楽譜管理ソフトウェア
内部設計書 v1

dir-en-gray

2023 年 12 月 10 日

第 1 章

システム概要

1.1 ユーザ側システム一覧

- 基本機能

  -**ユーザ登録機能**

  新規のユーザを登録する機能.

  -**ログイン機能**

  既存のユーザがログインする機能.

  -**ログアウト機能**

  ユーザがログイン状態でなくなる機能.

  -**ユーザ情報編集機能**

  ユーザが自分の情報を編集する機能．ユーザ名・メールアドレス・パスワードを変更できる.

  -**ユーザ削除機能**

  ユーザが自分の情報を削除する機能．DB からも情報が削除される.

- 楽譜データに関する機能

  -**楽譜データ登録機能**

  新規の楽譜データを登録する機能．曲名・作曲者・編曲者・難易度・演奏時間・使用楽器を登録できる.

  -**楽譜データ編集機能**

  既存の楽譜データを編集する機能．曲名・作曲者・編曲者・難易度・演奏時間・使用楽器を変更できる.

  -**楽譜データ削除機能**

  既存の楽譜データを削除する機能．DB からも情報が削除される.

  -**楽譜データ一覧閲覧機能**

  既存の楽譜データを一覧形式で閲覧する機能．通常，この一覧はログイン後のページに表示される.

  -**楽譜データ詳細閲覧機能**

  既存の楽譜データの詳細を閲覧する機能．個別の楽譜ごとに曲名・作曲者・編曲者・難

易度・演奏時間・使用楽器の情報を閲覧できる．

- 楽譜検索機能

  -作曲者，編曲者からの検索機能

  検索ボックスに入力された，作曲者・（または）編曲者から楽譜を検索する機能．入力文字が’含まれる’楽譜を取得する．

  -曲名からの検索機能

  検索ボックスに入力された，曲名から楽譜を検索する機能．入力文字が’含まれる’楽譜を取得する．

  -グレード（難易度）からの検索機能

  プルダウンで選択された，グレードから楽譜を検索する機能．

  -使用楽器からの検索機能

  プルダウンで選択された，使用楽器から楽譜を検索する機能．複数選択が可能である．

- 楽譜ソート機能

  -曲名から，五十音順にソートする機能

  楽譜データを，曲名から五十音順にソートする機能．その他のスキーマについては考慮されない．

  -作曲者，編曲者から，五十音順にソートする機能

  楽譜データを，作曲者，または編曲者から五十音順にソートする機能．その他のスキーマについては考慮されない．

  -グレード順にソートする機能

  楽譜データを，グレード[*1]から，昇順にソートする機能．その他のスキーマについては考慮されない．

  -演奏時間順にソートする機能

  楽譜データを，演奏時間の短い順にソートする機能．その他のスキーマについては考慮されない．

- お問い合わせ画面表示機能

  -お問い合わせ機能

  お問い合わせを受け付ける機能．ただし問い合わせ内容をユーザ情報と共に DB に保存等の操作はなく，指定したメールアドレスに直接連絡することを想定する．

- 広告表示機能

  -広告表示機能

  広告を表示する機能．全てのページのヘッダに共通して表示する．

[*1] グレードは 1〜5 までの整数値である．本来小数点以下も存在するが，四捨五入した値とする（1.5 なら 2 とする）

1.2 管理者側システム一覧

- 基本機能

  -ログイン機能

  管理者としてログインする機能.

  -ログアウト機能

  管理者がログイン状態でなくなる機能.

  -管理者情報編集機能

  管理者情報を編集する機能．管理者名・メールアドレス・パスワードを変更できる.

- ユーザに対する機能

  -ユーザデーター覧閲覧機能

  ユーザデータを一覧形式で閲覧する機能．通常，この一覧はログイン後のページに表示される.

  -ユーザデータ詳細閲覧機能

  ユーザデータの詳細を閲覧する機能．ユーザ名・メールアドレスが閲覧できる.

  -ユーザ情報編集機能

  ユーザ情報を編集する機能．ユーザ名・メールアドレスを編集できる．通常行わない.

  -ユーザ削除機能

  ユーザデータを削除する機能．DB からも完全に削除する.

  -楽譜データ削除機能

  ユーザデータに紐づいた楽譜データを削除する機能．ユーザを削除すると自動的に呼び出される.

- ユーザ検索機能

  -ユーザ名から検索する機能

  検索ボックスに入力された，ユーザ名からユーザを検索する機能．入力文字が’含まれる’ユーザを取得する.

  -ユーザ ID から検索する機能

  検索ボックスに入力された，ユーザ ID からユーザを検索する機能．入力文字が’含まれる’ユーザを取得する.

- ユーザソート機能

  -ユーザ名を五十音順にソートする機能

  ユーザ名から五十音順にソートする機能.

- 楽譜検索機能

  -作曲者，編曲者からの検索機能

  検索ボックスに入力された，作曲者・（または）編曲者から楽譜を検索する機能．入力文字が’含まれる’楽譜を取得する.

-曲名からの検索機能

検索ボックスに入力された，曲名から楽譜を検索する機能．入力文字が’含まれる’楽譜を取得する．

-グレード（難易度）からの検索機能

プルダウンで選択された，グレードから楽譜を検索する機能．

-使用楽器からの検索機能

プルダウンで選択された，使用楽器から楽譜を検索する機能．複数選択が可能である．

- 楽譜ソート機能

-曲名から，五十音順にソートする機能

楽譜データを，曲名から五十音順にソートする機能．その他のスキーマについては考慮されない．

-作曲者，編曲者から，五十音順にソートする機能

楽譜データを，作曲者，または編曲者から五十音順にソートする機能．その他のスキーマについては考慮されない．

-グレード順にソートする機能

楽譜データを，グレード[*1]から，昇順にソートする機能．その他のスキーマについては考慮されない．

-演奏時間順にソートする機能

楽譜データを，演奏時間の短い順にソートする機能．その他のスキーマについては考慮されない．

- 広告登録機能

-広告登録機能

広告を登録する機能．UI は持たず，コンソール上で命名規則に従ったファイルを GitHub にアップロードすることで登録する．

---

*1

第 2 章

システム実装方法

動作環境

ユーザ、管理者共に Javascript 使用に対応したブラウザ

開発環境

OS	Windows, Mac, Linux
開発フレームワーク	Ruby on Rails バージョン 7 以降
Ruby バージョン	3.2.2
開発言語	Ruby, HTML, ERB, SCSS
DBMS	SQLite
サーバ	AWS
IDE	Visual Studio Code
バージョン管理	Git, GitHub

第 3 章

規約

3.1 コーディング規約

ここでは，コーディングに関する規約を定める．変数名，ファイル名については，3.2 節で述べる．ここではコード整形についてのみ言及する．

コード整形には，`rubocop` というツールを用いる，rubocop が定めるルールに従う．rubocop の設定ファイルを以下に示す．

```yaml
inherit_from: .rubocop_todo.yml

require:
- rubocop-rails
- rubocop-rspec
- rubocop-performance

AllCops:
NewCops: enable
SuggestExtensions: false

Style/Documentation:
Enabled: false

Rails/I18nLocaleTexts:
Enabled: false

Metrics/AbcSize:
Max: 30

Metrics/PerceivedComplexity:
Max: 10

# ブロックの長さ制限
Metrics/BlockLength:
Max: 20
Exclude:
- 'config/environments/development.rb'

# メソッド内のながさ制限
Metrics/MethodLength:
Max: 20

# selfをつけるかどうか
Style/RedundantSelf:
Exclude:
- 'app/models/user.rb'

# validation をスキップする警告を無視
Rails/SkipsModelValidations:
Exclude:
- 'app/models/user.rb'
- 'test/helpers/sessions_helper_test.rb'
```

図 3.1: rubocop 設定ファイル

3.2 Rails フレームワーク規約

Rails のフレームワークに従って実装する．Rails は MVC モデルを採用しているため，それぞれに命名は Rails が定める規則に従う．

- Model
  1. ファイル
     モデルファイルは，app/model ディレクトリ内に配置する．モデルファイル名は，tableName.rb とする．tableName には作るデータテーブル名を単数形・小文字で書く．例えば，User に関するデータテーブルを作りたい場合には，user.rb とする．
  2. クラス
     1 つのファイルに，1 つのクラスを記述する．クラス名は，ファイル名の先頭を大文字にしたものを使う．例えば，ファイル名が user.rb である場合，クラス名は User である．必ず，ApplicationRecord を継承する．
  3. マイグレーションファイル
     a. ファイル
        マイグレーションファイルは，db/migrate ディレクトリ内に配置する．ファイル名は timestamp_verb.rb とする．verb には動作を記入，ファイル名はスネークケースを用いて表現し，全て小文字とする．例えば User データベース内の Email 列にインデックスを追加するマイグレーションファイル名は，timestamp_add_index_to_users_email.rb である．
     b. クラス
        1 つのファイルに, 1 つのクラスを記述する．クラス名は, ファイル名の timestamp と拡張子を除いてスネークケースで表現したものを用いる．上記の例に倣うと，クラス名は AddIndexToUsersEmail となる．
        必ず，ActiveRecord::Migration[7.1] を継承する．
     c. 関数名
        関数名は change とする．1 つのクラスに 1 つの関数のみを定義する．
- Controller
  1. ファイル
     コントローラファイルは app/controllers ディレクトリ内に配置する．コントローラのファイル名は，viewContents_controller.rb とする．viewContents は表示したいものを複数形・小文字で書く．例えば，User に関するコントローラの場合は，users_controller.rb とする．
  2. クラス
     1 つのファイルに，1 つのクラスを記述する．クラス名はキャメルケースを用いて表現

する．必ず，ApplicationController を継承する．例えば，
users_controller.rb 内で定義されているクラス名は，UsersController である．

3. アクション
コントローラ内のアクション名は，全て小文字で表現する．原則，以下のアクション名を用いる．これ以外に定義する場合，全体で相談する．
   - index
   - home
   - new
   - create
   - edit
   - show
   - update
   - destroy

- View
  1. ファイル
  ビューファイルは，app/view ディレクトリ内に配置する．ディレクトリ内にサブディレクトリを”コントローラ名”で作り，その中に各アクションに対応するビューを作る．例えば，users_controller.rb 内で定義されたアクション new に対応するビューを作るなら，そのファイル名は app/views/users/new.html.erb となる．

