詳細仕様書(内部設計)

1.ログイン機能の詳細設計

1.1.　概要

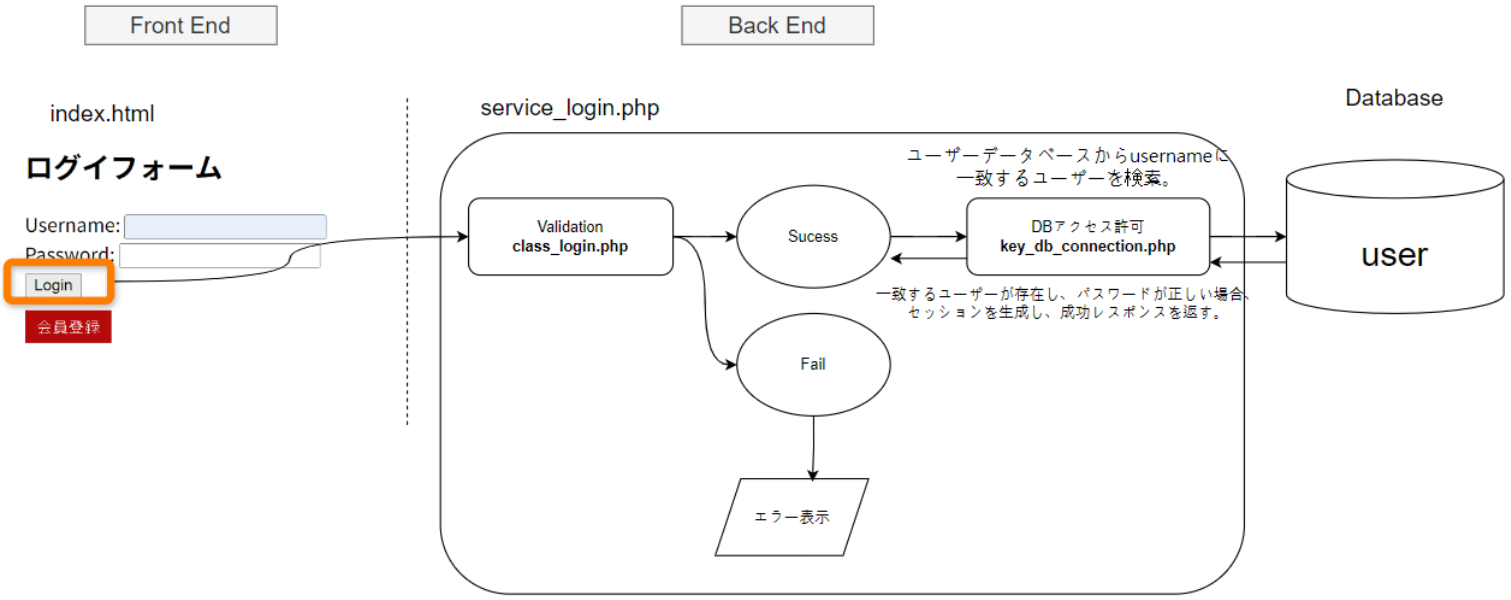
ユーザーがシステムにログインするための機能を提供する。ユーザーはメールアドレスとパスワードを入力し、認証が成功した場合にシステムにログインできる。

1.2. 入力仕様

- **username**：文字制限5文字以上、10文字以下。英数字のみ
    - 文字制限はregular expressionにより　`'/^[a-zA-Z0-9]+$/';`
- **password**：文字制限5文字以上、10文字以下。英数字のみ
    - 文字制限はregular expressionにより　`'/^[a-zA-Z0-9]+$/';`

1.3. 処理概要

1. フロントエンドでusernameとpasswordを入力するフォームを提供。
2. フォームが送信されると、バックエンドにリクエストを送信。
3. バックエンドで以下の処理を行う：
    - 入力されたusernameとpasswordの検証。(Validation)
    - ユーザーデータベースからusernameに一致するユーザーを検索。
    - 一致するユーザーが存在し、パスワードが正しい場合、セッションを生成し、成功レスポンスを返す。
    - 認証失敗の場合、エラーメッセージを返す。



1.4. エラーハンドリング

- 入力形式エラー：
    - username かpasswordかいずれか間違った場合：The Username or Password is Incorrect.

2.会員登録機能の詳細設計

2.1. 概要

新規ユーザーがシステムに登録するための機能を提供する。ユーザーは必要な情報を入力し、登録に成功するとシステムにアクセスできるようになる。

2.2. 入力仕様

- **username**：文字制限5文字以上、10文字以下。英数字のみ
    - 文字制限はregular expressionにより `'/^[a-zA-Z0-9]+$/';`
- **password**：文字制限5文字以上、10文字以下。英数字のみ
    - 文字制限はregular expressionにより `'/^[a-zA-Z0-9]+$/';`

2.3. 処理概要

1. フロントエンドで username、password を入力するフォームを提供。
2. フォームが送信されると、バックエンドにリクエストを送信。
3. バックエンドで以下の処理を行う：
    - 入力されたusernameとpasswordの検証。(Validation)
    - ユーザーデータベースに username が既に存在するか確認。
    - username が重複しない場合、パスワードをハッシュ化してデータベースに保存。
    - 登録成功の場合、成功レスポンスを返す。
    - 登録失敗またはバリデーションエラーの場合、エラーメッセージを返す。
4. フロントの方でログインできるようにする。

2.4. エラーハンドリング

- 入力形式エラー：
    - username 長さが満足できなかった場合：ユーザー名パスワードは5文字以上10以下である必要があります
    - username 文字コードが満足できなかった場合：ユーザー名には英数字のみが使用できます
    - password が長さが満足できなかった場合：パスワードは5文字以上10以下である必要があります
    - password 文字コードが満足できなかった場合：パスワードには英数字のみが使用できます

3.ログアウト機能の詳細設計

3.1. 概要

ユーザーがシステムからログアウトするための機能を提供する。ユーザーがログアウトすると、セッションが破棄され、システムからログアウトされる。

3.2. 入力仕様

- 特になし

3.3. 処理概要

1. フロントエンドでログアウトボタンを提供。
2. ユーザーがログアウトボタンをクリックすると、バックエンドにリクエストを送信。
3. バックエンドで以下の処理を行う：
    - 現在のセッションを破棄。

- ログアウト成功のレスポンスを返す。

4.データベース設計

Usersテーブル

ユーザーの情報を保管しているテーブル

- id : 主キー
- username：ユーザーID
- password：パスワード
- password_hash:ハッシュ化されたパスワード

Sessionテーブル

Sessionテーブルは、ウェブアプリケーションでセッションを管理するためのテーブル。各セッションは固有のセッションIDを持ち、セッションに関連するデータと作成日時が保存さる。

- id : 主キー
- session_id：各セッションの固有の識別子であり、セッションを識別するための文字列形式の値
- data：{username: xx}で保存される
- created_at：セッションレコードが作成された日時を示すフィールド

5. セキュリティ対策

- パスワードはハッシュ化（bcrypt）して保存。
  - `$password = password_hash($_POST['password'], PASSWORD_BCRYPT);`
- SQL Injection防止
  - SQL Injectionとは
    - 攻撃者がアプリケーションのデータベースに対して不正なSQLクエリを挿入することで、データベースの操作やデータの窃取を試みる攻撃手法
  - 対応方法
    - SQLをPrepared Statementsの形で作ってからそれを実行するようにする

- データベースと連携するPHPファイルをユーザーがアクセスできない場所へ保管する
    - 今回はしない
- ログイン試行回数の制限（例：一定回数失敗したらアカウントを一時的にロック）
    - 今回はしない