

Erstellung eines Arztbriefes mit Hilfe von Chat GPT

Ausgewählte Softwaresysteme - Programmierung IV (ASS-PR4) Sommersemester 2023

An der Fachhochschule Dortmund

Betreuer: Prof. Burkhard Igel, Dr. Georg Pietrek

vorgelegt von: Tim Friedemann (7202004) und Maximilian Arntzen (7202005)

Abgabetermin: 15.07.2023

Inhalt

1. Einleitung	3
2. Motivation	3
3.1 Erster Ansatz (Chatbot)	4
3.2 Code anpassen und Erscheinungsbild verbessern	10
4. Zweiter Ansatz (OpenAi API)	13
4.1 Vorbereitung	13
4.1 Code Generierung und Problembehandlung	14
5. Fazit	33

1. Einleitung

Im Rahmen unseres Moduls „Ausgewählte Softwaresysteme - Programmierung IV“ haben wir uns mit dem Thema Chat GPT und Künstliche Intelligenz beschäftigt. Wir wollten unter anderem klären, wie wir Chat GPT für unsere Zwecke geschickt nutzen können. Zudem fragten wir uns, ob wir die Ausgaben (Code für Programme oder Wissenschaftliche Aussagen) von Chat GPT überhaupt nachvollziehen können. Darauf aufbauend stellte sich die Frage, wo die Grenzen des Chat Bots liegen. Auf diese Fragestellungen werden wir in unserem Bericht eingehen und diese in unserem Fazit erneut aufgreifen.

Im Rahmen der obigen Fragestellungen haben wir uns mit dem Thema „Erstellung eines Arztbriefes in VBA, mit Hilfe von ChatGPT“ beschäftigt.

Wir haben Chat GPT genutzt, um uns VBA näher zu bringen und aufbauend darauf einen VBA-Code generieren zu lassen. Das Ziel war die automatisierte Ausgabe eines Arztbriefes. Umgesetzt haben wir das Projekt in dem wir zwei Ansätze verfolgten. Der erste Ansatz war die Erstellung des Arztbriefes unter ausschließlicher Nutzung des Chat Bots.

Unser zweiter Ansatz verfolgt ein deutlich komplexeres Thema, nämlich die Benutzung des Chat Bots mit der von OpenAI bereitgestellten API (Application Programming Interface).

2. Motivation

Die Motivation, Chat GPT als Textsystem im Krankenhaus einzusetzen, basiert auf mehreren Vorteilen. Grundlegend kann es eine schnellere und präzisere Generierung von Arztbriefen ermöglichen. Dies kann zu einem effizienteren Informationsaustausch zwischen Ärzten, Patienten und medizinischen Fachkräften führen. Eine große Rolle spielt auch der deutlich niedrige Arbeitsaufwand der Ärzte und des medizinischen Personals, dies kann zu einer erhöhten Effizienz führen. Diese erhöhte Effizienz kann sich bei den Patienten bemerkbar machen, in dem Die Ärzte und medizinisches Personal mehr Zeit haben, um Patienten zu betreuen. Somit könnten die Ärzte und Patienten, sowie das ganze Krankenhaus entlastet werden. Der Wirtschaftliche Aspekt spielt auch eine Rolle. Der Einsatz von Chat GPT als Textsystem im Krankenhaus bietet auch eine Gewisse Kosten Ersparnis, da das Krankenhaus kein Geld verdient in dem Moment in dem der Arzt einen Arztbrief schreib. Das verdiente Geld misst sich Anhand der behandelten Patienten, von denen nun mehr behandelt werden könnten. Letztendlich bietet das Erstellen automatisierter Arztbriefe auch eine geringe Fehleranfälligkeit im Vergleich zur manuellen Erstellung. Das sind einige der Hauptgründe warum Chat GPT als Textsystem im Krankenhaus als vielversprechende Lösung betrachtet werden kann.

3.1 Erster Ansatz (Chatbot)

Bevor wir uns ein VBA-Makro von Chat GPT generieren lassen, haben wir eine in Excel beispielhafte Tabelle erstellt, die in diesem Projekt als Datenbank gilt. Diese Datenbank ist nötig damit das Makro Daten hat, auf das es zugreifen kann, um den Arztbrief zu erstellen. Darin befinden sich in diesem Fall Patientendaten, wie zum Beispiel Symptome, Diagnosen und andere Persönliche Informationen.

	A	B	C	D	E	F	G	H
1	PatientenID	Name	Geburtsdatum	Wohnort	Symptome	Diagnose	Medikamente	Arzt Name
2	1	Max Arntzen	30.10.1998	Ahlen	Kopfschmerzen	Nackenspannung	keine	Dr. Mustermann
3	2	Tim Friedemann	27.04.1999	Ahlen	Halsschmerzen	Bronchitis	Cystinol akut	Dr. Mustermann

Abbildung 1 Screenshot Exceltabelle / Datenbank

Nachdem wir die Datenbank erstellt haben, stellten wir ChatGPT die erste Frage:

„Erstelle ein Excel Makro, welches auf eine Excel Tabelle zugreift, nach einem Namen fragt nach dem gesucht werden soll und mit den vorhandenen Daten einen Arztbrief erstellt. Dieser Arztbrief soll dann in einem Word Dokument gespeichert werden.“

Chat GPT:

Natürlich! Hier ist ein Beispiel für ein Excel-Makro, das auf eine Excel-Tabelle zugreift, nach einem Namen fragt und mit den vorhandenen Daten einen Arztbrief erstellt. Der Arztbrief wird dann in einem Word-Dokument gespeichert:

```
vba
Copy code
Sub ArztbriefErstellenUndSpeichern()
' Variablendeklarationen
Dim patientName As Variant
Dim patientDataSheet As Worksheet
Dim rowNr As Variant
Dim strBrief As String
' Definieren Sie das Arbeitsblatt mit den Patientendaten
Set patientDataSheet = ThisWorkbook.Sheets("Patientendaten")
' Holen Sie den Namen des Patienten von einem Userformular oder einer Inputbox ab
patientName = InputBox("Bitte geben Sie den Namen des Patienten ein:")
' Suchen Sie nach dem Namen des Patienten in der Spalte "A" des Arbeitsblatts mit den
Patientendaten
rowNr = Application.Match(patientName, patientDataSheet.Range("A:A"), 0)
' Überprüfen Sie, ob der Name des Patienten gefunden wurde
If Not IsError(rowNr) Then
```

```

' Wenn der Name des Patienten gefunden wurde, geben Sie die Daten für diesen Patienten
aus
strBrief = "Sehr geehrte Kolleginnen und Kollegen," & vbCrLf & vbCrLf _
& "hiermit möchten wir Ihnen den Fall des Patienten " & patientDataSheet.Cells(rowNr, 1) &
" vorstellen." _
& vbCrLf & vbCrLf & "Der Patient wurde am " & patientDataSheet.Cells(rowNr, 2) & "
geboren und wohnt in " _
& patientDataSheet.Cells(rowNr, 3) & "." & vbCrLf & vbCrLf & "In den vergangenen Wochen
wurde er/sie von uns behandelt wegen " _
& patientDataSheet.Cells(rowNr, 4) & "." & vbCrLf & "Die medikamentöse Therapie besteht
derzeit aus " _
& patientDataSheet.Cells(rowNr, 5) & "." & vbCrLf & vbCrLf & "Wir werden den Patienten in
den nächsten Wochen weiterhin betreuen und halten Sie gerne auf dem Laufenden." _
& vbCrLf & vbCrLf & "Mit freundlichen Grüßen," & vbCrLf & "" _
& patientDataSheet.Cells(rowNr, 6) & "."
' Word öffnen und neuen Dokument erstellen
Dim wdApp As Object
Dim wdDoc As Object
Set wdApp = CreateObject("Word.Application")
wdApp.Visible = True
Set wdDoc = wdApp.Documents.Add
' Text in Word-Dokument einfügen
wdDoc.Content.InsertAfter strBrief
' Dialog zum Speichern des Word-Dokuments anzeigen
Dim savePath As String
savePath = Application.GetSaveAsFilename(fileFilter:="Word-Dokument (*.docx), *.docx")
' Überprüfen Sie, ob der Benutzer einen Speicherort ausgewählt hat
If savePath <> "False" Then
' Dokument speichern
wdDoc.SaveAs2 savePath
wdDoc.Close SaveChanges:=False
Else
' Dokument nicht speichern und schließen
wdDoc.Close SaveChanges:=False
End If
' Word beenden
wdApp.Quit
Set wdDoc = Nothing
Set wdApp = Nothing
Else
' Wenn der Name des Patienten nicht gefunden wurde, geben Sie eine Fehlermeldung aus
MsgBox "Der angegebene Patient wurde nicht gefunden."
End If
End Sub

```

In diesem Beispiel gehe ich davon aus, dass die Patientendaten in einem Arbeitsblatt mit dem Namen "Patientendaten" und den Spalten A bis F vorliegen. Sie können den Code entsprechend anpassen, um den tatsächlichen Namen des Arbeitsblatts und die Spalten Ihrer Patientendaten zu berücksichtigen.