3.3 開発規約

ソースコードの差分管理には GitHub を用いる．差分ログを GitHub 上へ反映することを「PUSH」と書く．また，CI を GitHub Actions で実行する．内容は，rubocop と，rails test である．

3.3.1 PUSH する前に

PUSH する前に，テストを走らせ，rubocop を実行する．全て成功したら PUSH する．

```
$ bundle exec rubocop
$ bundle exec rails test
```

3.3.2 ブランチ

- メインブラインチを develop とする．
- develop ブランチへ PUSH しないこと．必ずブランチを切って編集し，そのブランチ名で

PUSH する．以下の項目が成功したら `develop` へマージする．

1. GitHub 上でプルリクエストを作成する．
2. CI が通っていることを確認する．
3. 溝口洸熙が Approve する．

- ブランチ名は以下のように定める．
  - `docs/*`
    ドキュメントに関する更新
  - `fix/*`
    バグの更新
  - `future/*`
    新規機能の追加
- 他ブランチ上で作業する時は，定期的に以下のコマンドを実行し，`develop` の更新を取り入れる．

```
$ git merge develop
```

3.3.3 タグ

リリース作業には git Tag を用いる．タグの命名は以下のように行う．

```
v[major-Version].[minor-Version]
```

開始番号を `0` とし，バグ修正を minor-Version Up，新機能追加を major-Version Up とする．タグの更新を GitHub Actions で検知し，AWS へオートデプロイ（CD）する．

第 4 章

モジュール設計

4.1 前提

我々は Ruby on rails を用いて開発を行う．この言語の慣習に則り，これ以降，

- index
- new
- show
- edit
- create
- destroy
- home

の 7 つは「アクション」と呼び，これ以外の関数を全て「メソッド」と呼ぶ.

アクションはコントローラが異なれば同名のものを使用するため，モジュール ID を「Controller 名. アクション名」とする．また，1 アクション 1 機能を持つ.

4.2 モジュール詳細

以下に，各モジュールについて「定義書」「フロー図」の順で示す.

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/2
	作成者	奥平舜理	更新日	2023/12/5
基本情報	概要	UsersController 内の index アクション		
	所属クラス	UsersController		
	モジュール名	index		
	モジュールID	UsersController.index		

処理説明

【処理内容】
ユーザの一覧を表示する.
【処理手順】
1. ユーザか管理者かの判定を行い，管理者でない場合は ScoresController.home を呼び出す.
2. User クラスの all メソッドを用いてデータベースからユーザの一覧を取得し，インスタンス変数 @users に格納する.
3. views/users/index.html.erb を描画する.
【補足】

入力値説明

なし

出力値説明

ユーザ一覧

他クラス・関数との関係

ApplicationController クラスを継承する.

図 4.1: UsersController.index 定義書

所属クラス	UsersController	作成日	2023/12/08
機能名	ユーザー一覧表示	更新日	2023/12/08
モジュールID	UsersController.index	作成者	奥平舜理
使用モジュールID	UsersController.home		

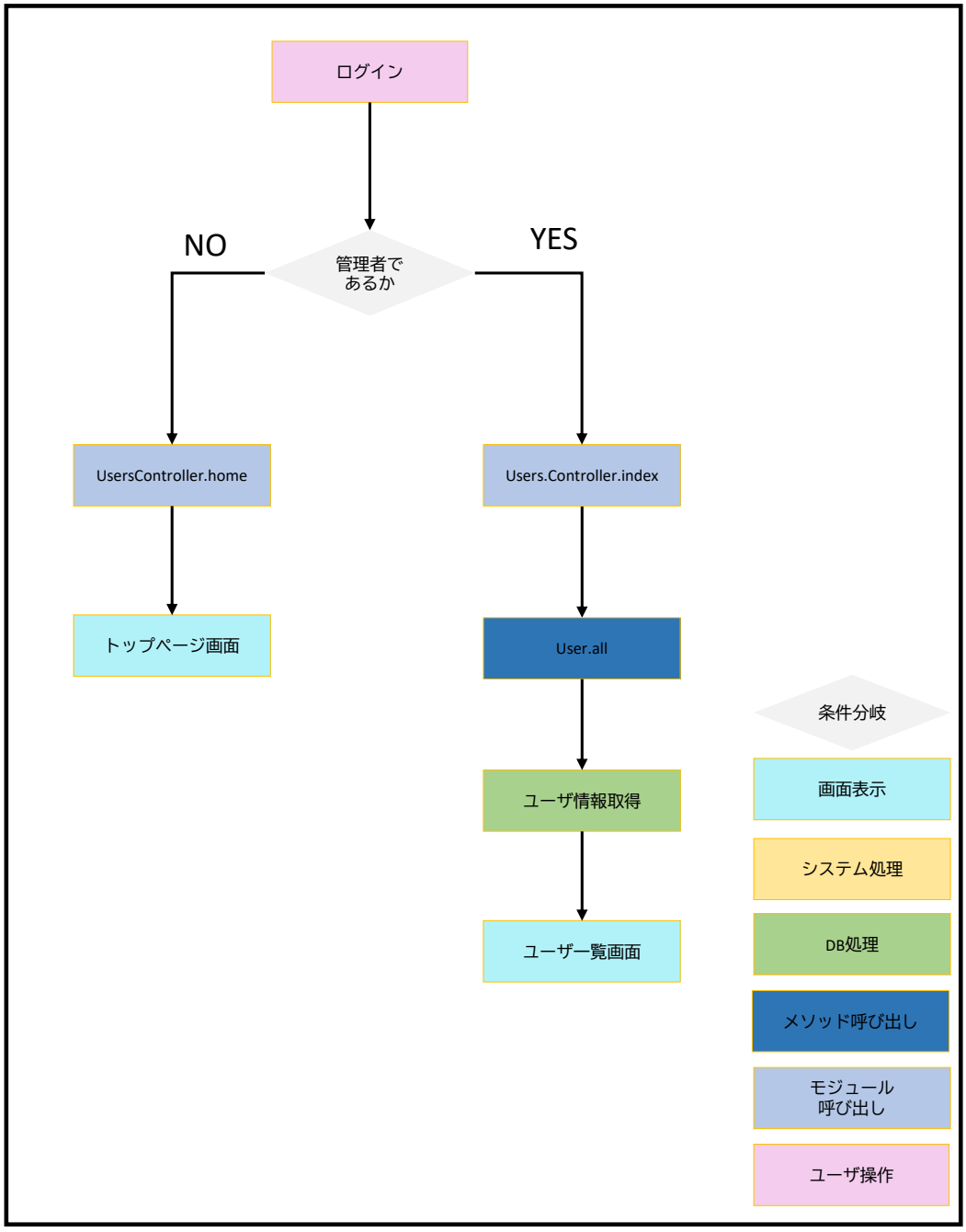

図 4.2: UsersController.index フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/2
	作成者	奥平舜理	更新日	2023/12/8
基本情報	概要	UsersController 内の home アクション		
	所属クラス	UsersController		
	モジュール名	home		
	モジュールID	UsersController.home		

処理説明

【処理内容】
楽譜データの一覧を表示する．
【処理手順】
1. 引き渡されたユーザidに対して，楽譜データ一覧を生成する．
2. views/scores/index.html.erb を描画する．
【補足】

入力値説明

ユーザid

出力値説明

楽譜データの一覧

他クラス・関数との関係

ApplicationController クラスを継承する．

図 4.3: UsersController.home 定義書

所属クラス	UsersController	作成日	2023/12/08
機能名	楽譜データー覧表示	更新日	2023/12/08
モジュールID	UsersController.home	作成者	奥平舜理
使用モジュールID	UsersController.index		

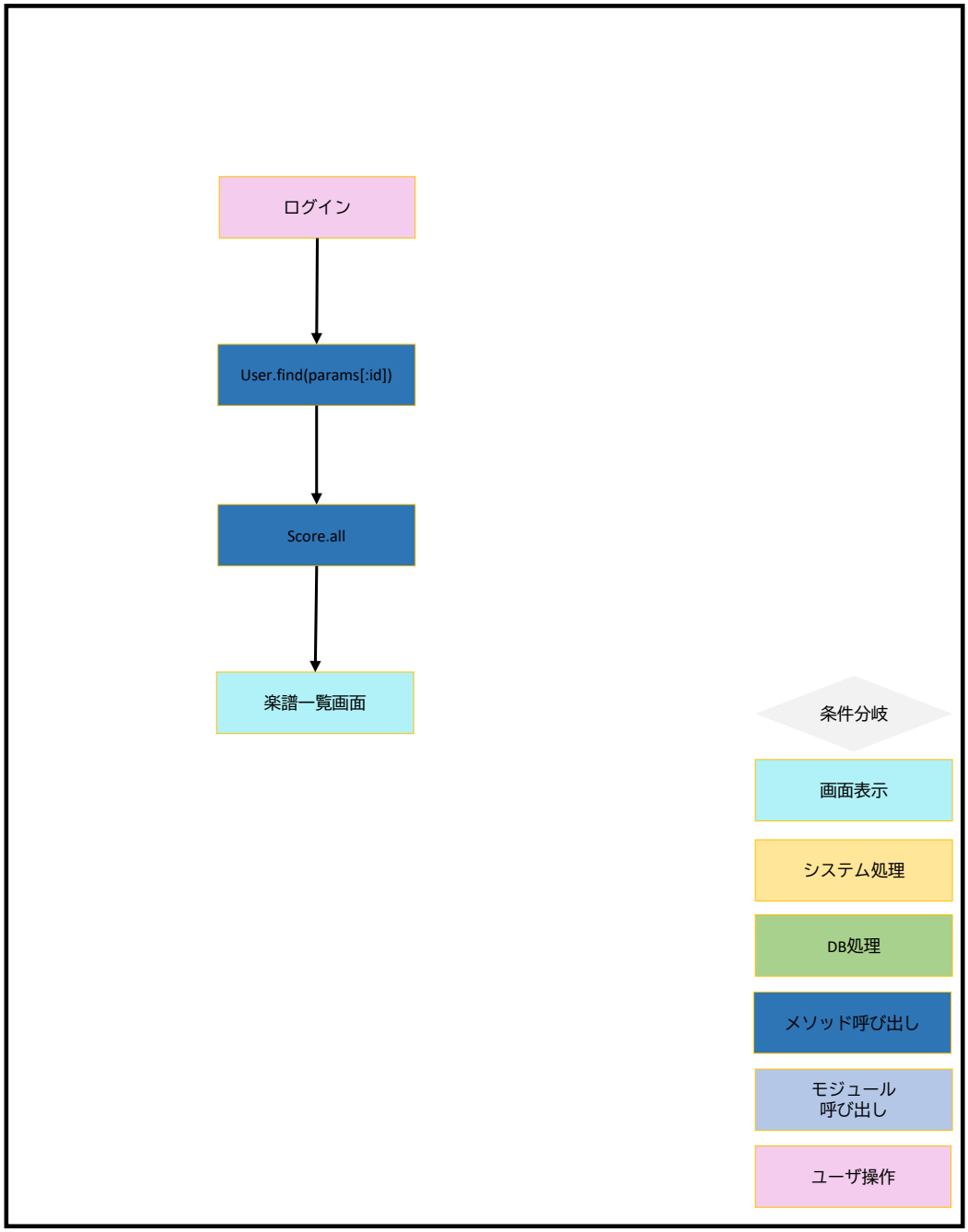

図 4.4: UsersController.home フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/1
	作成者	中村祐貴	更新日	2023/12/6
基本情報	概要	ScoresController 内の show アクション		
	所属クラス	ScoresController		
	モジュール名	show		
	モジュールID	ScoresController.show		

