Web プログラミング課題 仕様書レポート

学籍番号 氏名

2025 年 12 月 27 日

1　利用者向け仕様書

本セクションでは，開発した 3 つの Web アプリケーションのうち，代表的なシステムである「サーモンラン敵マニュアル」について，その機能と利用方法を解説する．

1.1　システム概要

「サーモンラン敵マニュアル」は，ゲーム『スプラトゥーン』の協力モード「サーモンラン」に登場する敵キャラクター（シャケ）の情報を管理・閲覧するための Web アプリケーションである．利用者は，敵キャラクターの名前，種類，攻撃方法，討伐方法などの詳細情報を確認できるほか，新しい敵情報の登録や，既存情報の修正・削除を行うことができる．

1.2　主な機能

本システムは以下の機能を提供する．

1. **一覧表示機能**: 登録されている敵キャラクターの一覧を表示する．各キャラクターのリンクをクリックすることで詳細画面へ遷移できる．
2. **詳細表示機能**: 特定の敵キャラクターに関する詳細な情報（種類，攻撃方法，正規討伐方法，別途討伐方法など）を表示する．
3. **新規登録機能**: 新しい敵キャラクターの情報をシステムに追加する．
4. **編集機能**: 既存の敵キャラクターの情報を変更する．
5. **削除機能**: 不要になった敵キャラクターの情報をシステムから削除する．

1.3　操作方法

1.3.1　情報の閲覧

トップページ（一覧画面）には，現在登録されている敵キャラクターの ID と名前がリスト形式で表示されている．詳細を知りたいキャラクターの名前（リンク）を選択すると，そのキャラクターの「詳細画面」が表示される．

1.3.2　情報の追加

一覧画面にある「追加」リンクを選択すると，「新規登録画面」へ遷移する．入力フォームに以下の項目を入力し，「登録」ボタンを押下することで，新しいデータが保存され一覧画面に反映される．

- 名前
- 種類（雑魚シャケ，オオモノシャケなど）
- 攻撃方法
- 正規討伐方法
- 別途討伐方法

1.3.3　情報の編集・削除

「詳細画面」の下部にある「編集」リンクを選択すると，登録内容を変更できる．また，「削除」リンクを選択すると，確認ダイアログが表示され，承認するとそのデータが削除される．

2　管理者向け仕様書

本セクションでは，本サービス全体（3 つのシステムを含む）の管理方針とシステムの全体像について記述する．

2.1　サービス全体概要

本サービスは，特定のテーマに基づいたデータ管理機能（CRUD）を提供する 3 つの独立した Web アプリケーション群で構成されている．

1. **サーモンラン敵マニュアル**: ゲームキャラクターのデータ管理
2. **第五人格ハンター一覧**: ゲームキャラクターのデータ管理
3. **D.LEAGUE 参画チーム一覧**: ダンスリーグチームのデータ管理

これらは共通の設計思想に基づき実装されており，管理者は統一された操作感で各データをメンテナンスすることが可能である．

2.2 データの管理と制約

本システムは学習用プロトタイプとして設計されているため，以下の運用上の制約が存在する．

- **データ永続性の欠如**: データベースを使用せず，サーバーのメモリ上（変数）にデータを保存している．そのため，**サーバープログラムを再起動または停止すると，追加・編集・削除したデータはすべて初期状態にリセットされる**．
- **同時アクセス**: 簡易的な実装であるため，多数のユーザーによる同時書き込みが発生した場合の排他制御は行われていない．

3 開発者向け仕様書

本セクションでは，実装された 3 つのシステムそれぞれの技術仕様について記述する．全てのシステムは Node.js 環境上で動作し，Web フレームワークとして Express，テンプレートエンジンとして EJS を使用している．また，データストアはオンメモリの配列変数を使用している．

3.1 共通仕様

すべてのシステムにおいて，以下の設計方針が統一されている．

- **アーキテクチャ**: MVC モデル（Model は配列変数で代用）に準じた構成．
- **HTTP メソッド**: GET（表示，フォーム取得），POST（登録，更新）を使用．削除は簡易的に GET メソッドで実装されている．
- **JavaScript モード**: `"use strict";` モードを使用．

3.2 システム A: サーモンラン敵マニュアル

3.2.1 データ構造

[cite$_s tart$] データはサーバーサイドの変数 `samochara`（配列）に格納される．各要素は以下のオブジェクト構造を持つ [$cite$: 3, 4].

3.2.2 リソースとエンドポイント

- **一覧表示**: `GET /samoran`
  `samoran.ejs` をレンダリングし，全データをリスト表示する．
- **新規登録フォーム**: `GET /samoran/create`
  静的ファイル `public/samoran_new.html` へリダイレクトする．
- **詳細表示**: `GET /samoran/:number`

配列のインデックス :number に対応するデータを samoran_detail.ejs で表示する．

- **新規登録実行**: POST /samoran
  フォームデータを配列に追加し，一覧画面を再レンダリングする．ID は配列長に基づき自動採番される．
- **編集フォーム**: GET /samoran/edit/:number
  対象データを埋め込んだ状態で samoran_edit.ejs を表示する．
- **更新実行**: POST /samoran/update/:number
  配列の該当インデックスのデータをフォームの内容で上書きし，一覧画面へリダイレクトする．
- **削除実行**: GET /samoran/delete/:number
  配列から該当要素を splice で削除し，一覧画面へリダイレクトする．

3.3 システム B: 第五人格ハンター一覧

3.3.1 データ構造

[cite$_s tart$] データは変数 hunter（配列）に格納される．各要素の構造は以下の通りである [$cite$: 7, 8].

3.3.2 リソースとエンドポイント

基本構造はシステム A と同様であるが，ベース URL が /daigo となる．

- 一覧: GET /daigo
- 詳細: GET /daigo/:number
- 登録: POST /daigo
- 更新: POST /daigo/update/:number
- 削除: GET /daigo/delete/:number

3.4 システム C: D.LEAGUE 参画チーム一覧

3.4.1 データ構造

[cite$_s tart$] データは変数 teams（配列）に格納される．各要素の構造は以下の通りである [$cite$: 11, 12].

3.4.2 リソースとエンドポイント

基本構造は他システムと同様であり，ベース URL は /dleague となる．

- 一覧: GET /dleague

- 詳細: `GET /dleague/:number` （詳細画面では `url` プロパティを利用した外部リンクが生成される）
- 登録: `POST /dleague`
- 更新: `POST /dleague/update/:number`
- 削除: `GET /dleague/delete/:number`