Um diesen code auszuführen, öffnet man Excel und klickt oben in der Liste auf den Button „Ansicht“ und dann auf Makro. Danach gibt man einen Namen ein und klickt auf Makro erstellen. Damit hat man ein neues Modul erstellt, in welchem jetzt der von Chat GPT generierte Code eingefügt wird. Daraufhin gibt man den Namen der Excel Tabelle an und führt den Code aus.

Es öffnet sich dann ein Fenster, in dem man den Namen einer Person eingeben kann, für die ein Arztbrief erstellt werden soll. Falls der Name nicht in der Tabelle steht, erscheint die Meldung „Der angegebene Patient wurde nicht gefunden“.

Dabei ist uns aufgefallen das im Code von ChatGPT, in der Spalte A nach dem Namen gesucht wird, in unserer Datenbank sind die Namen jedoch in Spalte B eingetragen. Wir baten daraufhin ChatGPT das zu ändern.

„Kannst du den Code so anpassen, das in Spalte B nach dem Namen gesucht wird“

Chat GPT:

Natürlich! Hier ist der angepasste Code, um in Spalte B nach dem Namen zu suchen:

```
vba
Copy code
Sub ArztbriefErstellenUndSpeichern()
' Variablendeklarationen
Dim patientName As Variant
Dim patientDataSheet As Worksheet
Dim rowNr As Variant
Dim strBrief As String
' Definieren Sie das Arbeitsblatt mit den Patientendaten
Set patientDataSheet = ThisWorkbook.Sheets("Patientendaten")
' Holen Sie den Namen des Patienten von einem Userformular oder einer Inputbox ab
patientName = InputBox("Bitte geben Sie den Namen des Patienten ein:")
' Suchen Sie nach dem Namen des Patienten in der Spalte "B" des Arbeitsblatts mit den
Patientendaten
rowNr = Application.Match(patientName, patientDataSheet.Range("B:B"), 0)
' Überprüfen Sie, ob der Name des Patienten gefunden wurde
If Not IsError(rowNr) Then
' Wenn der Name des Patienten gefunden wurde, geben Sie die Daten für diesen Patienten
aus
strBrief = "Sehr geehrte Kolleginnen und Kollegen," & vbCrLf & vbCrLf _
& "hiermit möchten wir Ihnen den Fall des Patienten " & patientDataSheet.Cells(rowNr, 2) &
" vorstellen." _
& vbCrLf & vbCrLf & "Der Patient wurde am " & patientDataSheet.Cells(rowNr, 3) & "
geboren und wohnt in " _
& patientDataSheet.Cells(rowNr, 4) & "." & vbCrLf & vbCrLf & "In den vergangenen Wochen
wurde er/sie von uns behandelt wegen " _
& patientDataSheet.Cells(rowNr, 5) & "." & vbCrLf & "Die medikamentöse Therapie besteht
derzeit aus " _
& patientDataSheet.Cells(rowNr, 6) & "." & vbCrLf & vbCrLf & "Wir werden den Patienten in
den nächsten Wochen weiterhin betreuen und halten Sie gerne auf dem Laufenden." _
& vbCrLf & vbCrLf & "Mit freundlichen Grüßen," & vbCrLf & "" _
& patientDataSheet.Cells(rowNr, 7) & "."
' Word öffnen und neuen Dokument erstellen
Dim wdApp As Object
```

```

Dim wdDoc As Object
Set wdApp = CreateObject("Word.Application")
wdApp.Visible = True
Set wdDoc = wdApp.Documents.Add
' Text in Word-Dokument einfügen
wdDoc.Content.InsertAfter strBrief
' Dialog zum Speichern des Word-Dokuments anzeigen
Dim savePath As String
savePath = Application.GetSaveAsFilename(fileFilter:="Word-Dokument (*.docx), *.docx")
' Überprüfen Sie, ob der Benutzer einen Speicherort ausgewählt hat
If savePath <> "False" Then
' Dokument speichern
wdDoc.SaveAs2 savePath
wdDoc.Close SaveChanges:=False
Else
' Dokument nicht speichern und schließen
wdDoc.Close SaveChanges:=False
End If
' Word beenden
wdApp.Quit
Set wdDoc = Nothing
Set wdApp = Nothing
Else
' Wenn der Name des Patienten nicht gefunden wurde, geben Sie eine Fehlermeldung aus
MsgBox "Der angegebene Patient wurde nicht gefunden."
End If
End Sub

```

In diesem angepassten Code wird nach dem Namen in Spalte B des Arbeitsblatts "Patientendaten" gesucht. Sie können den Code entsprechend anpassen, um den tatsächlichen Namen des Arbeitsblatts und die Spalte Ihrer Patientendaten zu berücksichtigen.

Dieser Code funktioniert bereits, liefert aber noch kein akkurates Ergebnis da im Grün markierten Teil (S.6), die Zuordnung der Informationen nicht mit unserer Tabelle übereinstimmt. Außerdem haben wir bevor wir den Code getestet haben, die Tabelle etwas angepasst.

	A	B	C	D	E	F	G	H
1	PatientenID	Name	Gebursdatum	Wohnort	Symptome	Diagnose	Medikamente	Artzt Name
2	1	Max Arntzen	30.10.1998	Ahlen	Schwindel, Übelkeit und Kurzatmigkeit bei Belastung	Bluthochdruck	Betablocker	Dr. Mustermann
3	2	Tim Friedemann	27.04.1999	Ahlen	Halsschmerzen	Bronchitis	Cystinol akut	Dr. Mustermann

Abbildung 2 Screenshot angepasste Exceltabelle / Datenbank

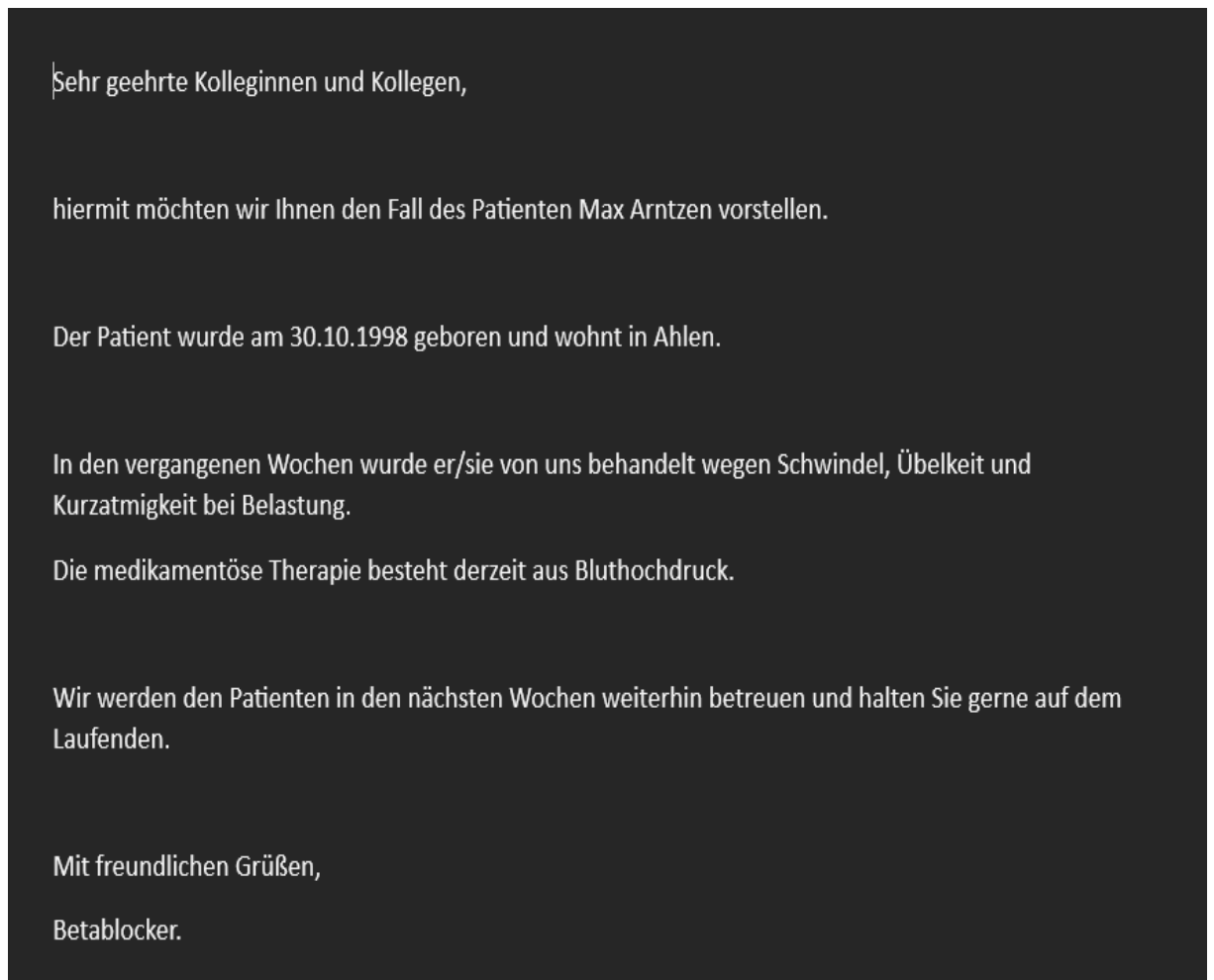


Abbildung 3 Screenshot Fehlerhafte Datenzuordnung / Erster Ansatz

Um dieses Problem zu beheben hat man die Möglichkeit entweder die Tabelle oder den Code anzupassen. Wir entschieden uns für den Code. Die Zuordnung der Daten haben wir korrigiert, indem wir die rot markierte 6 und 7 zu 7 und 8 geändert haben. Des Weiteren ist uns aufgefallen, dass die Diagnose aus unserer Tabelle nicht verwendet wird. Dazu haben wir die folgende Zeile hinzugefügt:

```
"Die Diagnose lautet " _ & patientDataSheet.Cells(rowNr, 6) & "
```