処理説明

【処理内容】

楽譜の詳細画面を表示する．

【処理手順】

1. Score クラスの find メソッドを用いて，インスタンスを変数 @score に格納する．
   find メソッドの引数は params[:id] とする．
2. views/scores/show.html.erb を描画する．

【補足】

入力値説明

score_id

出力値説明

なし

他クラス・関数との関係

ApplicationController を継承する．

図 4.5: ScoresController.show 定義書

所属クラス	ScoresController	作成日	2023/12/7
機能名	楽譜情報詳細	更新日	2023/12/8
モジュールID	ScoresController.show	作成者	中村祐貴
使用モジュールID			

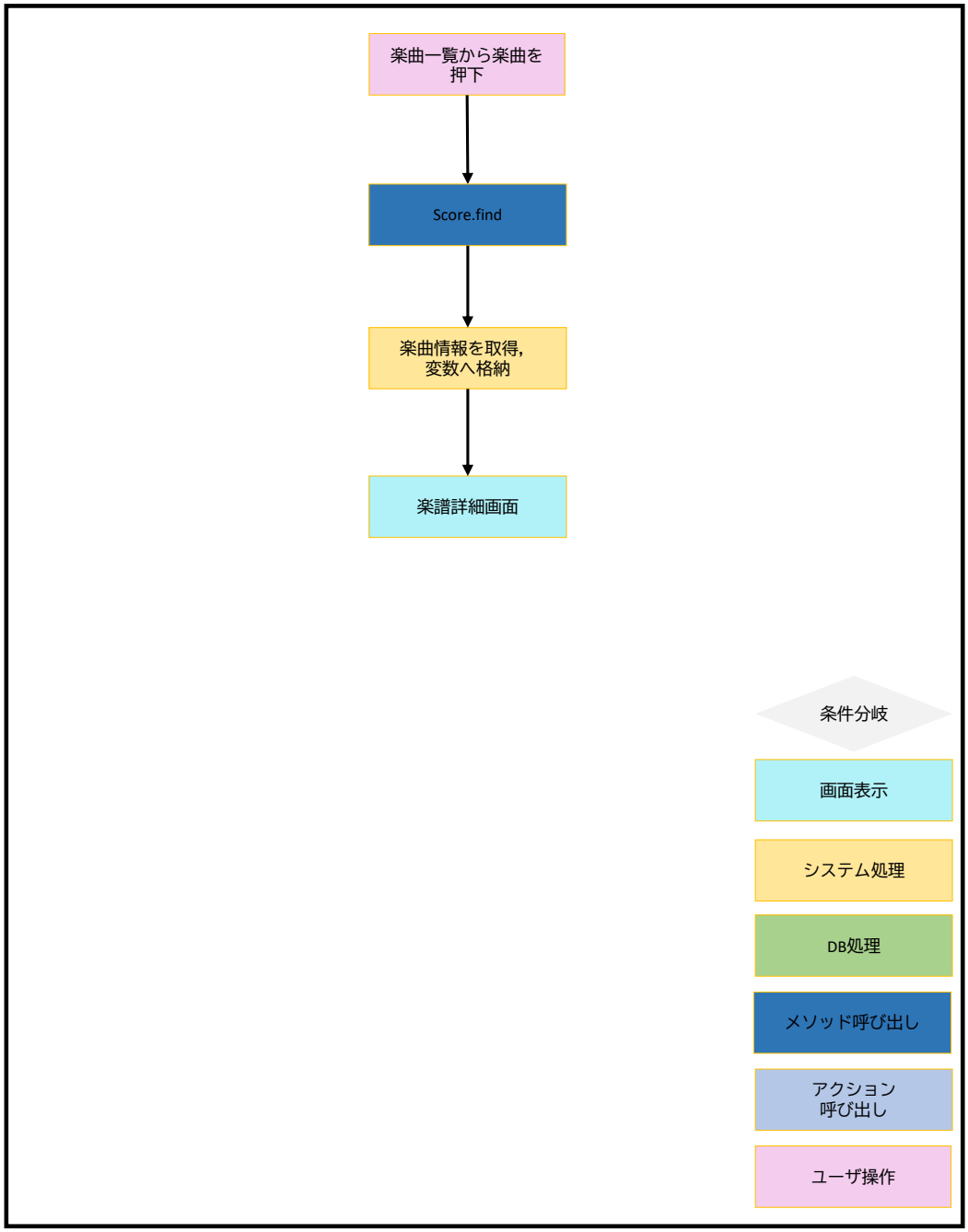

図 4.6: ScoresController.show フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/1
	作成者	中村祐貴	更新日	2023/12/7
基本情報	概要	UsersController 内の show アクション		
	所属クラス	UsersController		
	モジュール名	show		
	モジュールID	UsersController.show		

処理説明

【処理内容】
ユーザの詳細画面を表示する．
【処理手順】
1. current_user が admin か user か判定する．current_user が uesr なら自身の user_id しか取得できない．
（admin ならユーザの詳細画面，user なら自身のユーザ情報詳細画面が表示される．）
2. User クラスの find メソッドを用いて，インスタンスを変数 @user に格納する．
find メソッドの引数は params[:id] とする．
3. views/users/show.html.erb を描画する．
【補足】
ユーザが url 直接入力により，他のユーザ情報を閲覧しようとすると自身のユーザ情報画面が表示される．

入力値説明

user_id

出力値説明

なし

他クラス・関数との関係

ApplicationController を継承する．

図 4.7: UsersController.show 定義書

所属クラス	UsersController	作成日	2023/12/7
機能名	ユーザ情報詳細	更新日	2023/12/8
モジュールID	UsersController.show	作成者	中村祐貴
使用モジュールID			

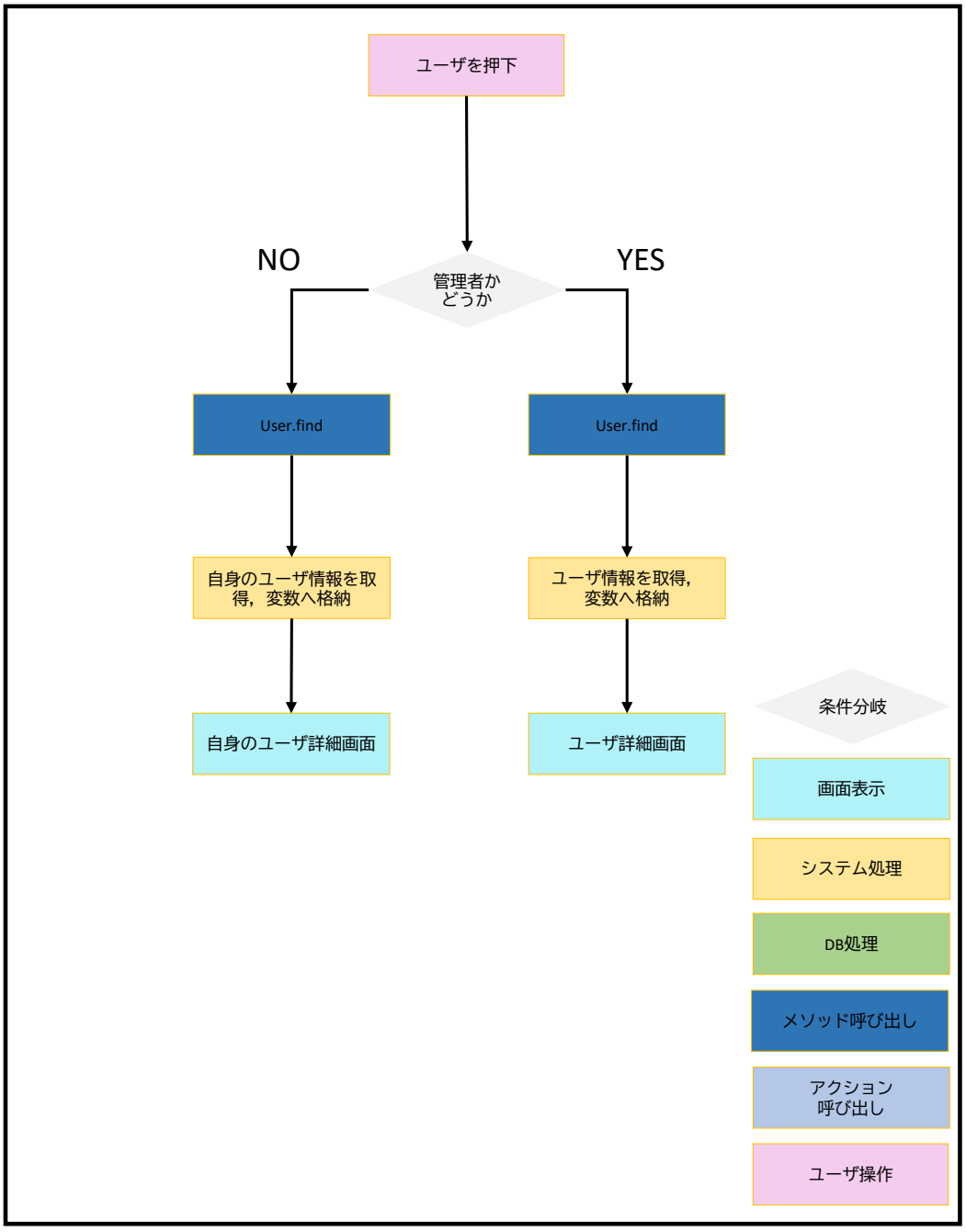

図 4.8: UsersController.show フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/4
	作成者	山本祥弘	更新日	2023/12/5
基本情報	概要	ScoresController 内の new アクション		
	所属クラス	ScoresController		
	モジュール名	new		
	モジュールID	ScoresController.new		

処理説明
【処理内容】 新しい楽譜データを作成するページを表示するアクション. 【処理手順】 1. views/scores/new.html.erb を描画する. 【補足】

入力値説明
なし

出力値説明
なし

他クラス・関数との関係
ApplicationController クラスを継承する.

