Aaron Feldmann Skill Auflistung

Hardware Reparatur

- Löten
- SMD Löten
- Feines Handwerk
- Exakt
- Kabelbau
- Manuals verstehen
- Level 1 Reparaturen
- Level 2 Reparaturen
- Level 3 Reparaturen

Hardware Kentnisse

- Laptop
- Tablet
- Smartphone
- Desktop PC
- Display v TV
- Hi-Fi Audio
- Video
- Eventtechnik
- Projektor
- iMac
- iPod
- Kopfhörer
- Lautsprecher
- Car-Hi-Fi
- Smartwatch
- DVB Receiver
- Elektrotechnik Bauteile
- Drucker

Arbeitsabläufe

- Ersatzteilverwaltung
- Kostenvoranschläge erstellen
- Kundensupport
- Zusammenarbeit mit Kunden
- Garantieabgeltungen
- Fehlersuche
- Fehlerbehebung
- Refurbishment
- Automatisierung von Arbeit
- Sicherheitsbewusstsein

Software Systeme

- Manjaro/Arch Linux
- Debian/Ubuntu/Kali/Fedora Linux
- Apple IOS
- Mac OS und Mac OS ARM
- Apple iPad OS
- Android
- Manjaro ARM
- Windows

Software Kentnisse

- Microsoft Office
- Google Dienste
- Jira/Confluence
- Citrix
- Umgang mit KI
- GitHub
- Asana
- Notion
- NeoVim
- Terminal
- SAP
- Bildbearbeitung
- MS Teams

