fügen Sie die Zeile `$mail.Attachments.Add("Pfad/zur/Datei.ext")` ein.
- HTML-E-Mail-Formatierung: Ändern Sie den Wert der Variable `$mail.BodyFormat` auf `2` (für HTML) und passen Sie den Inhalt der Variable `$email_text` entsprechend an.

#### **4.6 Zusammenfassung**

Dieses PowerShell-Script dient dazu, den Menüplan des Ambassador-Restaurants von deren Webseite abzurufen und per E-Mail an eine definierte E-Mail-Adresse zu senden. Es nutzt die Outlook-Anwendung, um die E-Mail zu erstellen und zu versenden, und kann nach Bedarf angepasst oder erweitert werden.

(Ausführliche Beschreibung der internen Funktionen  
oder Verweis zum Inline-Kommentar)

### **5 Betriebsdokumentation (Meilenstein C: individuelle Aufgabe 3)**

#### **5.1 Installations- / Bedienungsanleitung**

Es werden 3 files benötigt: "schedule\_email.ps1", "schedule\_open\_outlook.ps1" and "send\_email.ps1"

Zuerst müssen Sie "schedule\_email.ps1" in einem Text editor öffnen und auf Zeile 25 die Email-Adresse zu ihrer gewünschten Email-Adresse ändern.

```
24 # Set email properties
25 $mail.To = "luca.binder.privat@gmail.com"
```

Nach diesem Schritt lassen Sie die Skripts "schedule\_email.ps1" und "schedule\_open\_outlook.ps1" in Powershell laufen.

Jetzt wird jeden Morgen um 8:00 an Wochentagen ein Mail mit dem Restaurant-Menü versendet.

#### **5.2 Fehlersuche und Fehlerbehebung**

Sollten beim Ausführen des Scripts Fehler auftreten, überprüfen Sie bitte:

- Die Verbindung zum Internet
- Die korrekte Installation und Konfiguration von Outlook

## **Dokumentation Projekt Sunrise-Mensa**

- Die Verfügbarkeit und Struktur der Ambassador-Restaurant-Website, da das Script auf der aktuellen Struktur basiert und bei Änderungen möglicherweise angepasst werden muss