図 4.9: ScoresController.new 定義書

所属クラス	ScoresController	作成日	2023/12/07
機能名	楽譜データ登録画面表示	更新日	2023/12/07
モジュールID	ScoresController.new	作成者	山本祥弘
使用モジュールID			

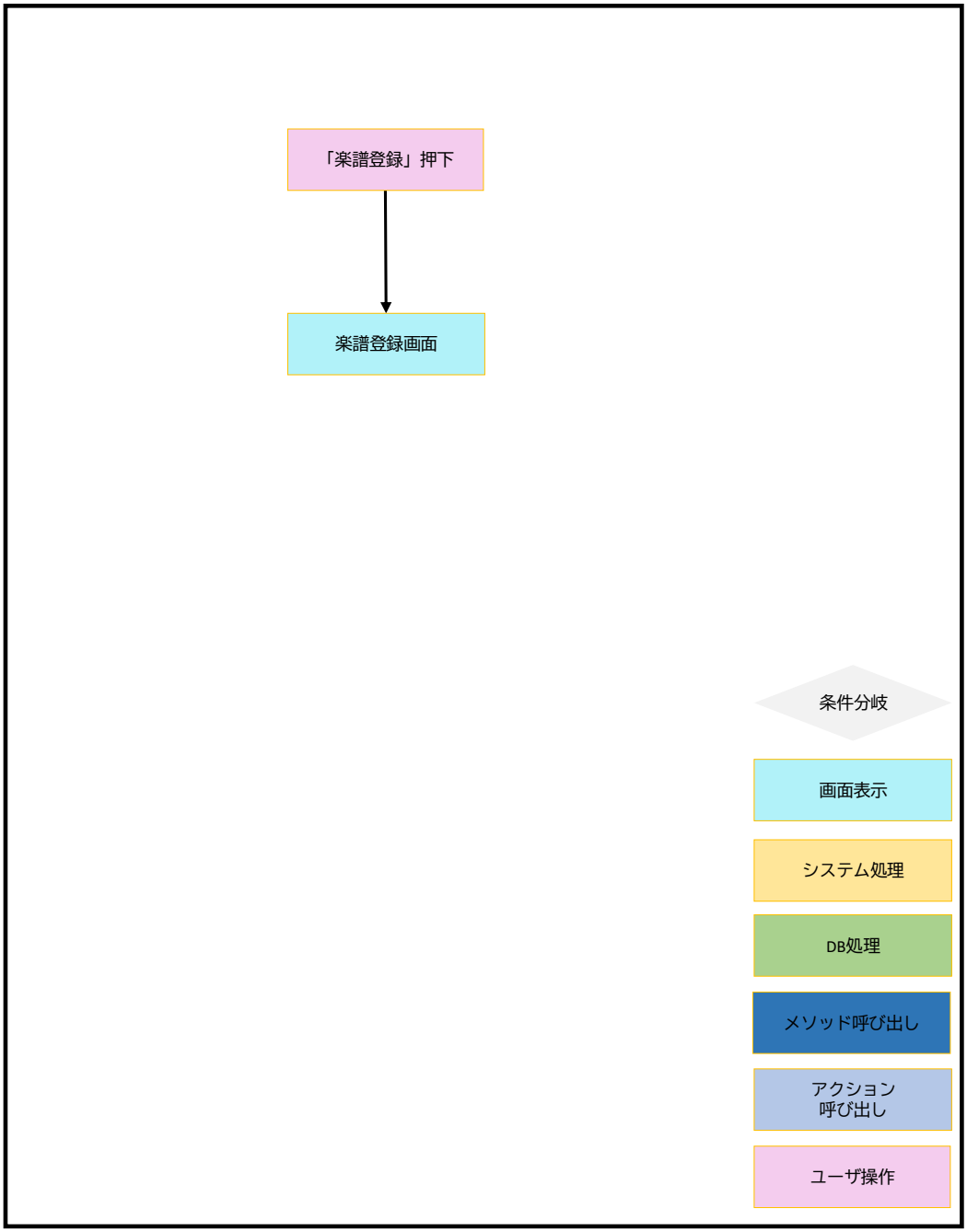

図 4.10: ScoresController.new フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/4
	作成者	山本祥弘	更新日	2023/12/5
基本情報	概要	UsersController 内の new アクション		
	所属クラス	UsersController		
	モジュール名	new		
	モジュールID	UsersController.new		

処理説明

【処理内容】

サインアップ画面を表示するアクション.

【処理手順】

1. views/users/new.html.erb を描画する.

【補足】

入力値説明

なし

出力値説明

なし

他クラス・関数との関係

ApplicationController クラスを継承する.

図 4.11: UsersController.new 定義書

所属クラス	UsersController	作成日	2023/12/07
機能名	サインアップ画面表示	更新日	2023/12/08
モジュールID	UsersController.new	作成者	山本祥弘
使用モジュールID			

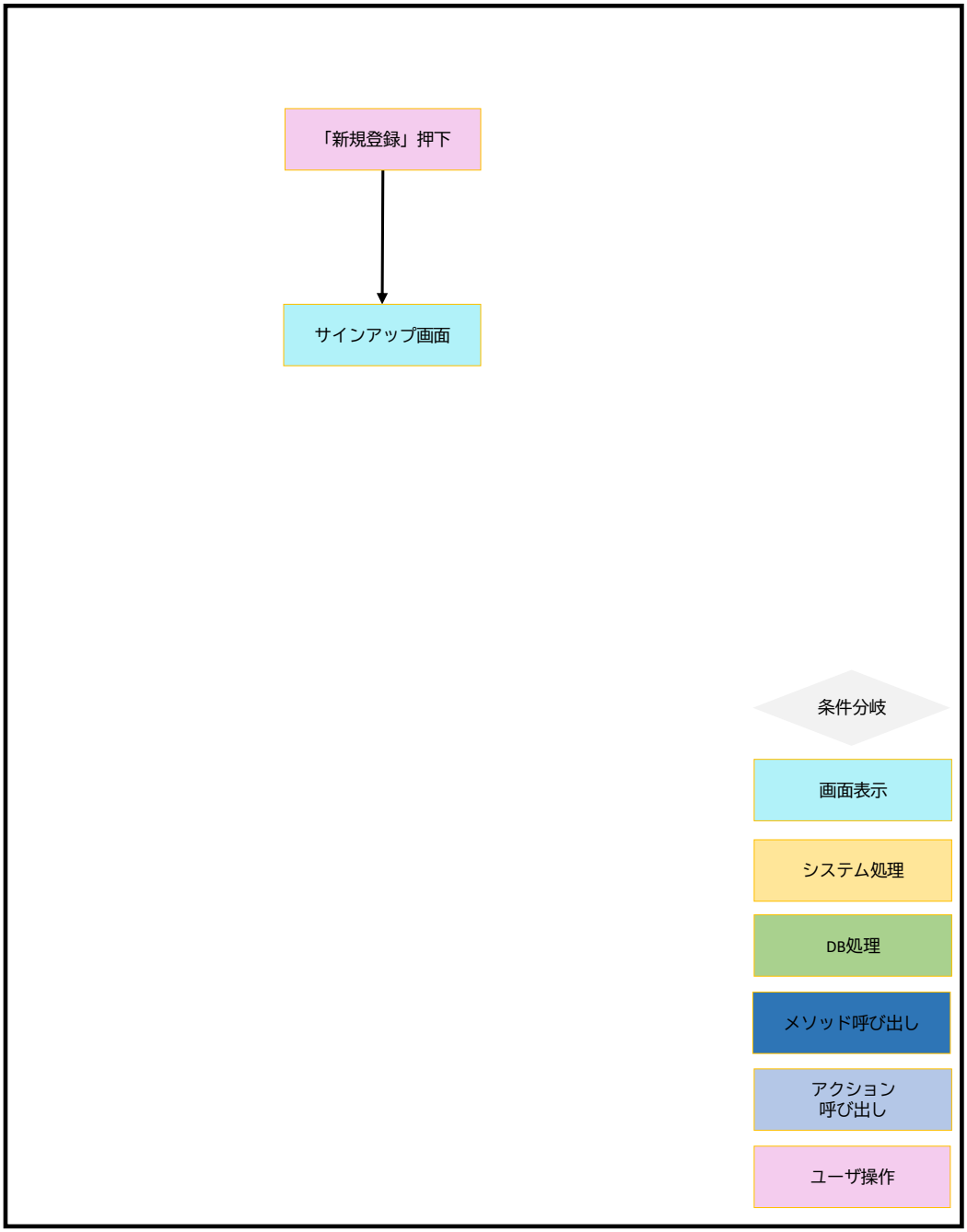

図 4.12: UsersController.new フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/3
	作成者	山田滉希	更新日	2023/12/5
基本情報	概要	ScoresController 内の edit アクション		
	所属クラス	ScoresController		
	モジュール名	edit		
	モジュールID	ScoresController.edit		

処理説明

【処理内容】

楽譜データについて編集ページを作成する．

【処理手順】

1. current_user_id と編集する対象の user_id が一致しているかの確認を行う．
2. 一致しておれば Score クラスの find メソッドを用いて，インスタンスを変数 @score に格納する．（find メソッドの引数は params[:id] とする．）
3. views/scores/edit.html.erb を描画する．

