**Pflichtenheft**

**1. Einführung**

Dieses Pflichtenheft beschreibt die technische Umsetzung des Projekts „Alt, arm, abgehängt? Datengetriebene Perspektiven auf Altersarmut und Rentensysteme in Europa“. Es basiert auf den Anforderungen des Lastenhefts und legt fest, wie diese durch technische Lösungen, Werkzeuge und Methoden realisiert werden. Ziel ist es, eine datengetriebene Analyse der Altersarmut und Rentensysteme in der EU durchzuführen, Muster und Trends zu identifizieren und diese in einer Data Story sowie einer Multimedia-Präsentation zu präsentieren.

**2. Systemübersicht**

Das System besteht aus einer Datenpipeline zur Erfassung, Validierung, Analyse und Visualisierung von EU-Daten zur Altersarmut und Rentenausgaben. Die Hauptkomponenten sind:

* **Datenerfassung**: Python-Skripte für den Import von Daten aus data.europa.eu und Eurostat.
* **Datenverarbeitung**: Datenbereinigung und -analyse mit Python-Bibliotheken.
* **Visualisierung**: Erstellung von Heatmaps, Zeitreihen und Scatterplots.
* **Ausgabe**: Data Story in Markdown/PDF und Multimedia-Video.

Die Daten werden lokal gespeichert und über GitHub versioniert.

**3. Technische Umsetzung der Anforderungen**

**3.1 Anforderung 1: Datenerhebung und Datenvalidierung**

**Priorität: Muss**

**3.1.1 Technische Lösung**

Die Datenerhebung erfolgt über data.europa.eu und Eurostat. Python-Skripte mit der Bibliothek Pandas werden entwickelt, um Daten im CSV- oder Excel-Format über APIs oder manuelle Downloads zu beziehen. Die Validierung umfasst:

* **Download**: Automatisierter Import von Datensätzen (z. B. Armutsgefährdungsquote ab 65 Jahren, Rentenausgaben in % des BIP, soziodemografische Merkmale) für mindestens 20 EU-Staaten über 5 Jahre (2018–2022).
* **Konsistenzprüfung**: Überprüfung auf einheitliche Formate (z. B. Währung, Zeiteinheiten) mit Pandas.isna() und Vergleich mit Eurostat-Metadaten.
* **Fehlwertbehandlung**: Imputation durch Mittelwerte oder Interpolation, falls methodisch vertretbar.
* **Dokumentation**: Erstellung eines Datenwörterbuchs in CSV-Format, das Quellen und Anpassungen dokumentiert.

**3.1.2 Werkzeuge und Technologien**

* Programmiersprache: Python 3.8+
* Bibliotheken: Pandas für Datenmanipulation und Strukturierung, Requests (für API-Zugriffe), Openpyxl (für Excel-Dateien).
* Speicherung: Lokale Verzeichnisse mit Backup auf GitHub.

**3.1.3 Wechselwirkungen**

Diese Anforderung ist die Grundlage für alle weiteren Schritte (Analyse, Visualisierung, Empfehlungen). Fehlerhafte Daten würden die Aussagekraft der Ergebnisse beeinträchtigen.

**3.1.4 Risiken**

* Uneinheitliche Datenformate: Lösung durch Transformation in ein Standardformat (CSV).
* Fehlende Daten: Fallback auf ältere Jahre oder andere Indikatoren.
* API-Probleme: Manuelle Downloads als Alternative.

**3.1.5 Vergleich mit bestehenden Lösungen**

Die automatisierte Verarbeitung mit Python ist effizienter als manuelle Excel-Analysen, die in vielen Studien üblich sind, und ermöglicht reproduzierbare Ergebnisse.

**3.1.6 Grobschätzung des Aufwands**

* Skriptentwicklung: 6 Stunden
* Datenvalidierung: 8 Stunden
* Dokumentation: 4 Stunden
* **Gesamt**: 18 Stunden

**3.1.7 Teststrategie**

* Unit-Tests für Download-Funktionen (z. B. pytest).
* Manuelle Stichprobenprüfung für 5 Länder und 2 Jahre.
* Plausibilitätsprüfung durch Vergleich mit Eurostat-Berichten.

