**システム全体の概要**

• **共通認証:**

すべてのサービス（MUSESを含む）は最初に共通のサインイン処理を行います。認証情報の入力、暗号化・保存、ログイン処理などは共通モジュールとして実装し、各サービスへのアクセス前に必ず実行されます。

• **サービス毎の処理:**

サインイン後は各サービス（MUSES、他の基幹システム、その他スクレイピングツールなど）ごとに異なるUI・処理が必要になります。サービス毎に独立したモジュールを用意して、スクレイピングや各種操作のロジックを分離します。

• **メインメニュー:**

将来的には、サインイン後にユーザーへメニュー画面を提示し、どのサービスのどの機能を実行するかを選択できる仕組みを導入します。メインメニューは、各サービスモジュールのエントリーポイントを呼び出す役割を担います。

―――――

**モジュール構成**

**1. 共通モジュール (Common)**

• **認証モジュール (Authentication):**

サインイン処理（ログイン画面の操作、認証情報の管理・暗号化、共通エラー処理など）

• **ユーティリティ (Utils):**

ログ出力、例外処理、共通ライブラリのラッパーなど、各モジュールで利用する汎用関数

**2. サービスモジュール (Services)**

• **MUSESモジュール:**

MUSESへの遷移、画面操作、スクレイピング処理など、MUSES特有の処理を実装

• **その他サービスモジュール:**

他のサービスごとに、必要な処理（各サービスのUI操作、データ取得、解析など）を個別モジュールとして実装

**3. メニュー・コントローラ (Main Menu / Controller)**

• サインイン後に起動するメインメニューのアプリケーション。

ユーザーがどのサービスに対してどの処理を行うか選択できるようにし、選択に応じて各サービスモジュールのエントリーポイントを呼び出す。

グラフィカル ユーザー インターフェイス, アプリケーション, テーブル

AI によって生成されたコンテンツは間違っている可能性があります。

**補足**

• **共通認証モジュール:**

例えば common/auth/signin\_module.py にサインイン処理を実装し、どのサービスモジュールからも最初にこれを呼び出すようにします。認証情報の管理方法（暗号化ファイルなど）はここで一元管理します。

• **サービスごとのモジュール:**

サービス固有の操作は、各サービスモジュール内に個別に実装し、例えば MUSES であれば services/muses/muses\_navigate.py に MUSESへの遷移・操作ロジックを実装します。

各サービスのUIが異なるため、サービス間でコードの再利用は難しい部分はそれぞれ独自に実装し、共通部分のみを common/utils/ で管理します。

• **メインメニュー:**

menu/main\_menu.py では、サインイン済みの WebDriver を受け取り、ユーザーに対してどのサービスのどの処理を実行するか選択肢を提示します。選択結果に応じて、対応するサービスモジュールの関数を呼び出す設計とします。

• **拡張性:**

サービス追加時は、services/ 配下に新しいフォルダを作成し、その中に新規モジュールを実装するだけで済むようにすることで、将来的な拡張やメンテナンスを容易にします。