【補足】

入力値説明

score_id，user_id，current_user_id

出力値説明

なし

他クラス・関数との関係

ApplicationController クラスを継承する．

図 4.13: ScoresController.edit 定義書

所属クラス	ScoresController	作成日	2023/12/5
機能名	楽譜編集画面生成	更新日	2023/12/8
モジュールID	ScoresController.edit	作成者	山田滉希
使用モジュールID	ScoresController.edit		

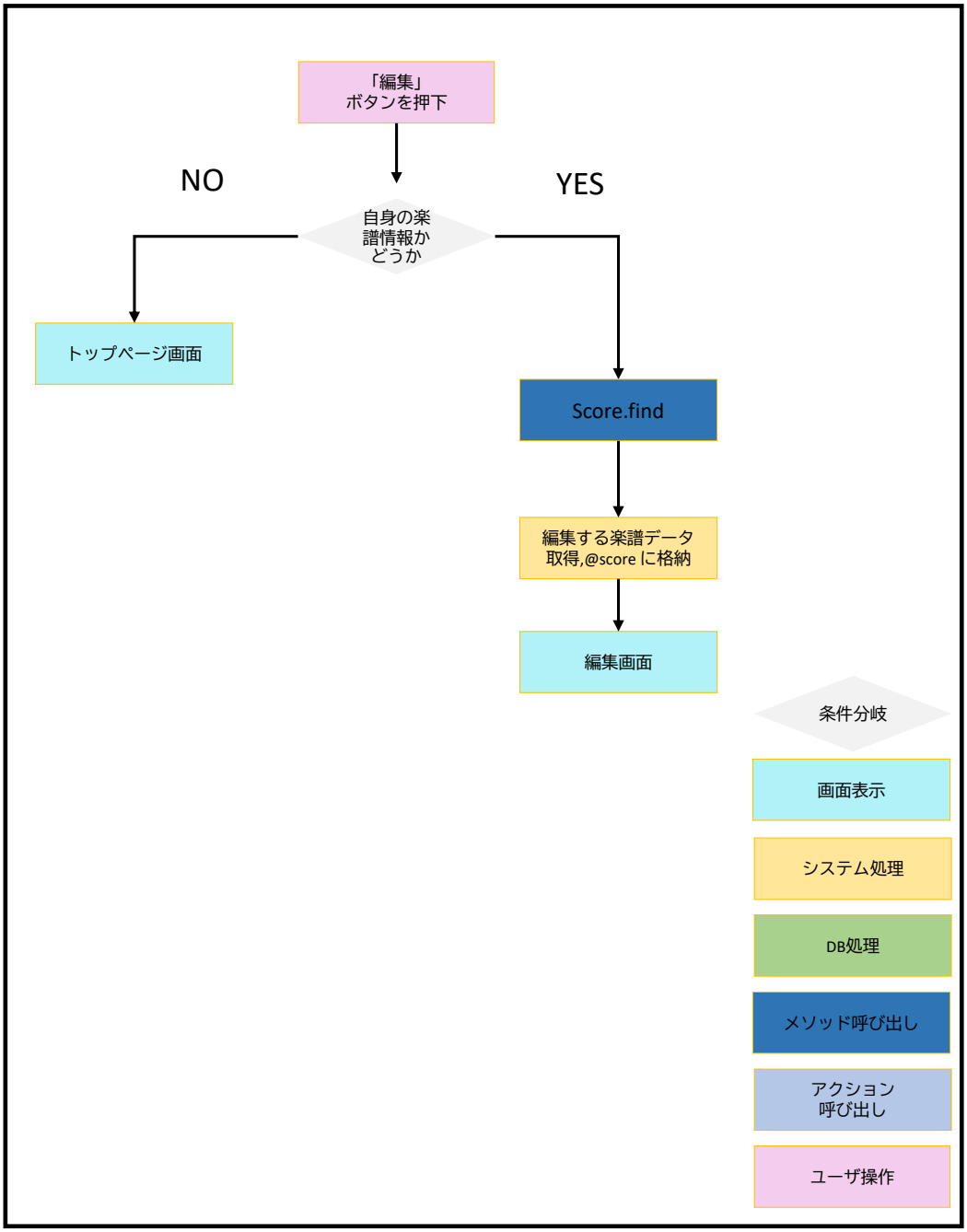

図 4.14: ScoresController.edit フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/3
	作成者	山田滉希	更新日	2023/12/5
基本情報	概要	UsersController 内の edit アクション		
	所属クラス	UsersController		
	モジュール名	edit		
	モジュールID	UsersController.edit		

処理説明

【処理内容】
ユーザについての編集ページを作成する．
【処理手順】
1. current_user_id と admin_id が一致しているか判定し， 一致しなければ変数 @user には current_user_id に紐づけられた user_id しか格納できない．
2. 一致しているなら変数 @user には任意の user_id を格納できる．
2. User クラスの find メソッドを用いて，インスタンスを変数 @user に格納する．(find メソッドの引数は params[:id] とする．)
3. views/users/edit.html.erb を描画する．
【補足】

入力値説明

user_id，current_user_id

出力値説明

なし

他クラス・関数との関係

ApplicationController クラスを継承する．

図 4.15: UsersController.edit 定義書

所属クラス	UsersController	作成日	2023/12/5
機能名	ユーザ編集画面生成	更新日	2023/12/8
モジュールID	UsersController.edit	作成者	山田滉希
使用モジュールID	UsersController.edit		

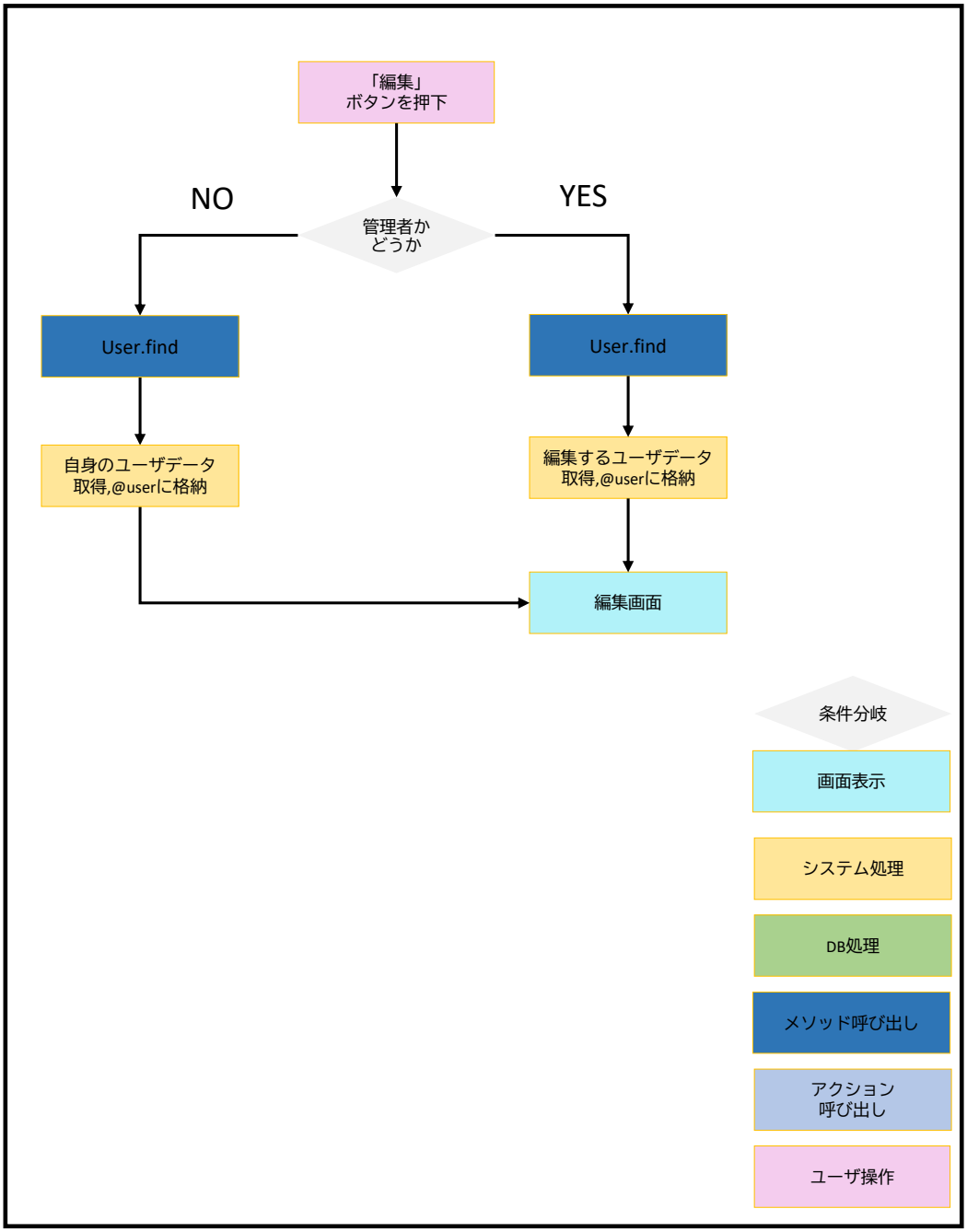

図 4.16: UsersController.edit フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	1.0.2
	工程名	内部設計	作成日	2023/12/03
	作成者	三上柊	更新日	2023/12/08
基本情報	概要	ScoresController 内の create アクション		
	所属クラス	ScoresController		
	モジュール名	create		
	モジュールID	ScoresController.create		

処理説明

【処理内容】
新たに作成された楽譜データをDBに登録する．

【処理手順】
1. score_params メソッドにて入力を取得する
2. Score クラスの new メソッドに score_params の戻り値を引数として渡し， @score 変数に結果を格納する．
3. save メソッドを用いてDBに @score の中身を保存する．
4. 保存が成功しているかを判定する．
5. 成功していれば成功を意味する flash を含めて ScoresController.index を呼び出す．
6. 失敗していればエラーを意味する flash を含めて ScoresController.new を呼び出す．

【補足】
・score_params メソッドは private である．new での入力内容を取得する．
・flash とは，リダイレクト時に，ページ上に一度だけメッセージを表示するメソッドである．
・saveとは，Railsに標準搭載されたDBにインスタンスを追加するメソッドである．

