### Schritt 1: Einrichtung der Entwicklungsumgebung

1. \*\*Installieren Sie Python:\*\* Stellen Sie sicher, dass Python auf Ihrem System installiert ist. Sie können es von [python.org](https://www.python.org/downloads/) herunterladen.

2. \*\*Installieren Sie Visual Studio Code:\*\* Falls noch nicht geschehen, installieren Sie Visual Studio Code (VS Code) von [code.visualstudio.com](https://code.visualstudio.com/).

3. \*\*Installieren Sie Python-Erweiterungen in VS Code:\*\* Öffnen Sie VS Code und installieren Sie die Python-Erweiterung über den Extensions-Marktplatz.

### Schritt 2: Projektinitialisierung

1. \*\*Erstellen Sie ein neues Verzeichnis für Ihr Projekt.\*\*

2. \*\*Öffnen Sie dieses Verzeichnis in VS Code.\*\*

3. \*\*Initialisieren Sie ein Git-Repository:\*\* Öffnen Sie das Terminal in VS Code und führen Sie `git init` aus.

4. \*\*Erstellen Sie eine virtuelle Umgebung:\*\* Führen Sie im Terminal `python -m venv venv` aus.

5. \*\*Aktivieren Sie die virtuelle Umgebung:\*\*

- Windows: `.\venv\Scripts\activate`

Hinweis: Unter Umständen lässt die Ausführungsrichtilinie von Windows diesen Befehl nicht zu. Lösung: temoräre Deaktivierung dieser Restriktion über PowerShell (Adminsitratorrechte notwendig):  
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

6. \*\*Erstellen Sie eine `requirements.txt` Datei für spätere Abhängigkeiten.\*\*  
🡪 nutzung von pip install pipreqs 🡪 pipreqs --force

### Schritt 3: Strukturierung des Projekts

1. \*\*Erstellen Sie Verzeichnisse für die verschiedenen Komponenten Ihrer App:\*\*

- `gui/` für die Benutzeroberfläche

- `logic/` für die Geschäftslogik

- `data/` für die Datenhaltung

- `tests/` für Unit-Tests

2. \*\*Erstellen Sie initiale Dateien:\*\* Zum Beispiel `main.py` als Einstiegspunkt Ihrer Anwendung.

### Schritt 4: Entwicklung der Benutzeroberfläche

1. \*\*Installieren Sie GUI-Frameworks:\*\* Zum Beispiel PyQt. Führen Sie `pip install PyQt5` aus.

2. \*\*Entwerfen Sie das GUI-Layout:\*\* Beginnen Sie mit dem Entwurf des Formulars für die Trainingsparameter.

3. \*\*Implementieren Sie die GUI-Logik:\*\* Verbinden Sie die GUI-Elemente mit Ereignishandlern.

### Schritt 5: Implementierung der Geschäftslogik

1. \*\*Trainingsplan-Generator:\*\* Schreiben Sie Funktionen, die auf Basis der Benutzereingaben Trainingspläne erstellen.

2. \*\*Exportfunktion:\*\* Implementieren Sie eine Funktion, um den Plan in Excel zu exportieren (z.B. mit `pandas`).

3. \*\*Sprachmanagement:\*\* Integrieren Sie eine Bibliothek wie Babel für Mehrsprachigkeit.

### Schritt 6: Datenhaltung

1. \*\*Wählen Sie eine geeignete Datenbank:\*\* Zum Beispiel SQLite.

2. \*\*Implementieren Sie das Datenmodell:\*\* Erstellen Sie Klassen oder Schemata für Ihre Daten.

### Schritt 7: Testing

1. \*\*Schreiben Sie Unit-Tests:\*\* Verwenden Sie ein Framework wie PyTest, um Ihre Funktionen zu testen.

### Schritt 8: Deployment-Vorbereitung

1. \*\*Erstellen Sie eine README-Datei:\*\* Dokumentieren Sie, wie Ihre Anwendung eingerichtet und verwendet wird.

2. \*\*Fügen Sie Ihre Abhängigkeiten in `requirements.txt` hinzu:\*\* Führen Sie `pip freeze > requirements.txt` aus.

### Schritt 9: Versionierung und Backup

1. \*\*Commiten Sie Ihre Änderungen in Git:\*\* Verwenden Sie Git, um Ihre Fortschritte zu verfolgen und zu sichern.

### Schritt 10: Iterative Entwicklung

1. \*\*Testen Sie Ihre Anwendung:\*\* Führen Sie die Anwendung aus und testen Sie sie iterativ.

2. \*\*Nehmen Sie Anpassungen vor:\*\* Basierend auf Tests und Feedback, passen Sie Ihre Anwendung an.

### Schritt 11: Dokumentation und Wartung

1. \*\*Dokumentieren Sie Ihre Codebasis:\*\* Stellen Sie sicher, dass Ihr Code gut kommentiert und dokumentiert ist.

2. \*\*Planen Sie Wartungsstrategien:\*\* Überlegen Sie, wie Sie Ihre App aktualisieren und warten werden.

Durch das Befolgen dieser Schritte können Sie systematisch eine robuste und funktionale Python-Anwendung entwickeln, die den Anforderungen Ihrer User Stories entspricht.