Der Code funktioniert jetzt wie erwartet und das Ergebnis sieht wie folgt aus:

Sehr geehrte Kolleginnen und Kollegen,

hiermit möchten wir Ihnen den Fall des Patienten Max Arntzen vorstellen.

Der Patient wurde am 30.10.1998 geboren und wohnt in Ahlen.

In den vergangenen Wochen wurde er/sie von uns behandelt wegen Schwindel, Übelkeit und Kurzatmigkeit bei Belastung.

Die Diagnose lautet Bluthochdruck und die medikamentöse Therapie besteht derzeit aus Betablocker.

Wir werden den Patienten in den nächsten Wochen weiterhin betreuen und halten Sie gerne auf dem Laufenden.

Mit freundlichen Grüßen,

Dr. Mustermann.

Abbildung 4 Screenshot Ergebnis / Erster Ansatz

3.2 Code anpassen und Erscheinungsbild verbessern

Da wir noch nicht ganz mit dem Erscheinungsbild des Arztbriefes fertig waren, versuchten wir mit Hilfe von Chat GPT das erzeugte Word Dokument, wie einen Brief aussehen zu lassen.

„kannst du aber den Teil, indem der Text geschrieben ist, so erweitern das es im Word Dokument wie ein normaler Brief mit Adresszeile und so weitergeschrieben ist. Die Daten, die dazu fehlen kannst du dir einfach ausdenken.“

Chat GPT:

Natürlich kann ich den Code erweitern, um den Text im Word-Dokument wie einen normalen Brief mit einer Adresszeile und weiteren Informationen zu gestalten. Hier ist eine aktualisierte Version des Codes:

```
vba
Copy code
Sub ArztbriefErstellenUndSpeichern()
' Variablendeklarationen
Dim patientName As Variant
Dim patientDataSheet As Variant
Dim wb As Variant
Dim rowNr As Variant
Dim strBrief As String

' Öffnen Sie die Arbeitsmappe mit den Patientendaten
Set wb = Workbooks.Open("X:\Downloads SSD X\Datenbeste.xlsx")

' Definieren Sie das Arbeitsblatt mit den Patientendaten
Set patientDataSheet = wb.Worksheets("1234")

' Holen Sie den Namen des Patienten von einem Userformular oder einer Inputbox ab
patientName = InputBox("Bitte geben Sie den Namen des Patienten ein:")
' Suchen Sie nach dem Namen des Patienten in der Spalte "B" des Arbeitsblatts mit den
Patientendaten
rowNr = Application.Match(patientName, patientDataSheet.Range("B:B"), 0)
' Überprüfen Sie, ob der Name des Patienten gefunden wurde
If Not IsError(rowNr) Then
' Wenn der Name des Patienten gefunden wurde, geben Sie die Daten für diesen Patienten
aus
strBrief = "Sehr geehrte Kolleginnen und Kollegen," & vbCrLf & vbCrLf _
& "hiermit möchten wir Ihnen den Fall des Patienten " & patientDataSheet.Cells(rowNr, 2) &
" vorstellen." _
& vbCrLf & vbCrLf & "Der Patient wurde am " & patientDataSheet.Cells(rowNr, 3) & "
geboren und wohnt in " _
& patientDataSheet.Cells(rowNr, 4) & "." & vbCrLf & vbCrLf & "In den vergangenen Wochen
wurde er/sie von uns behandelt wegen " _
& patientDataSheet.Cells(rowNr, 5) & "." & vbCrLf & "Die medikamentöse Therapie besteht
derzeit aus " _
```

```
& patientDataSheet.Cells(rowNr, 7) & "." & vbCrLf & vbCrLf & "Wir werden den Patienten in  
den nächsten Wochen weiterhin betreuen und halten Sie gerne auf dem Laufenden." _  
& vbCrLf & vbCrLf & "Mit freundlichen Grüßen," & vbCrLf & "" _  
& patientDataSheet.Cells(rowNr, 8) & "."
```

```
' Word öffnen und neuen Dokument erstellen
```

```
Dim wdApp As Object
```

```
Dim wdDoc As Object
```

```
Set wdApp = CreateObject("Word.Application")
```

```
wdApp.Visible = True
```

```
Set wdDoc = wdApp.Documents.Add
```

```
' Text in Word-Dokument einfügen
```

```
With wdDoc
```

```
' Adresse des Absenders
```

```
.Content.InsertAfter "Ihre Praxis" & vbCrLf
```

```
.Content.InsertAfter "Musterstraße 123" & vbCrLf
```

```
.Content.InsertAfter "12345 Musterstadt" & vbCrLf & vbCrLf
```

```
' Datum
```

```
.Content.InsertAfter "Musterstadt, " & Format(Date, "dd.mm.yyyy") & vbCrLf & vbCrLf
```

```
' Leerzeile
```

```
.Content.InsertAfter vbCrLf
```

```
' Adresse des Empfängers (angenommen)
```

```
.Content.InsertAfter "Dr. med. Max Mustermann" & vbCrLf
```

```
.Content.InsertAfter "Musterklinik" & vbCrLf
```

```
.Content.InsertAfter "Krankenhausstraße 456" & vbCrLf
```

```
.Content.InsertAfter "543 Musterort" & vbCrLf & vbCrLf
```

```
' Leerzeile
```

```
.Content.InsertAfter vbCrLf
```

```
' Leerzeile
```

```
.Content.InsertAfter vbCrLf
```

```
' Leerzeile
```

```
.Content.InsertAfter vbCrLf
```

```
' Briefinhalt
```

```
.Content.InsertAfter strBrief
```

```
' Leerzeile
```

```
.Content.InsertAfter vbCrLf
```

```
' Schließen Sie das Word-Dokument
```

```
.Close SaveChanges:=True
```

```
End With
```

```
Else
```

```
' Wenn der Name des Patienten nicht gefunden wurde, geben Sie eine Fehlermeldung aus  
MsgBox "Der angegebene Patient wurde nicht gefunden."
```

```
End If
```

```
End Sub
```

Uns ist aufgefallen das dieser Code an manchen Stellen etwas abgeändert wurde, trotzdem funktioniert er und liefert folgendes Ergebnis:

Ihre Praxis

Musterstraße 123

12345 Musterstadt

Musterstadt, 15.07.2023

Dr. med. Max Mustermann

Musterklinik

Krankenhausstraße 456

543 Musterort

Sehr geehrte Kolleginnen und Kollegen,

hiermit möchten wir Ihnen den Fall des Patienten Max Arntzen vorstellen.

Der Patient wurde am 30.10.1998 geboren und wohnt in Ahlen.

In den vergangenen Wochen wurde er/sie von uns behandelt wegen Schwindel, Übelkeit und Kurzatmigkeit bei Belastung.

Die Diagnose lautet Bluthochdruck und die medikamentöse Therapie besteht derzeit aus Betablocker.

Wir werden den Patienten in den nächsten Wochen weiterhin betreuen und halten Sie gerne auf dem Laufenden.

Mit freundlichen Grüßen,

Dr. Mustermann.

4. Zweiter Ansatz (OpenAi API)

4.1 Vorbereitung

Um uns über die Funktion und Nutzung der API zu informieren haben wir Chat GPT nach einer Erklärung gefragt.

Chat GPT:

„Die OpenAI API ist eine Programmierschnittstelle, die von OpenAI bereitgestellt wird, um Entwicklern und Unternehmen Zugang zu den neuesten Entwicklungen im Bereich der künstlichen Intelligenz zu ermöglichen. Mit der API können Entwickler auf die leistungsstarken KI-Modelle von OpenAI zugreifen, um eine Vielzahl von Anwendungen zu erstellen, einschließlich Spracherkennung, Textanalyse, maschinelles Lernen und mehr.