入力値説明

:composer：作曲者，:arranger：編曲者，
:name：曲名，:grade：難易度，:time：演奏時間，
使用楽器：:piccolo：ピッコロ, :c_flute:フルート 等

出力値説明

他クラス・関数との関係

ApplicationController クラスを継承する．
score_params メソッドを作成，利用する．このメソッドは private である．

図 4.17: ScoresController.create 定義書

所属クラス	ScoresController	作成日	2023/12/06
機能名	楽曲登録	更新日	2023/12/08
モジュールID	ScoresController.create	作成者	三上柊
使用モジュールID	ScoresController.index， UsersController.home		

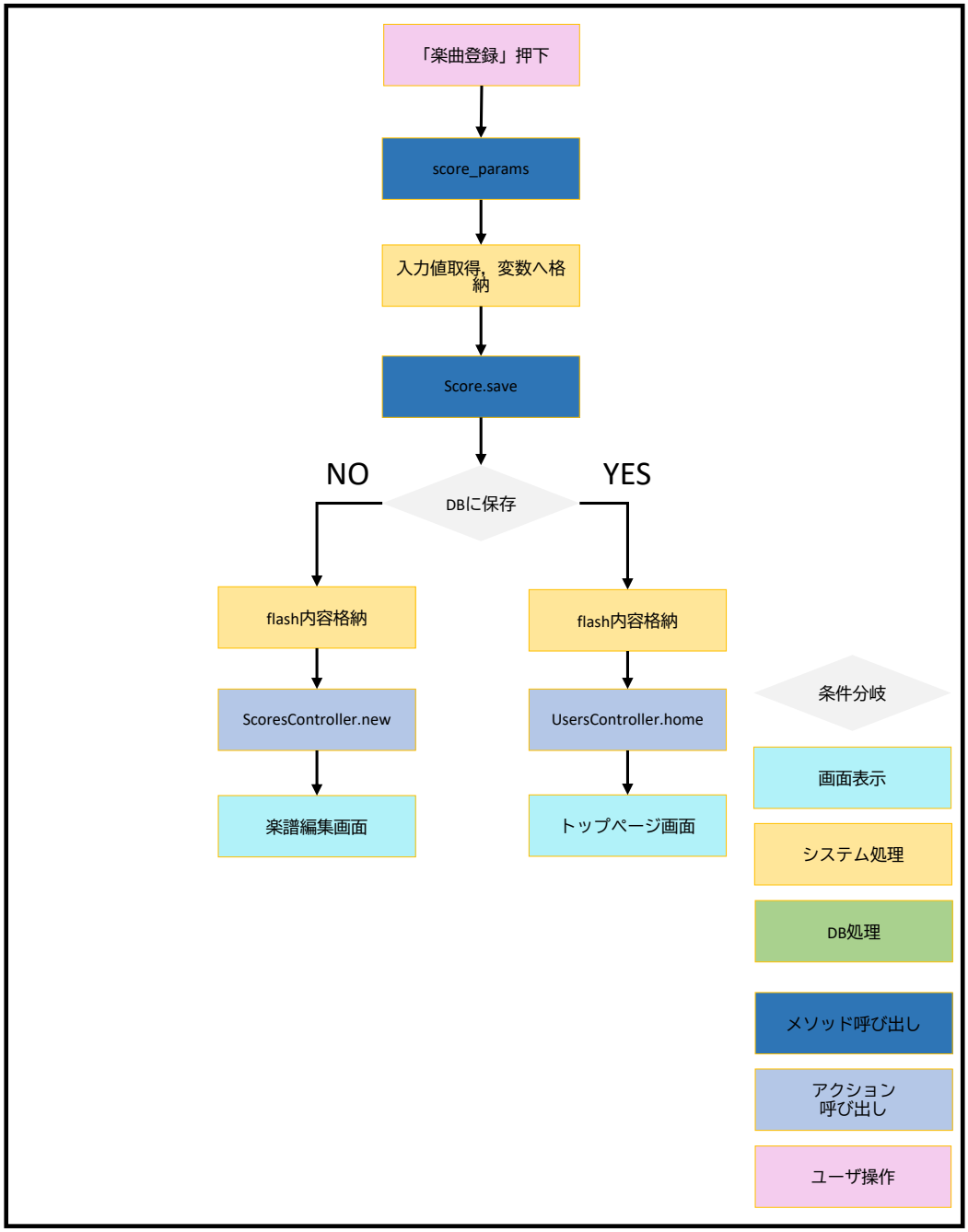

図 4.18: ScoresController.create フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	1.0.2
	工程名	内部設計	作成日	2023/12/03
	作成者	三上柊	更新日	2023/12/08
基本情報	概要	UsersController 内の create アクション		
	所属クラス	UsersController		
	モジュール名	create		
	モジュールID	UsersController.create		

処理説明

【処理内容】
新規のユーザをDBに登録する．
【処理手順】
1. user_params メソッドを用いて new ページでの入力内容を取得する．
2. User クラスの new メソッドを用いて，入力をインスタンス変数 @user に格納する．
3. save メソッドを用いて @user をDBに保存する．
4. 保存が成功したか判定する．
5. 成功していればログイン処理を行い，成功を意味する flash を含めて，ScoresController.index を呼び出す．
6. 失敗していたらエラーを意味する flash を含めて UsersController.new を呼び出す．
【補足】
・このアクションは new ページにて「登録」が押された際に呼び出される．
・「ログイン処理」は SessionsController.create を参照．
・user_params メソッドは private である．new での入力内容を取得する．
・flash とは，リダイレクト時に，一度だけページ上にメッセージを表示させるメソッドである．
・saveとは，Railsに標準搭載された，DBにインスタンスを追加するメソッドである．

入力値説明

:name：ユーザーネーム, :email：メールアドレス,
:pasword：パスワード, :password_confirmation：パスワード再確認

出力値説明

他クラス・関数との関係

ApplicationController クラスを継承する．
user_params メソッドを使用する．

図 4.19: UsersController.create 定義書

所属クラス	UsrsController	作成日	2023/12/06
機能名	ユーザ新規登録	更新日	2023/12/08
モジュールID	UsersController.create	作成者	三上柊
使用モジュールID	UsersController.new， UsersContoroller.home		

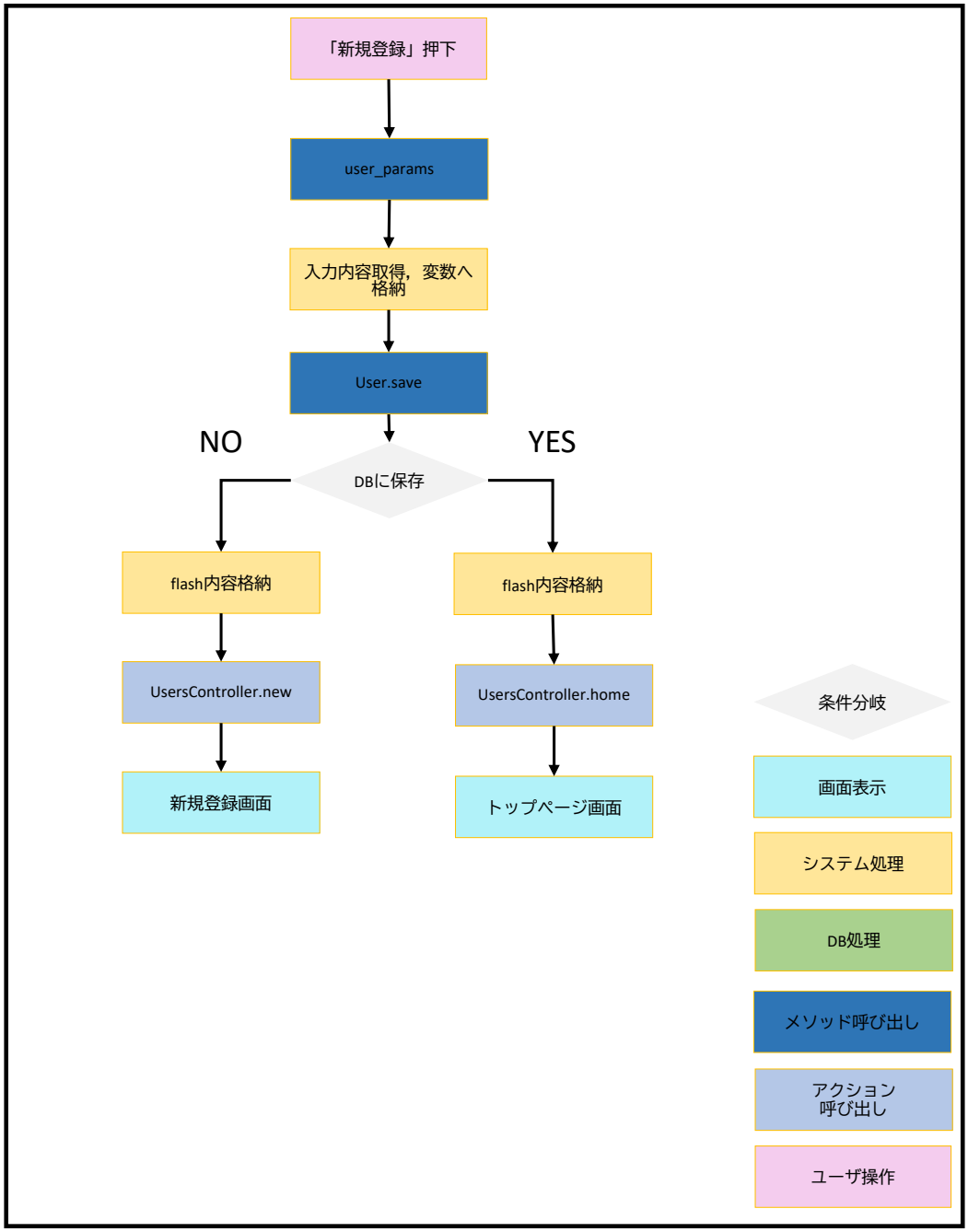

図 4.20: UsersController.create フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/4
	作成者	山本祥弘	更新日	2023/12/8
基本情報	概要	ScoresController 内の destroy アクション		
	所属クラス	ScoresController		
	モジュール名	destroy		
	モジュールID	ScoresController.destroy		

