1 Softwarearchitektur und Implementierung

Die Software zur Erstellung personalisierter Cocktailrezepte basiert auf etablierten Python-Bibliotheken wie `pandas`, `numpy`, `scikit-learn` und `TensorFlow/Keras`. Ziel der Architektur war es, eine flexible, skalierbare und wartbare Lösung zu schaffen, die komplexe Geschmacksprofile analysiert und darauf aufbauend automatisiert Cocktailrezepte generiert.

1.1 Softwarearchitektur

Die entwickelte Software folgt einem modularen Ansatz, der eine einfache Erweiterbarkeit und gute Wartbarkeit ermöglicht. Die Hauptkomponenten der Software umfassen:

- **Datenverwaltung**: Verwaltung und Normalisierung der Zutaten- und Geschmacksdaten.
- **Machine-Learning-Modell**: Ein Autoencoder, der latente Geschmackspräferenzen extrahiert und Nutzerprofile erstellt.
- **Optimierungsmodul**: Ermittlung der optimalen Cocktailmischung mittels Least-Squares-Optimierung.
- **Benutzerschnittstelle**: Einfache Interaktion zur Eingabe der Präferenzen und Ausgabe der Rezeptvorschläge.

1.2 Datenverwaltung und Vorverarbeitung

Die Basis der Software bildet eine JSON-basierte Datenbank, in der Zutaten mit ihren jeweiligen Geschmacksprofilen gespeichert werden. Ein Beispiel für eine solche Struktur:

```
{
    "Zutat": "Limettensaft",
    "Geschmack": [0.1, 0.9, 0.0, 0.0, 0.0]
}
```

Diese JSON-Daten werden mittels `pandas` in ein DataFrame geladen. Vor der weiteren Verarbeitung erfolgt eine Normalisierung mit `scikit-learn`, um eine gleichmäßige Skalierung der Geschmacksdimensionen zu gewährleisten.

1.3 Modellierung mittels Autoencoder

Das zentrale Machine-Learning-Modell des Systems ist ein Autoencoder, implementiert mit `TensorFlow/Keras`. Der Autoencoder reduziert das fünf-dimensionale Geschmacksprofil auf eine niedrigdimensionale latente Repräsentation und rekonstruiert dieses wieder:

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense

input_layer = Input(shape=(5,))
encoder = Dense(3, activation='relu')(input_layer)
decoder = Dense(5, activation='sigmoid')(encoder)

autoencoder = Model(inputs=input_layer, outputs=decoder)
autoencoder.compile(optimizer='adam', loss='mse')
```

Diese Architektur erlaubt, verborgene Strukturen in den Geschmacksdaten aufzudecken und effektiv für die Optimierung zu nutzen.

1.4 Stand der Technik

Aktuelle Entwicklungen im Bereich der künstlichen Intelligenz setzen häufig auf Transformer-basierte Modelle, da diese gut mit komplexen und umfangreichen Datensätzen umgehen können. Für dieses Projekt wurde jedoch bewusst ein Autoencoder gewählt, da er besonders geeignet ist, kleine Datensätze nicht-linear zu komprimieren und Muster effektiv zu extrahieren. Transformer würden hier aufgrund des überschaubaren Umfangs der Daten unnötige Komplexität und Rechenleistung erfordern.

1.5 Optimierung der Zutatenmischung

Auf Basis des rekonstruierten Nutzerprofils erfolgt die Berechnung der optimalen Zutatenmischung durch Lösung eines Least-Squares-Problems:

$$\min_{w} \|Aw - p_{\text{rekon}}\|^2 \tag{1}$$

Hierbei beschreibt A die Matrix der Zutatenprofile und w den Gewichtsvektor der Zutaten. Die Berechnung erfolgt mit `numpy.linalg.lstsq`, wobei negative Gewichte ausgeschlossen und die Ergebnisse normiert werden.

1.6 Zusammenfassung und Ausblick

Die entwickelte Software ermöglicht eine präzise, individuelle Rezeptgenerierung durch die Kombination von Machine Learning und mathematischer Optimierung. Obwohl der Ansatz flexibel und skalierbar ist, könnten Einschränkungen in der Qualität der Vorschläge auftreten, etwa wenn die Datengrundlage unvollständig oder nicht ausreichend umfangreich ist. Zukünftig könnte die Integration weiterer Zutaten, komplexerer Geschmacksprofile und tiefergehender Nutzerinteraktionen die Genauigkeit und Personalisierung weiter

verbessern. Außerdem wäre eine benutzerfreundliche App-Schnittstelle denkbar, um die Anwendung attraktiver und einfacher zugänglich zu gestalten.