**3.2 Anforderung 2: Analyse der Altersarmutsverteilung**

**Priorität: Muss**

**3.2.1 Technische Lösung**

Die Analyse wird mit Pandas, NumPy und SciPy durchgeführt:

* **Deskriptive Statistik**: Berechnung von Mittelwert, Median, Standardabweichung und Extremwerten mit Pandas.describe().
* **Zeitreihenanalyse**: Glättung von Trends mit gleitendem Durchschnitt (Pandas.rolling()).
* **Subgruppenanalyse**: Aggregation nach Geschlecht, Bildung und Haushaltsform mit Pandas.groupby().
* **Clustering**: Anwendung von k-Means (scikit-learn) zur Identifikation von Ländern mit ähnlichen Armutsmustern.

**3.2.2 Werkzeuge und Technologien**

* Bibliotheken: Pandas, NumPy, SciPy, scikit-learn.
* Entwicklungsumgebung: Jupyter Notebook für interaktive Analyse.

**3.2.3 Wechselwirkungen**

Basiert auf Anforderung 1; Ergebnisse fließen in Visualisierungen (3.3) und Empfehlungen (3.5) ein.

**3.2.4 Risiken**

* Unvollständige Subgruppen-Daten: Reduktion auf verfügbare Kategorien.
* Verzerrungen durch Erhebungsmethodik: Validierung mit Metadaten.
* Hoher Rechenaufwand: Optimierung durch Teil-Datensätze.

**3.2.5 Vergleich mit bestehenden Lösungen**

Die Kombination aus deskriptiver Statistik und Clustering bietet detailliertere Einblicke als einfache aggregierte Analysen in EU-Berichten.

**3.2.6 Grobschätzung des Aufwands**

* Deskriptive Analyse: 6 Stunden
* Zeitreihen- und Subgruppenanalyse: 8 Stunden
* Clustering: 6 Stunden
* **Gesamt**: 20 Stunden

**3.2.7 Teststrategie**

* Vergleich der Ergebnisse mit Eurostat-Berichten.
* Plausibilitätsprüfung (z. B. keine negativen Quoten).
* Kreuzvalidierung von Clustering-Ergebnissen.

**3.3 Anforderung 3: Visualisierung der Ergebnisse**

**Priorität: Muss**

**3.3.1 Technische Lösung**

Visualisierungen werden mit Matplotlib, Seaborn und Plotly erstellt:

* **Heatmaps**: Darstellung der Armutsquoten nach Ländern (Seaborn.heatmap()).
* **Zeitreihen**: Liniendiagramme für Trends (Matplotlib.plot()).
* **Scatterplots**: Zusammenhänge zwischen Rentenausgaben und Armutsquoten (Plotly.express.scatter()).
* **Barrierefreiheit**: Kontrastreiche Farben (WCAG 2.1), Schriftgröße ≥12pt.

**3.3.2 Werkzeuge und Technologien**

* Bibliotheken: Matplotlib, Seaborn, Plotly.
* Design: Canva für Feinabstimmung.

**3.3.3 Wechselwirkungen**

Abhängig von Anforderung 2; Voraussetzung für Data Story (3.5).

**3.3.4 Risiken**

* Missverständliche Darstellungen: Vereinfachung durch Reduktion der Datenpunkte.
* Export-Probleme: Test in mehreren Formaten (PNG, JPEG).
* Inkonsistentes Design: Einheitliches Farbschema (EU-Blau/Gelb).

**3.3.5 Vergleich mit bestehenden Lösungen**

Die Verwendung interaktiver Plotly-Grafiken hebt die Visualisierungen von statischen Berichten ab.

**3.3.6 Grobschätzung des Aufwands**

* Erstellung in Python: 6 Stunden
* Design-Optimierung: 4 Stunden
* Export und Barrierefreiheit: 3 Stunden
* **Gesamt**: 13 Stunden

**3.3.7 Teststrategie**

* Überprüfung der Lesbarkeit auf verschiedenen Geräten.
* Testexport in PNG/JPEG.
* Peer-Review auf Design-Klarheit.