Die OpenAI API nutzt die neuesten Fortschritte in der tiefen neuronalen Netzwerktechnologie, einschließlich der GPT-3-Technologie, die es ermöglicht, Sprache auf eine Weise zu verstehen und zu generieren, die der menschlichen Sprache ähnelt. Mit der OpenAI API können Entwickler hochwertige Anwendungen erstellen, die natürliche Sprache verstehen, interpretieren und generieren können.

Die OpenAI API ist eine cloud-basierte Plattform, die über eine RESTful-API-Schnittstelle verfügbar ist. Sie bietet Entwicklern eine einfache Möglichkeit, KI-Funktionen in ihre Anwendungen zu integrieren, ohne dass sie selbst KI-Modelle trainieren oder eine umfangreiche Infrastruktur aufbauen müssen.“

Zudem haben wir nach einer Anleitung zur Nutzung der OpenAI API mit Hilfe von Excel Makros gefragt.

Chat GPT:

„Ja, es ist möglich, die OpenAI API mit einem Excel-Makro in VBA zu nutzen. Dazu müssen Sie die API-Anfrage mit VBA-Code erstellen und die Antwort verarbeiten. Hier sind einige Schritte, die Sie befolgen können:

1. Stellen Sie sicher, dass Sie ein OpenAI-API-Konto haben und Ihre API-Schlüssel generiert haben.
2. Fügen Sie eine Referenz zu Microsoft WinHTTP Services 5.1 hinzu, um HTTP-Anforderungen in VBA zu senden.
3. Erstellen Sie eine HTTP-Anforderung, um die OpenAI-API mit den erforderlichen Anforderungsparametern aufzurufen. Die genauen Parameter hängen von der Funktion ab, die Sie verwenden möchten. Sie können die OpenAI-API-Dokumentation konsultieren, um mehr darüber zu erfahren, welche Parameter erforderlich sind.

4. Fügen Sie Ihren API-Schlüssel in der Anforderung als Header hinzu, um Ihre API-Anfragen zu authentifizieren.

5. Verarbeiten Sie die Antwort von OpenAI-API in Ihrem VBA-Makro. Dies kann entweder das Parsen von JSON oder das Lesen von Text umfassen, je nachdem, was von der API zurückgegeben wird.

Bitte beachten Sie, dass es auch andere Möglichkeiten gibt, die OpenAI-API zu nutzen, wie z.B. die Verwendung von Python oder anderen Programmiersprachen.“

4.1 Code Generierung und Problembehandlung

Bevor wir die erste Anfrage an Chat GPT gesendet haben, überlegten wir uns welche Funktionen der Code benötigt, um unser Ziel zu erreichen. Demnach war unser erster Punkt, dass dieses Makro auch zuerst nach einem Namen fragt, in dem es nach der Eingabe in einer Excel Tabelle sucht. Die zweite Funktion sollte sein, die Daten, die in dieser Zeile vorhanden sind, an die OpenAI API zu übergeben, um damit einen Arztbrief zu generieren. Zum Schluss soll dann der generierte Arztbrief kopiert und in ein Word Dokument eingefügt werden.

„Schreibe mir einen VBA code der folgenden Funktionen hat: 1. Nach dem Ausführen wird nach einem Namen gefragt, nachdem danach in einer Excel Tabelle gesucht wird. 2. Alle Daten, die in dieser Zeile gefunden wurden, werden dann an die OpenAI API übergeben. 3. Der von Chat GPT generierte Text soll dann in ein Word Dokument eingefügt werden“

Chat GPT:

„Hier ist ein Beispielcode, der diese Funktionen ausführt:

VBA

Copy code

Sub openAI_API()

Dim name As String

Dim url As String

Dim http As New WinHttpRequest

Dim response As String

Dim obj As Object

Dim text As String

Dim wordApp As Object

Dim wordDoc As Object

'Fragt nach dem Namen, der in der Tabelle gesucht werden soll

```

name = InputBox("Bitte geben Sie den Namen ein, nach dem Sie suchen möchten:")
'Sucht in der Excel-Tabelle nach dem Namen
Set obj = Worksheets("Sheet1").UsedRange.Find(name)
'Wenn der Name gefunden wird, werden die Daten in der Zeile an die OpenAI-API übergeben
If Not obj Is Nothing Then
url = "https://api.openai.com/v1/engines/davinci-codex/completions"
http.Open "POST", url, False
http.setRequestHeader "Content-Type", "application/json"
http.setRequestHeader "Authorization", "Bearer API_KEY" 'ersetzen Sie API_KEY durch Ihren
eigenen API-Schlüssel
http.Send "{"prompt"":"" & obj.EntireRow.Cells(1).Value &
""", ""max_tokens"":100, ""n"":1, ""stop"":"".""}"
response = http.responseText
Set json = JsonConverter.ParseJson(response)
text = json("choices")(1)("text")
'Kopiert den generierten Text in die Zwischenablage
With New MSForms.DataObject
.SetText text
.PutInClipboard
End With
'Fügt den generierten Text in ein Word-Dokument ein
Set wordApp = CreateObject("Word.Application")
Set wordDoc = wordApp.Documents.Add
wordApp.Selection.Paste
'Zeigt das Word-Dokument an
wordApp.Visible = True
Else

MsgBox "Der Name wurde nicht gefunden."

End If

End Sub

```

Bitte beachten Sie, dass Sie Ihren eigenen API-Schlüssel in den Code einfügen müssen, bevor Sie ihn ausführen. Sie müssen auch sicherstellen, dass Sie die "Microsoft WinHTTP Services 5.1" Referenz hinzugefügt haben, damit der Code funktioniert.“

Nachdem wir uns diesen code angeschaut haben, ist uns jedoch aufgefallen, dass die Suche nach dem Namen in der Excel Tabelle nicht zu unserer Vorstellung passt. Das lag aber auch daran, dass wir unsere Anfrage nicht präzise genug gewählt haben. Daher haben wir daraufhin die folgende Anfrage geschickt:

„Passe den Code so an, dass er in der Spalte B nach dem Namen in der Tabelle sucht“

Chat GPT:

Klar, hier ist der angepasste Code, der in der Spalte B nach dem Namen sucht:

```
VBA
Copy code
Sub openAI_API()

Dim name As String
Dim url As String
Dim http As New WinHttpRequest
Dim response As String
Dim obj As Object
Dim text As String
Dim wordApp As Object
Dim wordDoc As Object

'Fragt nach dem Namen, der in der Tabelle gesucht werden soll
name = InputBox("Bitte geben Sie den Namen ein, nach dem Sie suchen möchten:")
'Sucht in der Excel-Tabelle nach dem Namen in Spalte B
Set obj = Worksheets("Sheet1").Range("B:B").Find(name)
'Wenn der Name gefunden wird, werden die Daten in der Zeile an die OpenAI-API übergeben
If Not obj Is Nothing Then
url = "https://api.openai.com/v1/engines/davinci-codex/completions"
http.Open "POST", url, False
http.setRequestHeader "Content-Type", "application/json"
http.setRequestHeader "Authorization", "Bearer API_KEY" 'ersetzen Sie API_KEY durch Ihren
eigenen API-Schlüssel
http.Send "{"prompt":"",""" & obj.EntireRow.Cells(1).Value &
""", ""max_tokens"":100,""n"":1,""stop"":"".""}"

response = http.responseText

Set json = JsonConverter.ParseJson(response)

text = json("choices")(1)("text")

'Kopiert den generierten Text in die Zwischenablage

With New MSForms.DataObject

.SetText text

.PutInClipboard

End With

'Fügt den generierten Text in ein Word-Dokument ein

Set wordApp = CreateObject("Word.Application")
```



```
Set wordDoc = wordApp.Documents.Add

wordApp.Selection.Paste
'Zeigt das Word-Dokument an
wordApp.Visible = True
Else
MsgBox "Der Name wurde nicht gefunden."
End If

End Sub
```

In diesem Code wird die Range-Methode verwendet, um die Suche auf die gesamte Spalte B zu erweitern. Der Rest des Codes bleibt gleich.

Indem wir ein neues Modul erstellt, den Namen unserer Excel Tabelle sowie den API-Key eingetragen haben, konnten wir den Code das erste Mal testen. Dies hat leider nicht auf Anhieb funktioniert und wir haben die Fehlermeldung „Benutzerdefinierter Typ nicht definiert“ mit der Zeile „With New MSForms.DataObject“ im Code markiert bekommen. Den Ausgegebenen Fehler und die markierte Zeile haben wir Chat GPT übermittelt und gefragt, was an dieser Stelle das Problem ist.

Chat GPT:

„Es scheint, dass der Compiler eine Referenz auf das Objekt "Microsoft Forms 2.0 Object Library" benötigt, um den benutzerdefinierten Datentyp "MSForms.DataObject" zu erkennen.“

Um die Referenz hinzuzufügen, gehen Sie wie folgt vor:

1. Öffnen Sie die Visual Basic Editor durch Drücken von Alt + F11 in Excel.
2. Klicken Sie im Menü "Extras" auf "Verweise".
3. Scrollen Sie nach unten und aktivieren Sie das Kontrollkästchen neben "Microsoft Forms 2.0 Object Library".
4. Klicken Sie auf "OK", um die Änderungen zu speichern und den Verweis hinzuzufügen.