Erklärung

- Grundkentnisse
- Gutes theoretisches Wissen
- Erfahrung in Praktik und Theorie
- Gute praktische und theoretische Kentnisse
- Sehr gute praktische und theoretische Kentnisse

```python
import os
import csv
from reportlab.lib.colors import HexColor
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4
from reportlab.platypus import Paragraph
from reportlab.lib.styles import getSampleStyleSheet


def read_data_from_csv(file_path):
    """
    Liest die Daten aus einer CSV-Datei und gibt sie als Dictionary zurück.
    """
    data = {}
    with open(file_path, newline='', encoding='utf-8') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            category = row['Kategorie']
            icon = row['Icon'].strip() if row['Icon'] else None
            value = row['Wert'].strip(
            ) if 'Wert' in row and row['Wert'] else None
            if category not in data:
                data[category] = []
            data[category].append({
                "Name": row['Name'],
                "Icon": icon,
                "Value": value
            })

    return data


def set_icon_parameters(height=13, width=13, y_offset=3):
    """
    Definiert Standardparameter für Icons.
    """
    return {"height": height, "width": width, "y_offset": y_offset}


def get_color_for_value(value):
    """
    Gibt die entsprechende Farbe für einen numerischen Wert zurück.
    """
    if value == "1":
        return HexColor("#ADD8E6")  # Hellblau
    elif value == "2":
        return HexColor("#87CEEB")  # Mittelblau
    elif value == "3":
        return HexColor("#4682B4")  # Dunkelblau
    elif value == "4":
        return HexColor("#5F9EA0")  # Blaugrün
    elif value == "5":
        return HexColor("#2E8B57")  # Dunkelgrün
    else:
        return HexColor("#FFFFFF")  # Standard: Weiß


def create_pdf(filename, data, icons_folder, code_file):
    """
    Erstellt ein PDF mit den Daten aus der CSV-Datei, Icons und Python-Code.
    """
    c = canvas.Canvas(filename, pagesize=A4)
    width, height = A4
    margin = 50

    # Überschrift und Hauptinhalt
    add_header(c, width, height, margin)
    draw_categories(c, data, icons_folder, width, height,
                margin, (width - 2 * margin) / 2)
    add_explanation(c, data, margin, (width - 2 * margin) / 2, 100)

    # Python-Code hinzufügen
    add_code_with_columns(c, code_file, width, height, margin, num_columns=4)

    c.save()


def add_header(c, width, height, margin):
    """
    Fügt die Überschrift und die Trennlinie zum PDF hinzu.
    """
    c.setFont("Helvetica-Bold", 14)
    c.drawString(margin, height - margin - -15,
                 "Aaron Feldmann Skill Auflistung")
    c.line(margin, height - margin - -10,
           width - margin, height - margin - -10)


def draw_categories(
        c, data, icons_folder, width, height, margin, column_width):
    """
    Zeichnet die Kategorien, deren Namen und Icons in das PDF.
    """
    y_position_left = height - margin - 1
    y_position_right = height - margin - 1

    # Kategorien links und rechts definieren
    left_categories = ["Hardware Kentnisse",
                       "Arbeitsabläufe", "Hardware Reparatur"]
    right_categories = ["Software Systeme",
                        "Programiersprachen", "Software Kentnisse"]

    icon_params = set_icon_parameters(height=13, width=13, y_offset=3)

    for category, items in data.items():
        # Bestimme Position basierend auf der Kategorie
        if category in left_categories:
            y_position = y_position_left
            x_position = margin
        elif category in right_categories:
            y_position = y_position_right
            x_position = margin + column_width
        else:
            continue

        # Kategorieüberschrift
        c.setFont("Helvetica-Bold", 12)
        c.setFillColor(HexColor("#000000"))
        c.drawString(x_position, y_position, category)
        y_position -= 15

        # Einträge in der Kategorie
        for item in items:
            draw_item(c, item, x_position, y_position,
                      icons_folder, icon_params)
            y_position -= 20

        # Aktualisiere die Y-Position der entsprechenden Spalte
        if category in left_categories:
            y_position_left = y_position
        elif category in right_categories:
            y_position_right = y_position


def draw_item(c, item, x_position, y_position, icons_folder, icon_params):
    """
    Zeichnet einen einzelnen Eintrag mit Namen, Icons und Farben.
    """
    name = item["Name"]
    icon_path = os.path.join(
        icons_folder, item["Icon"]) if item["Icon"] else None
    color = get_color_for_value(item["Value"])
    num_icons = int(
        item["Value"]) if item["Value"] and item["Value"].isdigit() else 1

    # Namen zeichnen
    c.setFillColor(HexColor("#000000"))
    c.setFont("Helvetica", 12)
    c.drawString(x_position + 5, y_position - 10, name)
    text_width = c.stringWidth(name, "Helvetica", 12)

    # Icons zeichnen
    if icon_path and os.path.isfile(icon_path):
        try:
            for i in range(num_icons):
                icon_x_position = x_position + 25 + \
                    text_width + i * (icon_params["width"]
                c.setFillColor(color)
                c.rect(
                    icon_x_position - 5,
                    y_position - 17,
                    icon_params["width"] + 10,
                    icon_params["height"] + 5,
                    stroke=0,
                    fill=1,
                )
                c.drawImage(
                    icon_path,
                    icon_x_position,
                    y_position - 15,
                    width=icon_params["width"],
                    height=icon_params["height"],
                    mask="auto",
                )
        except Exception as e:
            print(f"Fehler bei Icon '{icon_path}': {e}")


def add_explanation(c, data, margin, column_width, bottom_margin):
    """
    Fügt die Erklärung mit farbigen Blöcken zum PDF hinzu.
    """
    explanation_x = margin + column_width
    explanation_y = bottom_margin + 200

    c.setFont("Helvetica-Bold", 15)
    c.setFillColor(HexColor("#000000"))
    c.drawString(explanation_x, explanation_y + 20, "Erklärung")
    c.line(
        explanation_x,
        explanation_y + 18,
        explanation_x + c.stringWidth("Erklärung", "Helvetica-Bold", 15),
        explanation_y + 18,
    )

    c.setFont("Helvetica", 10)
    for item in data.get("Erklärung", []):
        name = item["Name"]
        color = get_color_for_value(item["Value"])
        c.setFillColor(color)
        c.rect(explanation_x, explanation_y, 25, 15, stro
        c.setFillColor(HexColor("#000000"))
        c.drawString(explanation_x + 5, explanation_y + 3,
        explanation_y -= 20


def add_logo_and_description(c, icons_folder, width, margi
    """
    Fügt das Python-Logo und den Beschreibungstext zum PDF
    """
    logo_path = os.path.join(icons_folder, "pl.png")
    logo_width = 150
    logo_height = 150
    logo_x = width - margin - logo_width - 50
    logo_y = height - margin - logo_height - 120

    # Logo hinzufügen
    if os.path.isfile(logo_path):
        try:
            c.drawImage(
                logo_path,
                logo_x,
                logo_y,
                width=logo_width,
                height=logo_he
                mask="auto",
            )
        except Exception as e:
            print(f"Fehler bei
    else:
        print("Das Python-Logo

    # Beschreibungstext unter
    text = (
        "Dieses Dokument wurde
        "erstellt. Den Source
    )
    styles = getSampleStyleShe
    style = styles["Normal"]
    style.fontName = "Helvetic
    style.fontSize = 10
    style.leading = 12
    style.textColor = HexColor
    paragraph = Paragraph(text
    paragraph.wrapOn(c, logo_w
    paragraph.drawOn(c, logo_x


def add_code_with_columns(c, c
    """
    Fügt den Python-Code auf e
    verteilt auf mehrere Spalt
    """
    font_size = 8  # Schriftgr
    line_spacing = 2  # Zeilen
    column_width = (width - 2
    column_margin = 10  # Abst

    c.showPage()  # Neue Seite
    c.setFont("Courier", font_
    y_position = height - marg
    column_index = 0  # Spalte

    with open(code_file, "r")
        for line in file:
            # Berechne X-Posit
            x_position = margi

            # Wechsle zu nächs
            if y_position < ma
                column_index +
                y_position = h

                # Wechsle zu n
                if column_inde
                    c.showPage
                    c.setFont(
                    column_ind
                    y_position

            # Füge die aktuel
            c.drawString(x_pos
            y_position -=


if __name__ == "__main__":
    script_dir = os.path.dirname(o
    csv_file_path = os.path.join(s
    icons_folder = os.path.join(sc
    output_pdf_path = os.path.join

    data = read_data_from_csv(csv_
    code_file_path = os.path.join(
    create_pdf(output_pdf_path, da
```