**3.4 Anforderung 4: Korrelation Rentenausgaben und Altersarmut**

**Priorität: Muss**

**3.4.1 Technische Lösung**

* **Korrelation**: Pearson-Koeffizient mit scipy.stats.pearsonr().
* **Regression**: Lineare Regression mit statsmodels.OLS().
* **Robustheit**: Prüfung auf Drittvariablen (z. B. Bildung) mit multipler Regression.

**3.4.2 Werkzeuge und Technologien**

* Bibliotheken: SciPy, Statsmodels.
* Daten: Kombinierte Datensätze aus Anforderung 1.

**3.4.3 Wechselwirkungen**

Verknüpft Anforderungen 1 und 2; Ergebnisse fließen in Anforderung 5 ein.

**3.4.4 Risiken**

* Drittvariablen: Berücksichtigung durch multiple Regression.
* Fehlende Daten: Fallback auf aggregierte Werte.
* Überinterpretation: Fokus auf statistische Signifikanz (p < 0.05).

**3.4.5 Vergleich mit bestehenden Lösungen**

Die quantitative Analyse mit Regressionsmodellen ist präziser als qualitative Studien.

**3.4.6 Grobschätzung des Aufwands**

* Korrelation und Regression: 5 Stunden
* Validierung: 5 Stunden
* Visualisierung: 2 Stunden
* **Gesamt**: 12 Stunden

**3.4.7 Teststrategie**

* Kreuzvalidierung mit Teil-Datensätzen.
* Signifikanzprüfung (p < 0.05).
* Vergleich mit Literaturwerten.

**3.5 Anforderung 5: Politische Handlungsempfehlungen**

**Priorität: Muss**

**3.5.1 Technische Lösung**

* **Ableitung**: Synthese der Ergebnisse in einem Markdown-Dokument.
* **Formulierung**: Strukturierte Empfehlungen mit Verweis auf statistische Ergebnisse.
* **Integration**: Einbindung von Visualisierungen zur Untermauerung.

**3.5.2 Werkzeuge und Technologien**

* Text: Markdown, konvertiert zu PDF.
* Visualisierung: Integration von Grafiken aus Anforderung 3.

**3.5.3 Wechselwirkungen**

Abhängig von allen vorherigen Anforderungen; zentral für Data Story und Video.

**3.5.4 Risiken**

* Fehlende Generalisierbarkeit: Fokus auf spezifische Länder/Gruppen.
* Sensible Formulierungen: Team-Review zur Neutralität.

**3.5.5 Vergleich mit bestehenden Lösungen**

Konkrete, datenbasierte Vorschläge heben sich von allgemeinen EU-Strategien ab.

**3.5.6 Grobschätzung des Aufwands**

* Auswertung: 4 Stunden
* Formulierung: 4 Stunden
* Review: 2 Stunden
* **Gesamt**: 10 Stunden

**3.5.7 Teststrategie**

* Peer-Review auf Logik und Verständlichkeit.
* Abgleich mit Projektzielen.

**4. Systemarchitektur**

Die Datenpipeline umfasst:

* **Eingabe**: Rohdaten (CSV/Excel) von data.europa.eu.
* **Verarbeitung**: Jupyter Notebooks für Datenbereinigung, Analyse und Visualisierung.
* **Ausgabe**: CSV-Tabellen, PNG/JPEG-Grafiken, Markdown/PDF-Bericht, Video.
* **Versionskontrolle**: GitHub für Code, Daten und Dokumente.

**5. Entwicklungsumgebung**

* Tools: Python 3.8+, Jupyter Notebook, VS Code, Git/GitHub.
* Bibliotheken: Pandas, NumPy, SciPy, Matplotlib, Seaborn, Statsmodels, Plotly, Openpyxl, Requests.

**6. Risiken und Maßnahmen**

* **Datenprobleme**: Fallback auf alternative Jahre oder Indikatoren.
* **Technische Fehler**: Debugging-Zeit (10% des Aufwands) einplanen.
* **Teamkoordination**: Wöchentliche Check-ins und klare Aufgabenverteilung.

**7. Zeitplan**

Die Umsetzung folgt den Meilensteinen des Projektauftrags:

* Datenrecherche: bis 29. Mai 2025
* Analyse: bis 12. Juni 2025
* Visualisierungen: bis 26. Juni 2025
* Data Story und Video: bis 10. Juli 2025
* Abschlusspräsentation: 11. Juli 2025