Sobald Sie den Verweis hinzugefügt haben, sollte der Code ohne Probleme ausgeführt werden.“

Wir sind diesen Schritten gefolgt, haben jedoch dabei festgestellt, dass die Microsoft Forms 2.0 Object Library nicht in unserer Liste vorhanden ist. Auch das haben wir an Chat GPT übergeben und als Antwort darauf kann eine weitere Anleitung:

Chat GPT:

In diesem Fall kann es sein, dass Ihre Excel-Version diese Bibliothek nicht unterstützt oder dass sie nicht auf Ihrem Computer installiert ist.

Eine alternative Möglichkeit besteht darin, den generierten Text in eine Variable zu speichern und dann manuell in das Word-Dokument einzufügen. Hier ist eine überarbeitete Version des Codes:

```
VBA
Copy code
Sub openAI_API()

Dim name As String
Dim url As String
Dim http As New WinHttpRequest
Dim response As String
Dim obj As Object
Dim text As String
Dim wordApp As Object
Dim wordDoc As Object
Dim wordRange As Object
'Fragt nach dem Namen, der in der Tabelle gesucht werden soll
name = InputBox("Bitte geben Sie den Namen ein, nach dem Sie suchen möchten:")
'Sucht in der Excel-Tabelle nach dem Namen in Spalte B
Set obj = Worksheets("Sheet1").Range("B:B").Find(name)
'Wenn der Name gefunden wird, werden die Daten in der Zeile an die OpenAI-API übergeben
If Not obj Is Nothing Then

url = "https://api.openai.com/v1/engines/davinci-codex/completions"

http.Open "POST", url, False

http.setRequestHeader "Content-Type", "application/json"

http.setRequestHeader "Authorization", "Bearer API_KEY" 'ersetzen Sie API_KEY durch Ihren
eigenen API-Schlüssel

http.Send "{""prompt"":"" & obj.EntireRow.Cells(1).Value &
"", ""max_tokens"":100, ""n"":1, ""stop"":"".""}"

response = http.responseText
```

```
Set json = JsonConverter.ParseJson(response)
```

```
text = json("choices")(1)("text")
```

```
'Kopiert den generierten Text in eine Variable
```

```
Dim generatedText As String
```

```
generatedText = text
```

```
'Fügt den generierten Text in ein Word-Dokument ein
```

```
Set wordApp = CreateObject("Word.Application")
```

```
Set wordDoc = wordApp.Documents.Add
```

```
Set wordRange = wordDoc.Range
```

```
wordRange.InsertAfter generatedText
```

```
'Zeigt das Word-Dokument an
```

```
wordApp.Visible = True
```

```
Else
```

```
MsgBox "Der Name wurde nicht gefunden."
```

```
End If
```

```
End Sub
```

In diesem Code wird der generierte Text in der Variablen "generatedText" gespeichert und dann mithilfe der InsertAfter-Methode in das Word-Dokument eingefügt.

Dieser Code führte zu der Fehlermeldung „Laufzeitfehler 424 Objekt erforderlich“.

Chat GPT:

Der Fehler "Laufzeitfehler 424: Objekt erforderlich" tritt normalerweise auf, wenn das Skript versucht, auf ein Objekt zuzugreifen, das nicht definiert oder nicht initialisiert wurde.

In Ihrem Code kann dieser Fehler auftreten, wenn die Suche in der Tabelle keinen Treffer ergibt und die Variable "obj" nicht zugewiesen wird.

Um diesen Fehler zu vermeiden, sollten Sie sicherstellen, dass die Variable "obj" immer initialisiert wird. Sie können dies tun, indem Sie eine Bedingung hinzufügen, um zu prüfen, ob der Name gefunden wurde, bevor Sie versuchen, auf das Objekt zuzugreifen.

Hier ist eine überarbeitete Version des Codes, die diesen Fehler vermeiden sollte:

```
VBA
```

```
Copy code
```

```
Sub openAI_API()
```

```
Dim name As String
```

```
Dim url As String
```

```
Dim http As New WinHttpRequest
```

```
Dim response As String
```

```
Dim obj As Range
```

```
Dim text As String
```

```
Dim wordApp As Object
```

```
Dim wordDoc As Object
```

```
Dim wordRange As Object
```

```
'Fragt nach dem Namen, der in der Tabelle gesucht werden soll
```

```
name = InputBox("Bitte geben Sie den Namen ein, nach dem Sie suchen möchten:")
```

```
'Sucht in der Excel-Tabelle nach dem Namen in Spalte A
```

```
Set obj = Worksheets("Sheet1").Range("A:A").Find(name)
```

```
'Überprüft, ob der Name gefunden wurde, bevor auf das Objekt zugegriffen wird
```

```
If Not obj Is Nothing Then
```

```
'Wenn der Name gefunden wird, werden die Daten in der Zeile an die OpenAI-API übergeben
```

```
url = "https://api.openai.com/v1/engines/davinci-codex/completions"
```

```
http.Open "POST", url, False
```

```
http.setRequestHeader "Content-Type", "application/json"
```

```
http.setRequestHeader "Authorization", "Bearer API_KEY" 'ersetzen Sie API_KEY durch Ihren  
eigenen API-Schlüssel
```

```
http.Send "{"prompt"":"" & obj.EntireRow.Cells(1).Value &
```

```
""", ""max_tokens"":100, ""n"":1, ""stop"":"".""}"
```

```
response = http.responseText
```

```
Set json = JsonConverter.ParseJson(response)
```

```
text = json("choices")(1)("text")
```

```
'Kopiert den generierten Text in die Zwischenablage
```

```
With New MSForms.DataObject
```

```
.SetText text
```

```
.PutInClipboard
```

```
End With
```

```
'Fügt den generierten Text in ein Word-Dokument ein
```

```
Set wordApp = CreateObject("Word.Application")
```

```
Set wordDoc = wordApp.Documents.Add
```

```
Set wordRange = wordDoc.Range
```

```
wordRange.PasteAndFormat wdFormatPlainText
```

```
'Zeigt das Word-Dokument an
```

```
wordApp.Visible = True
```

```
Else
```

```
MsgBox "Der Name wurde nicht gefunden."
```

```
End If
```

End Sub

In diesem Code wird zuerst überprüft, ob der Name in der Tabelle gefunden wurde, bevor auf das Objekt "obj" zugegriffen wird. Wenn der Name nicht gefunden wird, wird eine Fehlermeldung ausgegeben und das Skript wird beendet. Wenn der Name gefunden wird, wird der generierte Text in die Zwischenablage kopiert und in das Word-Dokument eingefügt.

Anschließend trat erneut der Fehler „Benutzerdefinierter Typ nicht definiert“ auf, mit der Zeile „With New MSForms.DataObject“ markiert.

Chat GPT:

„Der Fehler "Benutzerdefinierter Typ nicht definiert " tritt normalerweise auf, wenn das Skript versucht, auf ein Objekt zuzugreifen, das in der aktuellen Umgebung nicht verfügbar ist.

In Ihrem Code wird die Klasse "MSForms.DataObject" verwendet, um Daten in die Zwischenablage zu kopieren. Diese Klasse ist Teil der "Microsoft Forms 2.0 Object Library", die möglicherweise nicht auf Ihrem Computer registriert ist oder nicht in Ihrem Projekt referenziert wurde.

Um dieses Problem zu beheben, gehen Sie wie folgt vor:

1. Klicken Sie in der VBA-Editor-Umgebung auf "Extras" > "Verweise".
2. Suchen Sie in der Liste der verfügbaren Verweise nach "Microsoft Forms 2.0 Object Library" und aktivieren Sie das Kontrollkästchen neben diesem Verweis.
3. Klicken Sie auf "OK", um die Änderungen zu speichern.

Wenn der Verweis nicht in der Liste enthalten ist, müssen Sie möglicherweise die "Microsoft Forms 2.0 Object Library" manuell registrieren. Gehen Sie dazu wie folgt vor:

1. Öffnen Sie die Eingabeaufforderung als Administrator.
2. Geben Sie den folgenden Befehl ein und drücken Sie die Eingabetaste:

```
regsvr32 C:\Windows\System32\FM20.DLL
```

Beachten Sie, dass der Pfad zu FM20.DLL je nach Ihrem Betriebssystem variieren kann.

3. Öffnen Sie das VBA-Projekt erneut und gehen Sie zu "Extras" > "Verweise".
4. Suchen Sie nach "Microsoft Forms 2.0 Object Library" und aktivieren Sie das Kontrollkästchen neben diesem Verweis.
5. Klicken Sie auf "OK", um die Änderungen zu speichern.

