**Pflichtenheft**

**1. Einführung**

Dieses Pflichtenheft beschreibt die technische Umsetzung des Projekts "Alt, arm, abgehängt? Datengetriebene Perspektiven auf Altersarmut und Rentensysteme in Europa," basierend auf den Anforderungen des Lastenhefts. Es legt fest, wie die im Lastenheft beschriebenen Anforderungen durch technische Lösungen, Werkzeuge und Methoden realisiert werden. Ziel ist es, eine datengetriebene Analyse der Altersarmut und Rentensysteme in der EU durchzuführen und die Ergebnisse 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 technischen Komponenten umfassen Python-Skripte für Datenverarbeitung, Visualisierungswerkzeuge für grafische Darstellungen und Multimedia-Software für die Videoerstellung.

**3. Technische Umsetzung der Anforderungen**

**3.1 Anforderung 1: Datenerhebung und Datenvalidierung**

**3.1.1 Technische Lösung**

* **Datenquellen**: Daten werden ausschließlich von data.europa.eu und Eurostat bezogen, z. B. Armutsgefährdungsquoten, Rentenausgaben (% BIP) und soziodemografische Merkmale.
* **Automatisierung**: Ein Python-Skript wird entwickelt, um Datensätze im CSV- oder Excel-Format über APIs oder manuelle Downloads zu beziehen.
* **Validierung**: Daten werden auf Vollständigkeit (mind. 20 EU-Staaten, 5 Jahre), Konsistenz (einheitliche Definitionen) und Korrektheit (plausible Werte) geprüft.

**3.1.2 Werkzeuge und Technologien**

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

**3.1.3 Umsetzungsdetails**

* **Download**: Skript lädt Daten für mindestens 20 EU-Staaten über 5 Jahre (z. B. 2018–2022).
* **Validierungsschritte**:
  + Prüfung fehlender Werte mit Pandas (isna()).
  + Konsistenzprüfung durch Vergleich mit Metadaten von Eurostat.
  + Dokumentation in einem Datenwörterbuch (CSV-Datei).
* **Performance**: Verarbeitung von Datensätzen bis 100 MB innerhalb von 2 Stunden.

**3.1.4 Teststrategie**

* Unit-Tests für Download-Funktionalität.
* Manuelle Stichprobenprüfung (z. B. 5 Länder, 2 Jahre) auf Plausibilität.

**3.2 Anforderung 2: Analyse der Altersarmutsverteilung**

**3.2.1 Technische Lösung**

* **Deskriptive Analyse**: Berechnung von Mittelwert, Median, Standardabweichung und Extremwerten der Armutsquoten.
* **Zeitreihenanalyse**: Untersuchung von Trends über 5 Jahre pro Land.
* **Subgruppenanalyse**: Vergleich nach Geschlecht, Bildung und Haushaltsform.

**3.2.2 Werkzeuge und Technologien**

* **Bibliotheken**:
  + Pandas für Datenaggregation.
  + NumPy und SciPy für statistische Berechnungen.
  + Matplotlib für explorative Visualisierungen.

**3.2.3 Umsetzungsdetails**

* **Verteilung**: Aggregierte Statistiken pro Land mit Pandas.groupby().
* **Trends**: Zeitreihen mit gleitendem Durchschnitt zur Glättung.
* **Subgruppen**: Filtern der Daten mit Pandas.query() und Berechnung separater Statistiken.
* **Ausgabe**: Tabellen (CSV) und Rohgrafiken zur Weiterverarbeitung.

**3.2.4 Teststrategie**

* Validierung der Ergebnisse durch Vergleich mit Eurostat-Berichten.
* Plausibilitätsprüfung (z. B. keine negativen Armutsquoten).

**3.3 Anforderung 3: Visualisierung der Ergebnisse**

**3.3.1 Technische Lösung**

* **Visualisierungstypen**:
  + Heatmaps für Ländervergleiche.
  + Zeitreihendiagramme für Trends.
  + Scatterplots für Korrelationen.
* **Export**: Grafiken als PNG/JPEG mit hoher Auflösung.
* **Barrierefreiheit**: Kontrastreiche Farben, klare Beschriftungen.