処理説明

【処理内容】
楽譜削除ボタンが押されたときに呼び出す．楽譜データを削除する．
【処理手順】
1. current_user.id が管理者もしくはログインしたユーザであるかを判定する．
2. current_user.id がログインしたユーザである場合，ユーザ削除確認ボックスを表示する．
current_user.id が管理者であった場合，パスワード認証を含むユーザ削除ボックスを表示する．
3.「戻る」ボタンが押されると UsersController.home を呼び出し処理を終了する．「削除」ボタンが押されると， Score.find(id).destroy を実行し，楽譜データを削除する．楽譜データを削除する際，
current_user.id が管理者である場合にはパスワード認証を通過している必要がある．
4. UsersController.home を呼び出し，削除完了の flash を表示する．
【補足】

入力値説明

:score_id : スコアID，:user_id : ユーザID

出力値説明

なし

他クラス・関数との関係

ApplicationController クラスを継承する．

図 4.21: ScoresController.destroy 定義書

所属クラス	ScoresController	作成日	2023/12/07
機能名	楽譜データ削除	更新日	2023/12/08
モジュールID	ScoresController.destroy	作成者	山本祥弘
使用モジュールID	UsersController.home		

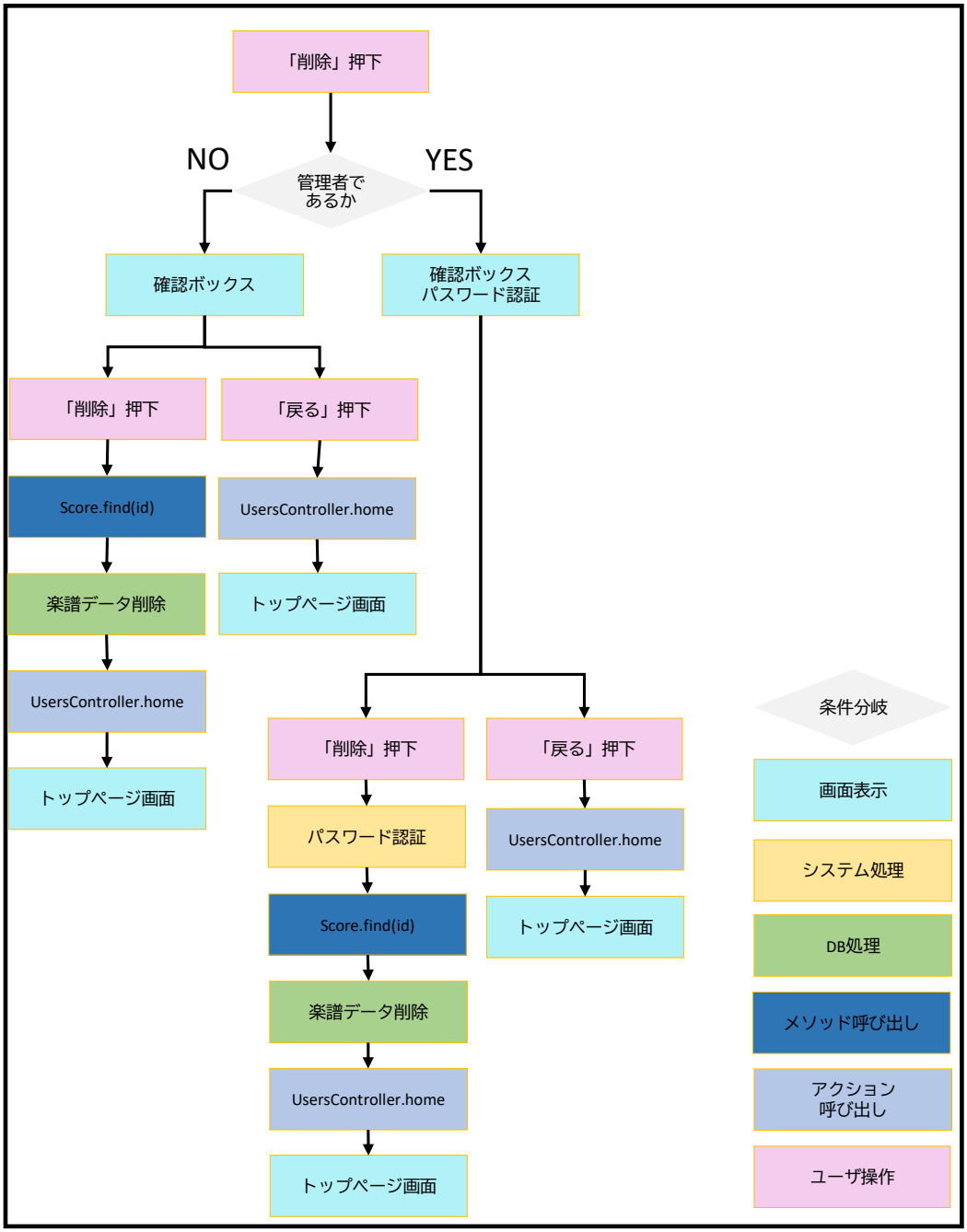

図 4.22: ScoresController.destroy フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/4
	作成者	山本祥弘	更新日	2023/12/5
基本情報	概要	UsersController 内の destroy アクション		
	所属クラス	UsersController		
	モジュール名	destroy		
	モジュールID	UsersController.destroy		

処理説明

【処理内容】
ユーザ削除ボタンが押されたときに呼び出す．データベースに登録されているユーザを削除する．
【処理手順】
1. current_user.id が管理者もしくはログインしたユーザであるかを判定する．
2. current_user.id がログインしたユーザである場合，ユーザ削除確認ボックスを表示して手順3-1に移行する．current_user.id が管理者であった場合，パスワード認証を含むユーザ削除ボックスを表示して手順4-1に移行する．
3-1.「戻る」ボタンが押されると UsersController.show を呼び出し処理を終了する．「削除」ボタンが押されると， current_user.id と削除するユーザが同じであるかを判定し，同じである場合のみ User.find(id).destroy を実行してユーザを削除する．
3-2. UsersController.new を呼び出し，削除完了の flash を表示して処理を終了する．
4-1.「戻る」ボタンが押されると UsersController.index を呼び出し処理を終了する．パスワード認証を通過し，かつ「削除」ボタンが押されると， User.find(id).destroy を実行し，ユーザを削除する．
4-2. UsersController.index を呼び出し，削除完了の flash を表示する．
【補足】
命令 has_many :scores, dependent: :destroy により，削除されるユーザに関連づけられた楽譜データも同時に削除される．

入力値説明

:user_id : ユーザID

出力値説明

なし

他クラス・関数との関係

ApplicationController クラスを継承する．

図 4.23: UsersController.destroy 定義書

所属クラス	UsersController	作成日	2023/12/07
機能名	ユーザ削除	更新日	2023/12/07
モジュールID	UsersController.destroy	作成者	山本祥弘
使用モジュールID	UsersController.show， UsersController.new， UsersController.index		

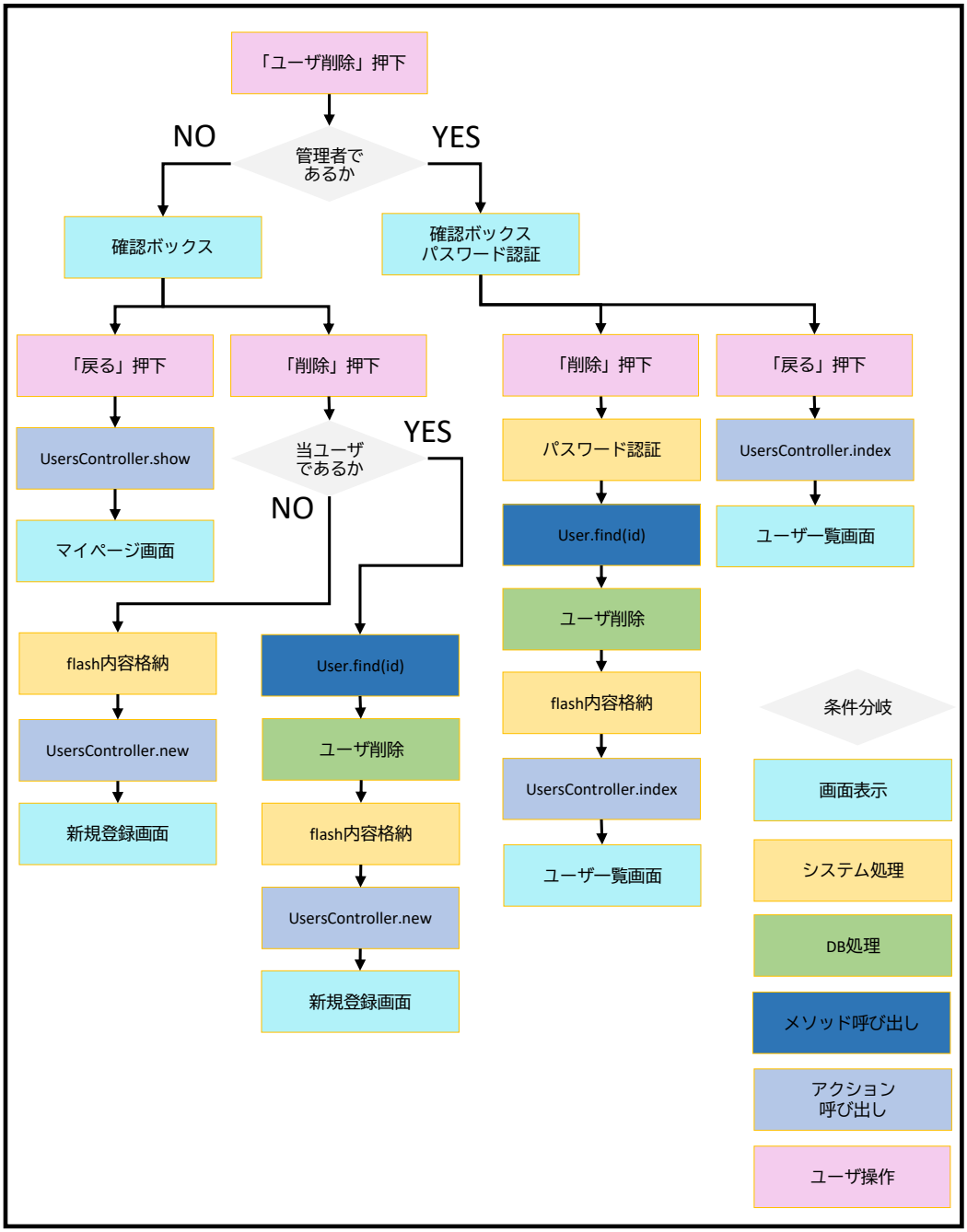

図 4.24: UsersController.destroy フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/05
	作成者	溝口洸熙	更新日	2023/12/05
基本情報	概要	SessionsController 内の new メソッド		
	所属クラス	SessionsController		
	モジュール名	new		
	モジュールID	SessionsController.new		

処理説明

【処理内容】

セッションを作成する画面（ログイン画面）を表示するアクション.