Sobald die "Microsoft Forms 2.0 Object Library" referenziert wurde, sollte der Fehler "Benutzerdefinierter Typ nicht definiert" nicht mehr auftreten.“

Bei dieser Antwort war vor allem der zweite Teil von Interesse. Da wir jedoch den Dateipfad der FM20.DLL nicht finden konnten, sind wir an dieser Stelle nicht weitergekommen und haben die Betreuer im nächsten Meeting nach Hilfe gefragt. Während des Meetings haben wir im Internet nach einer Lösung des Problems gesucht und sind schnell auf Stack Overflow zu einer ziemlich leichten Lösung gekommen.

<https://stackoverflow.com/questions/35610429/why-do-i-not-see-the-microsoft-forms-2-0-object-library>

Dort haben wir herausgefunden das man einfach im Dateibrowser auf dem PC nach der Datei FM20.DLL suchen kann und diese dann mit einem einmaligen Öffnen das Problem behebt. Danach war die „Microsoft Forms 2.0 Object Library“ in der Liste vorhanden und man konnte sie unter Extras > Verweise, mit einem Haken einbinden.

Nach einem weiteren ausführen des Codes sind wir auf den nächsten Fehler gestoßen. Diesmal hier der Fehler „Laufzeitfehler 13 typen unverträglich“ und die Zeile „text = json(“choices”)(1)(“text““)“ war markiert. Auf der Suche nach einer Lösung, sagte der Betreuer das, dass Problem bei der Extrahierung des generierten Textes liegt, da dieser in einem JSON-Format vorliegt und wir einen JSON Converter benötigen, um den Text in Word darzustellen. Wir sind anschließend im Internet auf eine GitHub (<https://github.com/VBA-tools/VBA-JSON/blob/master/README.md>)

Seite gestoßen, in dem der aktuelle JSON-Converter zum Download verfügbar und beiliegend eine Anleitung, um diesen zu installieren. An dieser Stelle war das Meeting jedoch zu ende.

Den Tag danach haben wir als Team den JSON-Converter installiert und getestet, um herauszufinden, ob das unser Problem löst.

Die Installation und Einbindung behoben das Problem allerdings noch nicht. Es kam die gleiche Fehlermeldung wie zuvor, „Laufzeitfehler 13 typen unverträglich“.

Da wir keinerlei Erfahrung mit VBA haben, fragten wir erneut Chat GPT nach einer Lösung.

Chat GPT:

„Es tut mir leid für den Fehler. Der Laufzeitfehler 13 tritt auf, wenn es einen Typenkonflikt gibt. In diesem Fall scheint es so, als ob das JSON-Objekt nicht wie erwartet aufgebaut ist oder dass das "choices"-Array möglicherweise nicht existiert oder nicht den erwarteten Index hat.“

Um das Problem zu beheben, überprüfen Sie bitte die Struktur des JSON-Antworttextes, den Sie von der OpenAI-API erhalten. Stellen Sie sicher, dass das "choices"-Array existiert und dass es den Index 1 hat.

Sie können auch eine Fehlerbehandlung hinzufügen, um sicherzustellen, dass der Code nicht abstürzt, wenn ein unerwartetes JSON-Format vorliegt. Hier ist eine angepasste Version des Codes mit Fehlerbehandlung:

```
```vba
' ...
```

```
Set json = JsonConverter.ParseJson(response)
```

```
On Error Resume Next
```

```
text = json("choices")(1)("text")
```

```
On Error GoTo 0
```

```
If text <> "" Then
```

```
' Kopiert den generierten Text in die Zwischenablage
```

```
With New MSForms.DataObject
```

```
.SetText text
```

```
.PutInClipboard
```

```
End With
```

```
' Fügt den generierten Text in ein Word-Dokument ein
```

```
Set wordApp = CreateObject("Word.Application")
```

```
Set wordDoc = wordApp.Documents.Add
```

```
Set wordRange = wordDoc.Range
```

```
wordRange.PasteAndFormat wdFormatPlainText
```

```
' Zeigt das Word-Dokument an
```

```
wordApp.Visible = True
```

```
Else
```

```
MsgBox "Fehler beim Extrahieren des generierten Textes."
```

```
End If
```

```
' ...
' ...
```

Die Fehlerbehandlung wird durch `On Error Resume Next` aktiviert. Wenn ein Fehler auftritt, wird der Code fortgesetzt, und `text` bleibt leer. Anschließend wird mit `On Error GoTo 0` die normale Fehlerbehandlung wiederhergestellt. Dann wird überprüft, ob `text` nicht leer ist, um sicherzustellen, dass der generierte Text erfolgreich extrahiert wurde, bevor er in das Word-Dokument eingefügt wird.“

Chat GPT gab uns eine Möglichkeit genauer herauszufinden, wo unser Fehler liegt, indem wir diesen Abschnitt in unseren Code einbauen. Nach ausführen des aktualisierten Codes erschien nun die Meldung „Fehler beim Extrahieren des generierten Textes“. Als wir das Chat GPT wieder übermittelt haben gab es uns einen weiteren Code Abschnitt, um den Fehler noch genauer zu diagnostizieren.

Chat GPT:

„Es tut mir leid, dass es immer noch Probleme beim Extrahieren des generierten Textes gibt. Der Fehler tritt höchstwahrscheinlich aufgrund einer inkonsistenten JSON-Struktur auf. Um das Problem weiter zu diagnostizieren, können Sie die Antwort der OpenAI-API überprüfen und sicherstellen, dass das JSON-Objekt die erwarteten Eigenschaften enthält.

Hier ist ein angepasster Code, der zusätzliche Fehlerbehandlung und Debugging-Informationen enthält, um das Problem weiter einzukreisen:

```
``vba
' ...
```

```
Set json = JsonConverter.ParseJson(response)
```

```
On Error Resume Next
text = json("choices")(1)("text")
On Error GoTo 0
```

```
If text <> "" Then
' Kopiert den generierten Text in die Zwischenablage
With New MSForms.DataObject
.SetText text
.PutInClipboard
End With
' Fügt den generierten Text in ein Word-Dokument ein
Set wordApp = CreateObject("Word.Application")
Set wordDoc = wordApp.Documents.Add
Set wordRange = wordDoc.Range
wordRange.PasteAndFormat wdFormatPlainText
' Zeigt das Word-Dokument an
```

```
wordApp.Visible = True
Else
' Fehlerbehandlung und Debugging-Informationen
Dim errorMessage As String
errorMessage = "Fehler beim Extrahieren des generierten Textes." & vbCrLf & vbCrLf
errorMessage = errorMessage & "JSON-Struktur: " & vbCrLf & response & vbCrLf & vbCrLf
errorMessage = errorMessage & "JSON-Objekt: " & vbCrLf & response_text
MsgBox errorMessage
End If
```

```
' ...
```

```
```
```

In diesem Code wird ein `errorMessage` erstellt, das sowohl die JSON-Struktur als auch das vollständige JSON-Objekt enthält. Die `MsgBox` zeigt diese Informationen an, um bei der Fehlerdiagnose zu helfen. Überprüfen Sie die angezeigten Informationen, um festzustellen,

ob das JSON-Objekt die erwarteten Eigenschaften enthält oder ob es unerwartete Unterschiede gibt.

Falls möglich, teilen Sie mir bitte den Inhalt der Fehlermeldung mit, damit ich Ihnen weiterhelfen kann, das Problem zu lösen.“

Diese Antwort hat uns geholfen das nächste Problem zu erkennen, denn es kam die Meldung:

„JSON-Structure: { error : { message : You exceeded your current quota, please check your plan and billing details. type: insufficient_quota, param: null, code, null“

Welche uns schon aufmerksam auf das eventuelle Problem machte und durch Chat GPT's Erklärung bestätigt wurde.

Chat GPT:

„Die Fehlermeldung besagt, dass Sie Ihr aktuelles Kontingent überschritten haben und Ihre Pläne und Abrechnungsdetails überprüfen sollten. Der Fehlercode lautet "insufficient_quota", was auf ein unzureichendes Kontingent hinweist.

Es scheint, dass Sie Ihr aktuelles Kontingent für die OpenAI-API überschritten haben, sodass keine weiteren Anfragen gesendet werden können. In diesem Fall müssen Sie Ihren OpenAI-Plan überprüfen und sicherstellen, dass Ihr Konto ausreichend Guthaben oder ein geeignetes Abonnement hat, um die API-Nutzung fortzusetzen.

Bitte überprüfen Sie Ihre OpenAI-Kontodetails, um sicherzustellen, dass Ihr Konto korrekt eingerichtet ist und über ausreichende Ressourcen verfügt, um die API-Anfragen durchzuführen. Wenn Sie weitere Unterstützung benötigen, wenden Sie sich bitte direkt an den OpenAI-Support.“

Diese Erklärung hat uns dazu gebracht unser OpenAI Konto zu überprüfen, ob wir noch Token zu Verfügung haben. Es stellte sich heraus, dass man bei einer Eröffnung eines OpenAI Kontos, Token im Wert von 5 Dollar erlangt, die drei Monate gültig sind. Unsere Tokens waren jedoch schon abgelaufen, daher eröffnet wir ein neues Konto und hatten dadurch Zugriff auf die zu Verfügung gestellten Token.

