トレーニングAI分析アプリ システム設計書（初版）

*作成日: 2025-10-12 / 作成者: 章馬 吉田（個人開発）*

# 1. システム概要

|  |  |
| --- | --- |
| システム名 | トレーニングAI分析アプリ（Training AI Analytics） |
| 開発目的 | ユーザーのトレーニング履歴を記録・可視化し、AIが傾向分析・提案を行い継続的なパフォーマンス向上を支援する。 |
| 対象ユーザー | 筋トレ愛好者、パーソナルトレーナー、AI活用に関心のある個人 |
| 主要技術 | Python（FastAPI, Streamlit, pandas, SQLAlchemy, scikit-learn） |
| データベース | PostgreSQL（ローカル開発はSQLite） |
| インフラ/運用 | Docker（開発/デプロイの再現性）、RenderまたはRailwayにデプロイ |

# 2. 機能一覧

|  |  |  |
| --- | --- | --- |
| 機能カテゴリ | 機能名 | 概要 |
| ユーザー管理 | ユーザー登録・ログイン | FastAPIでJWT認証APIを提供（将来拡張含む） |
| トレーニング記録 | 記録登録 | 種目・重量・回数・セット数・メモを登録 |
| トレーニング記録 | 記録一覧表示 | 日付・種目・期間フィルタ／編集・削除 |
| データ可視化 | 推移グラフ | 種目ごとの重量・回数推移、頻度ヒートマップ、ボリューム集計 |
| データ分析 | 傾向分析 | 進捗・頻度・ボリューム・停滞検出 |
| AI提案 | トレーニングアドバイス | 分析結果をもとに自然言語で提案を生成 |
| 分析自動化 | 定期分析スクリプト | Cron/Airflowで日次・週次実行 |
| インフラ | Docker構成 | バックエンド・フロント・DBのコンテナ化 |

# 3. アーキテクチャ構成

ユーザー → Streamlit（フロント） → FastAPI（バックエンドAPI） → PostgreSQL（DB） → pandas/scikit-learn（分析） → OpenAI API（提案生成）

通信: REST/JSON, 認証: JWT（計画）, CORS設定有効化

# 4. データベース設計

## 4.1 users テーブル

|  |  |  |
| --- | --- | --- |
| カラム名 | 型 | 説明 |
| id | INTEGER (PK) | ユーザーID |
| username | VARCHAR(50) UNIQUE | ユーザー名 |
| email | VARCHAR(255) UNIQUE | メールアドレス |
| password\_hash | VARCHAR(255) | bcryptハッシュ |
| created\_at | TIMESTAMP | 登録日時（UTC） |

## 4.2 training\_records テーブル

|  |  |  |
| --- | --- | --- |
| カラム名 | 型 | 説明 |
| id | INTEGER (PK) | 記録ID |
| user\_id | INTEGER (FK -> users.id) | ユーザーID |
| date | DATE | 記録日 |
| exercise\_name | VARCHAR(100) | 種目 |
| weight | REAL | 重量(kg) |
| reps | INTEGER | 回数 |
| sets | INTEGER | セット数 |
| notes | TEXT | メモ |
| created\_at | TIMESTAMP | 登録日時（UTC） |

## 4.3 analysis\_results テーブル

|  |  |  |
| --- | --- | --- |
| カラム名 | 型 | 説明 |
| id | INTEGER (PK) | 分析ID |
| user\_id | INTEGER (FK -> users.id) | 対象ユーザー |
| avg\_weight | REAL | 平均重量 |
| total\_volume | REAL | 総ボリューム |
| weak\_point | VARCHAR(50) | 弱点部位 |
| recommendation | TEXT | AIによる提案内容 |
| analyzed\_at | TIMESTAMP | 分析日時（UTC） |

# 5. 画面設計（Streamlit）

|  |  |  |
| --- | --- | --- |
| 画面名 | 機能 | 主要要素 |
| トップ画面 | 概要とナビゲーション | 「記録する」「分析を見る」ボタン |
| 記録画面 | フォーム入力 | 日付, 種目, 重量, 回数, セット数, メモ, 登録ボタン |
| 分析画面 | グラフ＋提案 | 推移グラフ, ボリューム集計, AIコメント |
| 履歴一覧 | テーブル表示 | 検索/フィルタ, 編集・削除 |

# 6. API設計（FastAPI）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| エンドポイント | メソッド | 機能 | リクエスト例 | レスポンス例 |
| /records/ | POST | 記録登録 | {"exercise\_name":"Bench Press","weight":60,"reps":8,"sets":3} | {"message":"Record saved"} |
| /records/ | GET | 全記録取得 | - | [{"id":1,"exercise\_name":"Bench Press","weight":60}] |
| /analysis/ | GET | 最新分析結果 | - | {"total\_volume":12345,"weak\_point":"Legs"} |
| /ai/recommend | POST | AI提案生成 | {"user\_id":1} | {"recommendation":"Increase squat frequency."} |

# 7. 非機能要件

**パフォーマンス:** 1万件の記録データでも1秒以内のレスポンスを目標

**セキュリティ:** bcryptでパスワードハッシュ化、JWT、CORS、入力バリデーション

**可用性:** Dockerにより開発・本番の再現性を担保

**拡張性:** LINE通知、クラウド連携、ユーザー別学習などを容易に追加

# 8. 実装ロードマップ

・Step1: Streamlitでフォームと基本グラフを作成

・Step2: FastAPI + DB（SQLAlchemy）でCRUD API実装

・Step3: pandasでボリューム・傾向分析

・Step4: scikit-learnで回帰/クラスタリングの軽量モデル

・Step5: OpenAI APIで自然言語提案を統合

・Step6: Docker化＆Render/Railwayへデプロイ

# 9. 付録：ディレクトリ構成（案）

training-ai-app/  
├── backend\_fastapi/  
│ ├── main.py  
│ ├── routers/  
│ │ ├── users.py  
│ │ ├── records.py  
│ │ └── analysis.py  
│ ├── models/  
│ │ ├── user.py  
│ │ └── record.py  
│ ├── db.py  
│ ├── schemas.py  
│ └── ai/  
│ └── recommender.py  
├── frontend\_streamlit/  
│ ├── app.py  
│ └── components/  
│ └── charts.py  
├── data/  
│ └── training\_logs.csv  
└── docker-compose.yml