**3.3.2 Werkzeuge und Technologien**

* **Bibliotheken**:
  + Matplotlib und Seaborn für statische Visualisierungen.
  + Plotly für interaktive Grafiken (falls in Video eingebettet).
* **Feinabstimmung**: Canva für Design-Optimierung.

**3.3.3 Umsetzungsdetails**

* **Heatmaps**: Erstellung mit Seaborn.heatmap() basierend auf aggregierten Armutsquoten.
* **Zeitreihen**: Liniendiagramme mit Matplotlib.plot() und Beschriftung pro Jahr.
* **Scatterplots**: Punktdiagramme mit Plotly.express.scatter() für dynamische Darstellung.
* **Design**: Farbschema gemäß EU-Richtlinien (Blau/Gelb), Schriftgröße ≥12pt.

**3.3.4 Teststrategie**

* Überprüfung der Lesbarkeit auf verschiedenen Bildschirmen.
* Testexport in mehreren Formaten.

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

**3.4.1 Technische Lösung**

* **Korrelationsanalyse**: Berechnung des Pearson-Koeffizienten für lineare Zusammenhänge.
* **Regression**: Lineare Regression zur Quantifizierung des Einflusses von Rentenausgaben auf Armutsquoten.

**3.4.2 Werkzeuge und Technologien**

* **Bibliotheken**:
  + SciPy.stats für Korrelation (pearsonr).
  + Statsmodels für Regressionsmodelle.
* **Datenbasis**: Kombinierte Datensätze aus Anforderung 1.

**3.4.3 Umsetzungsdetails**

* **Pearson**: Berechnung mit scipy.stats.pearsonr() inkl. p-Wert zur Signifikanzprüfung.
* **Regression**: Modell mit statsmodels.OLS(); Ausgabe von Koeffizienten und R².
* **Alternative**: Spearman-Korrelation (scipy.stats.spearmanr()) bei nicht-normalverteilten Daten.
* **Dokumentation**: Ergebnisse in Tabellen und Grafiken (Scatterplot mit Regressionslinie).

**3.4.4 Teststrategie**

* Kreuzvalidierung mit Teil-Datensätzen.
* Überprüfung der Signifikanz (p < 0.05).

**3.5 Anforderung 5: Politische Handlungsempfehlungen**

**3.5.1 Technische Lösung**

* **Ableitung**: Synthese der Analyseergebnisse (Anforderungen 2–4) in narrativer Form.
* **Dokumentation**: Strukturierte Empfehlungen in der Data Story.

**3.5.2 Werkzeuge und Technologien**

* **Textverarbeitung**: Markdown für die Data Story, konvertiert zu PDF.
* **Visualisierung**: Integration relevanter Grafiken zur Untermauerung.

**3.5.3 Umsetzungsdetails**

* **Struktur**: Empfehlungen nach Ländern, Risikogruppen und politischen Stellschrauben gegliedert.
* **Begründung**: Verweis auf statistische Ergebnisse (z. B. „Land X reduziert Armut um Y% bei Z% BIP-Ausgaben“).
* **Formatierung**: Klare Absätze, Hervorhebungen für Schlüsselvorschläge.

**3.5.4 Teststrategie**

* Peer-Review im Team auf Logik und Verständlichkeit.
* Abgleich mit Projektzielen aus dem Projektauftrag.

**4. Systemarchitektur**

* **Datenfluss**:
  + Eingabe: Rohdaten von data.europa.eu.
  + Verarbeitung: Python-Skripte in Jupyter Notebooks.
  + Ausgabe: Visualisierungen und Berichte.
* **Komponenten**:
  + Lokale Datenspeicherung (CSV/Excel).
  + GitHub als Versionskontrolle und Backup.

**5. Entwicklungsumgebung**

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

**6. Risiken und Maßnahmen**

* **Datenprobleme**: Fallback auf alternative Jahre/Länder bei fehlenden Daten.
* **Technische Fehler**: Debugging-Zeit einplanen (10% des Aufwands).
* **Teamkoordination**: Wöchentliche Check-ins zur Aufgabenabstimmung.

**7. Zeitplan**

Die Umsetzung orientiert sich an den Meilensteinen des Projektauftrags (z. B. Datenrecherche bis 29. Mai 2025, Analyse bis 12. Juni 2025).