All diese Schritte haben trotzdem nicht zu einem funktionierendem code geführt. Die nächste Fehlermeldung lautete wieder „Fehler beim Extrahieren des generierten Textes“. An dieser Stelle entschieden wir uns Chat GPT nach einem alternativen Code zu fragen: „kannst du uns zu dem letzten genannten Code, eine Alternative geben“

Chat GPT:

```
Sub opAPI()  
Dim name As String  
Dim url As String  
Dim http As Object ' XMLHTTP-Objekt  
Dim response As String  
Dim obj As Range  
Dim wordApp As Object  
Dim wordDoc As Object  
Dim wordRange As Object  
' Fragt nach dem Namen, der in der Tabelle gesucht werden soll  
name = InputBox("Bitte geben Sie den Namen ein, nach dem Sie suchen möchten:")  
' Sucht in der Excel-Tabelle nach dem Namen in Spalte A  
  
Set obj = Worksheets("1234").Range("B:B").Find(name)  
' Überprüft, ob der Name gefunden wurde, bevor auf das Objekt zugegriffen wird  
If Not obj Is Nothing Then  
' Wenn der Name gefunden wird, werden die Daten in der Zeile an die OpenAI-API übergeben  
url = "https://api.openai.com/v1/engines/davinci-codex/completions"  
Set http = CreateObject("MSXML2.XMLHTTP")  
  
http.Open "POST", url, False  
http.setRequestHeader "Content-Type", "application/json"  
http.setRequestHeader "Authorization", "Bearer sk-  
jMScn8DjKNLyOKCqzGTkT3BlbkFJSphUnvSO4OetcF9reaTt" 'ersetzen Sie API_KEY durch Ihren  
eigenen API-Schlüssel  
http.Send "{"prompt"":"" & obj.EntireRow.Cells(1).Value &  
""", "max_tokens"":100, "n"":1, "stop"":""}""  
response = http.responseText  
' Extrahiert den generierten Text aus der API-Antwort  
Dim responseJson As Object  
Set responseJson = JsonConverter.ParseJson(response)  
  
Dim choices As Collection  
Set choices = responseJson("choices")  
  
If choices.Count > 0 Then  
Dim choice As Object  
Set choice = choices(1)  
Dim textArray As Variant  
textArray = choice("text")  
Dim text As String  
text = textArray(0)  
' Kopiert den generierten Text in die Zwischenablage  
With New MSForms.DataObject  
.SetText text  
.PutInClipboard  
End With  
' Fügt den generierten Text in ein Word-Dokument ein  
Set wordApp = CreateObject("Word.Application")  
Set wordDoc = wordApp.Documents.Add
```

```

Set wordRange = wordDoc.Range
wordRange.PasteAndFormat wdFormatPlainText
' Zeigt das Word-Dokument an
wordApp.Visible = True
Else
MsgBox "Der generierte Text konnte nicht abgerufen werden."
End If
' Kopiert den generierten Text in die Zwischenablage
With New MSForms.DataObject
.SetText text
.PutInClipboard
End With
' Fügt den generierten Text in ein Word-Dokument ein
Set wordApp = CreateObject("Word.Application")

Set wordDoc = wordApp.Documents.Add
Set wordRange = wordDoc.Range
wordRange.PasteAndFormat wdFormatPlainText
' Zeigt das Word

wordApp.Visible = True
Else
MsgBox "Der Name wurde nicht gefunden."
End If
End Sub

```

Damit hatten wir zwei verschiedene Versionen des Codes und haben uns entschieden mit der neuen Variante bis zum nächsten Meeting zu warten und die Betreuer um Hilfe zu fragen. Wir zeigten ihnen unseren aktuellen Code und demonstrierten das Problem. Kurz darauf hatte Dr. Georg Pietrek eine Idee, wie wir herausfinden können, wo im Code der Fehler vorliegt.

Dazu fügten wir die folgenden Zeilen in den Code ein:

```
Open "c:\test\log.txt" For Output As #1
```

Damit wird ein Text Dokument geöffnet. Verschiedene Print Anweisungen, die an unterschiedlichen Stellen um Code eingebaut wurden, sollten dann dabei helfen herauszufinden, bis wohin der Code korrekt funktioniert.

```

Sub opAPI()
Dim name As String
Dim url As String
Dim http As Object ' XMLHTTP-Objekt
Dim response As String
Dim obj As Range
Dim wordApp As Object
Dim wordDoc As Object
Dim wordRange As Object

```

```

Open "c:\test\log.txt" For Output As #1
Print #1, "test"
' Fragt nach dem Namen, der in der Tabelle gesucht werden soll
name = InputBox("Bitte geben Sie den Namen ein, nach dem Sie suchen möchten:")
Print #1, name
Close #1

```

In diesem Beispiel kann man sehen, wie dies umgesetzt wurde. Die Grün markierte Zeile, öffnet die .txt Datei. „Print #1, "test"“ steht direkt in der Zeile darunter, bevor der Code etwas ausführt. Das bedeutet das nach, das nach dem Ausführen, das Wort „test“ in der .txt Datei stehen sollte. Die zweite Anweisung „Print #1, name“ sollte demnach den Namen anzeigen, welcher in dem Violett markierten Zeile abgespeichert wird. Nach diesem Prinzip sind wir weiter vorgegangen und sind auf das erste Problem gestoßen.

```

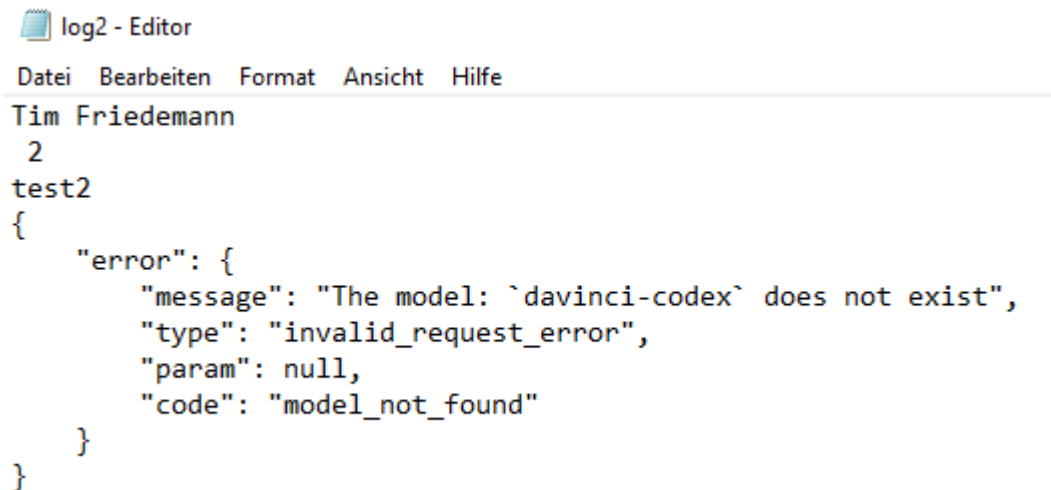
Open "c:\test\log2.txt" For Output As #1
' Sucht in der Excel-Tabelle nach dem Namen in Spalte A
Set obj = Worksheets("1234").Range("B:B").Find(name)
Print #1, obj
Print #1, obj.EntireRow.Cells(1).Value
' Überprüft, ob der Name gefunden wurde, bevor auf das Objekt zugegriffen wird
If Not obj Is Nothing Then
Print #1, "test2"

' Wenn der Name gefunden wird, werden die Daten in der Zeile an die OpenAI-API übergeben
url = "https://api.openai.com/v1/engines/davinci-codex/completions"
Set http = CreateObject("MSXML2.XMLHTTP")
http.Open "POST", url, False

http.setRequestHeader "Content-Type", "application/json"
http.setRequestHeader "Authorization", "Bearer sk-
jMSCn8DjKNLyOKCqzGtKt3BlbkFJSphUnvSO4OetcF9reaTt" 'ersetzen Sie API_KEY durch Ihren
eigenen API-Schlüssel
http.Send "{"prompt":"",""" & obj.EntireRow.Cells(1).Value &
""", ""max_tokens"":100,""n"":1,""stop"":"".""}"
response = http.responseText
Print #1, response
Close #1

```

Die .txt lieferte uns dann den ersten richtigen Hinweis.



```
log2 - Editor
Datei Bearbeiten Format Ansicht Hilfe
Tim Friedemann
2
test2
{
  "error": {
    "message": "The model: `davinci-codex` does not exist",
    "type": "invalid_request_error",
    "param": null,
    "code": "model_not_found"
  }
}
```

Abbildung 5 Screenshot Logdatei

Es ist auf diesem Bild zu sehen, was uns durch die Print Anweisungen ausgegeben wurde. Die ersten drei Print Anweisungen haben genau das ausgegeben, was erwartet war. Jedoch haben wir durch „Print#1, response“, festgestellt das das Modell, welches wir durch Chat GPT erhalten haben, nicht existiert oder die URL Fehlerhaft ist. Auf der OpenAi Website fanden wir die korrekte URL und fügten diese ein. Dadurch wurde das Problem noch nicht behoben und wir legten eine kurze Pause ein, in der wir Chat GPT den aktuellen code übermittelten und dazu schrieben „Wenn ich diesen Code ausführe, kommt die Meldung „Model existiert nicht“.

