詳細設計書

# 1. システム概要

## 1.1 アーキテクチャ

フロントエンド: HTML, CSS, JavaScript, Thymeleaf

バックエンド: Java, Spring Boot, Spring Security, JPA/Hibernate

データベース: MySQL

インフラ: AWSまたはHeroku、GitHub Actionsによる自動デプロイ

# 2. フロントエンド設計

## 2.1 ユーザーインターフェース

### 2.1.1 HTML/CSS

レイアウト: FlexboxおよびGridを使用してレスポンシブデザインを実現

スタイリング: Bootstrapを使用し、カスタムCSSでデザイン調整

### 2.1.2 JavaScript

フォームバリデーション: JavaScriptによるクライアントサイドバリデーション

AJAX通信: REST APIとの非同期通信に使用

### 2.1.3 Thymeleaf

テンプレートエンジン: サーバーサイドでHTMLを生成し、動的コンテンツを表示

## 2.2 画面設計

### 2.2.1 トップページ

機能: 最新のトレーニング記録の表示、ナビゲーションメニュー

要素: トレーニングの一覧、フィルター機能、進捗バー

### 2.2.2 ユーザー登録/ログイン

機能: ユーザー登録、ログイン

要素: フォーム入力、エラーメッセージ表示

### 2.2.3 トレーニング記録

機能: トレーニング記録の追加、編集、削除

要素: トレーニング種目選択、記録入力フォーム、保存ボタン

### 2.2.4 トレーニング種目管理

機能: トレーニング種目の追加、編集、削除

要素: フォーム入力、エラーメッセージ表示、保存ボタン

### 2.2.5 進捗管理

機能: グラフやチャートでの進捗表示、目標設定

要素: グラフ表示、目標入力フォーム

### 2.2.6 エクスポート

機能: トレーニングデータのCSVおよびPDF形式でのエクスポート

要素: エクスポートボタン、ダウンロードリンク

# 3. バックエンド設計

## 3.1 REST API

### 3.1.1 エンドポイント

#### ユーザー管理

POST /api/users/register: ユーザー登録

POST /api/users/login: ログイン

GET /api/users/{id}: ユーザー情報取得

PUT /api/users/{id}: ユーザー情報更新

#### トレーニング記録

POST /api/exercises: トレーニング種目の追加

GET /api/exercises/{id}: トレーニング種目取得

PUT /api/exercises/{id}: トレーニング種目の更新

DELETE /api/exercises/{id}: トレーニング種目の削除

#### セッション記録

POST /api/sessions: トレーニングセッションの追加

GET /api/sessions/{id}: トレーニングセッションの取得

PUT /api/sessions/{id}: トレーニングセッションの更新

DELETE /api/sessions/{id}: トレーニングセッションの削除

#### セッションセット記録

POST /api/sessions/{session\_id}/sets: セッションセットの追加

GET /api/sessions/{session\_id}/sets: セッションセットの取得

PUT /api/sessions/{session\_id}/sets/{set\_id}: セッションセットの更新

DELETE /api/sessions/{session\_id}/sets/{set\_id}: セッションセットの削除

#### 進捗管理

GET /api/progress/{userId}: 進捗データ取得

POST /api/goals: 目標設定

GET /api/goals/{userId}: 目標取得

#### エクスポート

GET /api/export/csv: CSVエクスポート

GET /api/export/pdf: PDFエクスポート

## 3.2 データベース設計

### 3.2.1 テーブル設計

#### ユーザー (users)

id: INT, PK, AUTO\_INCREMENT

name: VARCHAR(50), UNIQUE

email: VARCHAR(255), UNIQUE

password: VARCHAR(255)

family\_name: VARCHAR(50)

first\_name: VARCHAR(50)

age: INT

weight: DECIMAL(5,2)

height: DECIMAL(5,2)

gender: INT

profile\_image: TEXT

created\_at: DATETIME

updated\_at: DATETIME

#### トレーニング種目 (exercises)

id: INT, PK, AUTO\_INCREMENT

name: VARCHAR(50), UNIQUE

main\_part: VARCHAR(50)

sub\_part: VARCHAR(50)

profile\_image: TEXT

created\_at: DATETIME

updated\_at: DATETIME

#### セッション (sessions)

id: INT, PK, AUTO\_INCREMENT

exercise\_id: INT, FK (exercises.id)

user\_id: INT, FK (users.id)

session\_date: DATE

created\_at: DATETIME

updated\_at: DATETIME

#### セッションセット (session\_sets)

id: INT, PK, AUTO\_INCREMENT

session\_id: INT, FK (sessions.id)

weight: DECIMAL(5,2)

reps: INT

set\_number: INT

created\_at: DATETIME

updated\_at: DATETIME

## 3.3 セキュリティ

### 3.3.1 認証と認可

Spring Security: ユーザー認証とロールベースのアクセス制御

JWT: JSON Web Tokenを使用したセッション管理

### 3.3.2 データ保護

パスワードのハッシュ化: BCryptを使用してパスワードをハッシュ化

データ暗号化: 機密データはAESなどの暗号化アルゴリズムで保護

# 4. インフラ設計

## 4.1 デプロイメント

AWS: EC2, RDS, S3を使用してホスティング

Heroku: アプリケーションの簡易デプロイメント

GitHub Actions: CI/CDパイプラインによる自動デプロイ

## 4.2 スケーラビリティ

ロードバランサー: 複数のインスタンスにトラフィックを分散

データベースのスケーリング: RDSのリードレプリカを使用して読み取り負荷を分散

# 5. テスト計画

## 5.1 単体テスト

JUnit: 各コンポーネントの単体テストを実施

Mockito: モックを使用した依存関係のテスト

## 5.2 結合テスト

Spring Boot Test: アプリケーションの統合テスト

## 5.3 UIテスト

Selenium: フロントエンドの自動テストを実施

# 6. ドキュメント

## 6.1 開発者向けドキュメント

API仕様書: REST APIのエンドポイントと使用方法

データベース設計書: テーブル構造とリレーションの詳細

## 6.2 ユーザーマニュアル

使用ガイド: アプリケーションの基本操作方法とFAQ