Chat GPT:

Es tut mir leid für das Missverständnis. Tatsächlich benötigt die OpenAI API den Modellparameter, um anzugeben, welches spezifische Modell für die Anfrage verwendet werden soll. Um das Modell in deinem Code anzugeben, musst du den payload-Teil der Anfrage entsprechend anpassen. Hier ist eine aktualisierte Version des Codes:

```
Sub opAPI()
Dim name As String
Dim url As String
Dim http As Object ' XMLHTTP-Objekt
Dim response As String
Dim obj As Range
```

```

Dim wordApp As Object
Dim wordDoc As Object
Dim wordRange As Object
' Fragt nach dem Namen, der in der Tabelle gesucht werden soll
name = InputBox("Bitte geben Sie den Namen ein, nach dem Sie suchen möchten:")
' Sucht in der Excel-Tabelle nach dem Namen in Spalte A
Set obj = Worksheets("1234").Range("B1:B10").Find(name)
' Überprüft, ob der Name gefunden wurde, bevor auf das Objekt zugegriffen wird
If Not obj Is Nothing Then
' Wenn der Name gefunden wird, werden die Daten in der Zeile an die OpenAI-API übergeben
url = "https://api.openai.com/v1/engines/davinci/completions"
Set http = CreateObject("MSXML2.XMLHTTP")
http.Open "POST", url, False
http.setRequestHeader "Content-Type", "application/json"
http.setRequestHeader "Authorization", "Bearer DEIN_API_SCHLÜSSEL_HIER_EINFÜGEN"
Dim payload As String
payload = "{"prompt":&obj.EntireRow.Cells(1).Value &
""", "max_tokens":100, "n":1, "stop":&""}"
http.send payload
response = http.responseText
' Extrahiert den generierten Text aus der API-Antwort
Dim responseJson As Object
Set responseJson = JsonConverter.ParseJson(response)
Dim choices As Collection
Set choices = responseJson("choices")
If choices.Count > 0 Then
Dim choice As Object
Set choice = choices(1)

Dim textArray As Variant
textArray = choice("text")
Dim text As String
text = textArray(0)
' Kopiert den generierten Text in die Zwischenablage
With New MSForms.DataObject
.SetText text
.PutInClipboard
End With
' Fügt den generierten Text in ein Word-Dokument ein
Set wordApp = CreateObject("Word.Application")
Set wordDoc = wordApp.Documents.Add
Set wordRange = wordDoc.Range
wordRange.PasteAndFormat wdFormatPlainText
' Zeigt das Word-Dokument an
wordApp.Visible = True
Else
MsgBox "Der generierte Text konnte nicht abgerufen werden."
End If
Else
MsgBox "Der Name wurde nicht gefunden."
End If
End Sub

```

Die URL des Models wurde immer noch falsch angegeben (orange), daher ersetzten wir sie durch die korrekte. Dem Betreuer ist aufgefallen das die rot markierte Zeile, nicht wie erwartet aussieht und kein Prompt vorhanden ist, der an „payload“ weitergegeben wird. Die rot markierte Zeile wurde uns dann wie folgt diktiert:

```
payload = "{ \"model\": \"text-davinci-003\", \"prompt\": \" \" & Frage & \"\", \"max_tokens\": 1000, \"n\": 1 }"
```

Außerdem haben wir die folgende Zeile hinzugefügt, welche eine Beispiel Frage enthält.

```
Frage = "Was ist der größte Berg der Welt?"
```

Zum Schluss bekamen wir noch die Anweisung, den grün markierten Teil zu entfernen und ihn durch die folgenden zwei Zeilen zu ersetzen:

```
Dim text As String  
text = responseJson("choices")(1)("text")
```

All diese Schritte resultierten dann in einem voll funktionsfähigen Code.

Der nächste Schritt war es dann für uns, die „Frage“ entsprechend für unser Ziel anzupassen. Um unser gewünschtes Ziel zu erreichen. Hierzu muss man jedoch den Datenschutz beachten. Sobald wir Persönliche Informationen an die API weitergeben, wäre der Datenschutz nicht mehr gewährleistet. Eine Lösung zu diesem Problem ist es, nur bestimmte Informationen an die API zu geben, die relevant für Erstellung des Arztbriefes sind. Wie ließen also Informationen wie Name und Geburtsdatum in der Anfrage weg. Die Frage änderten wir zu:

```
Frage = "Schreibe eine Arztbrief für einen Patienten mit dem Symptom " &  
obj.EntireRow.Cells(5).Value & " mit der Diagnose " & obj.EntireRow.Cells(6).Value & " und  
den verschriebenen Medikamenten " & obj.EntireRow.Cells(7).Value
```

Hier entsprechen die Argumente „obj.EntireRow.Cells(X).Value“ den einzelnen Spalten der Tabelle aus der die Informationen genommen werden. Diese Anpassungen führten zu einem Ergebnis, mit dem wir uns zufrieden gestellt haben. In der Praxis müsste man dann im nachhinein Persönliche Daten wie Name, Geburtsdatum und Wohnort ergänzen.

Sehr geehrter Herr X,

Ihr Besuch bei mir am gestrigen Tag zeigte, dass Sie unter Schwindel, Übelkeit und Kurzatmigkeit bei Belastung leiden. Daher habe ich eine gründliche Untersuchung durchgeführt. Nachdem alle Testergebnisse ausgewertet wurden, haben wir eine Diagnose des arteriellen Hypertonus (Bluthochdruck) bestätigt.

Aufgrund des vorliegenden Befunds habe ich Ihnen Betablocker verschrieben. Nehmen Sie die Medikamente täglich ein und stellen Sie sicher, dass die empfohlene Menge eingehalten wird.

Ich empfehle Ihnen auch, sich einer ausgewogenen Ernährung und einem regelmäßigen Trainingsprogramm zu widmen. Wenden Sie sich in dieser Hinsicht an Ihren Hausarzt, damit er Ihnen weitere Informationen geben kann.

Sollten sich die Symptome ohne erhebliche Besserung nach der Einnahme der verschriebenen Medikamente innerhalb von zwei Wochen verschlimmern, rufen Sie mich bitte an.

Mit freundlichen Grüßen,

Dr. X

Abbildung 6 Endergebnis Zweiter Ansatz

5. Fazit

Abschließend kann man sagen, dass die Nutzung von Chat GPT für Arztbriefe sowohl positive als auch negative Aspekte vorweisen kann. Bei komplizierten Fragestellungen, die über den Tellerrand hinaus gehen können, Chat GPT unzuverlässig sein und aufgrund der Fehler, die bei komplexeren Fragestellungen entstehen, kann ein erhöhter Zeitaufwand entstehen um diese zu beseitigen oder gar nachvollziehen zu können. Zudem sollten sich Programmierer bewusst sein, dass die Ausführung von Chat GPT nicht immer den effizientesten Code, oder die optimale Lösung generiert.

Für die Nutzung von Chat GPT spricht in jedem Fall, dass Chat GPT über die Grundlagen hinaus arbeitet und komplexe Inhalte begreifen und weiterverarbeiten kann. Des Weiteren schafft es eine schnelle, erste Grundlage für die Programmierung und agiert als schnelle Ersthilfe.

Betrachtet man die in diesem Projekt herausgearbeiteten Aspekte, so kann gesagt werden, dass Chat GPT als Ausgangspunkt fungieren kann, um grundlegende Codeabschnitte oder Konzepte zu verstehen und anschließend Lösungen zu finden. In der Verwendung von Chat GPT wird es wichtig sein zu erkennen, dass es an Grenzen stößt. Somit werden bei komplexen Aufgaben zusätzliche Überprüfungen und Optimierungen notwendig sein. Demnach sollten die Ergebnisse durch Chat GPT immer kritisch hinterfragt werden und diese mit anderen Quellen verifizieren, um Genauigkeit und Qualität sicherzustellen.

| | |
|---|----|
| Abbildung 2 Screenshot angepasste Exceltabelle / Datenbank | 7 |
| Abbildung 3 Screenshot Fehlerhafte Datenzuordnung / Erster Ansatz | 8 |
| Abbildung 4 Screenshot Ergebnis / Erster Ansatz | 9 |
| Abbildung 5 Screenshot Logdatei | 29 |
| Abbildung 6 Endergebnis Zweiter Ansatz | 32 |