【処理手順】

1. views/sessions/new.html.erb を描画する.

【補足】

入力値説明

なし

出力値説明

なし

他クラス・関数との関係

ApplicationController を継承する.

図 4.25: SessionsController.new 定義書

所属クラス	SessionsController	作成日	2023/12/08
アクション名	new	更新日	2023/12/08
モジュールID	SessionsController.new	作成者	田中諒
使用モジュールID			

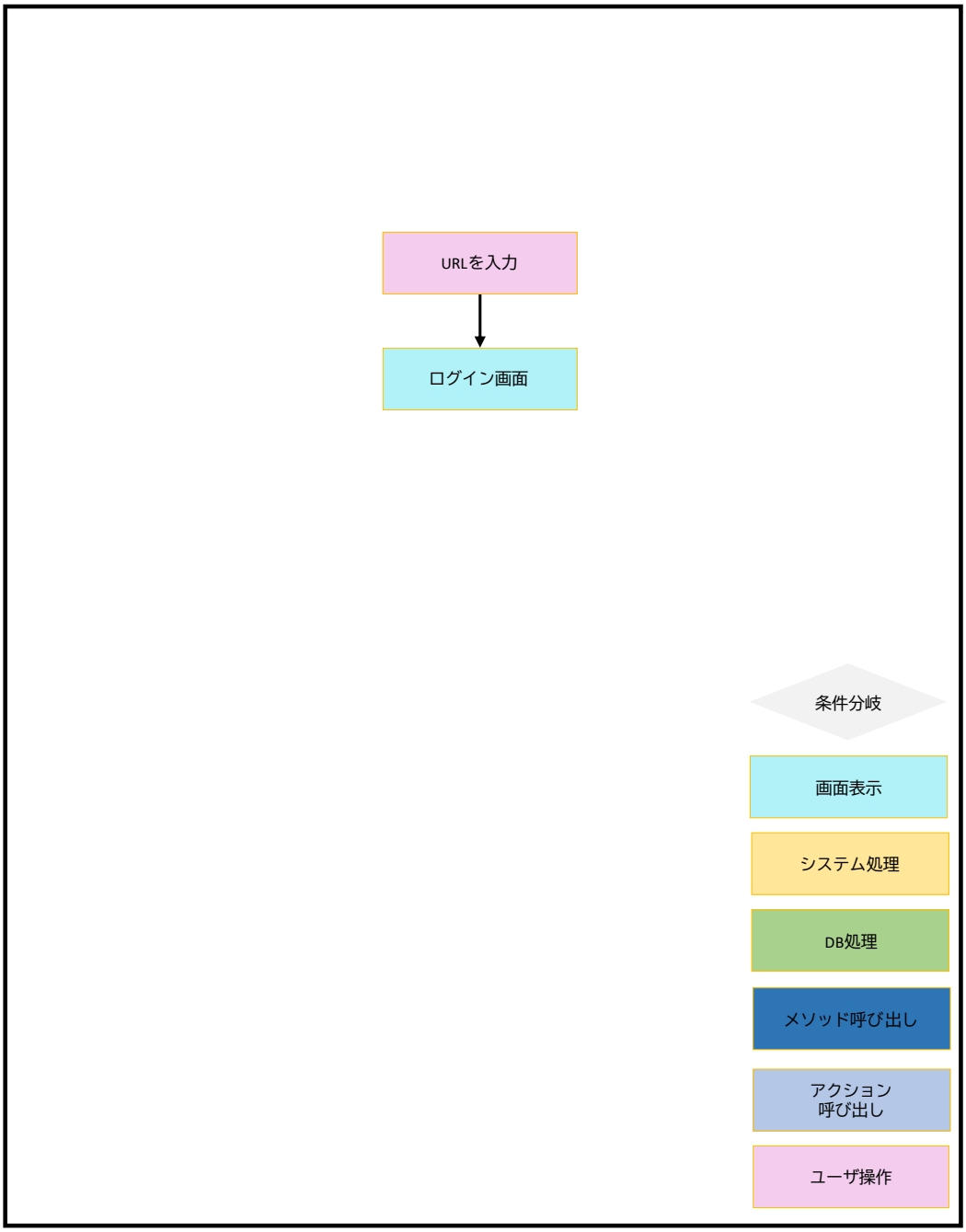

図 4.26: SessionsController.new フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/04
	作成者	溝口洸熙	更新日	2023/12/04
基本情報	概要	SessionsController 内の create メソッド		
	所属クラス	SessionsController		
	モジュール名	create		
	モジュールID	SessionsController.create		

処理説明

【処理内容】
セッションを作成する画面（ログイン画面）を処理する．
【処理手順】
1. User クラスの find_by 関数を用いて，入力にある :email と一致するユーザインスタンスを変数名 user に格納する．
2. 取得した :password を用いて認証する．
3. 認証に通過した場合，ログイン処理をする．ログイン処理は，session の :user_id に， user_id を渡す処理をする．ここで，ログイン時刻を users テーブルに格納する．
4. 認証に失敗した場合は，HTTPステータスコードを「422 Unprocessable Entity」として応答し，認証失敗のflashを表示し， SessionsController.new を呼び出す．
【補足】

入力値説明

:email：ログイン情報のEメールのアドレス
:password：ログイン情報のパスワード

出力値説明

なし

他クラス・関数との関係

ApplicationController を継承する．

図 4.27: SessionsController.create 定義書

所属クラス	SessionsController	作成日	2023/12/06
アクション名	create	更新日	2023/12/08
モジュールID	SessionsController.create	作成者	三上柊
使用モジュールID	UsersController.home		

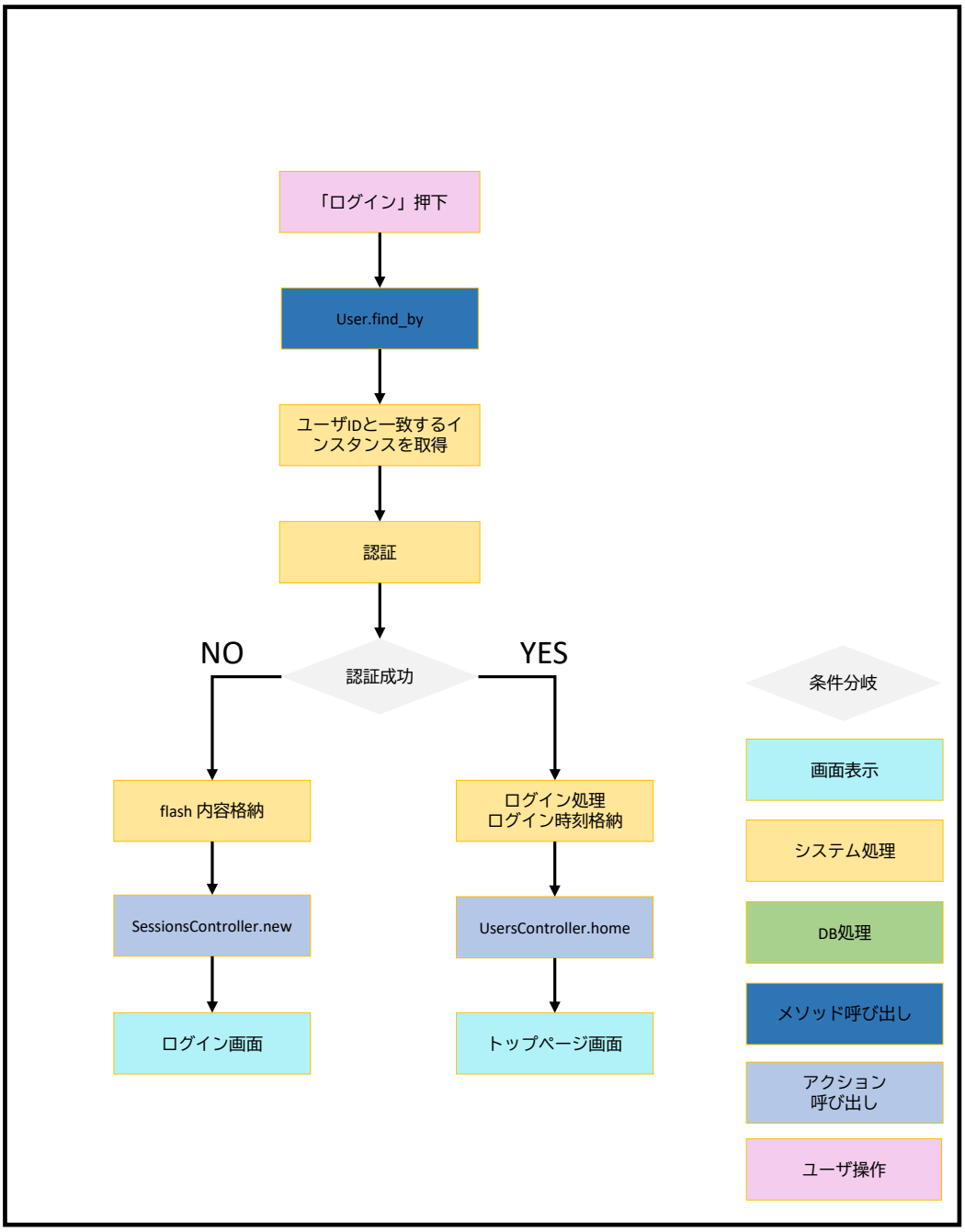

図 4.28: SessionsController.create フロー図

モジュール定義				
管理情報	システム名	楽譜管理ソフトウェア	バージョン	
	工程名	内部設計	作成日	2023/12/05
	作成者	溝口洸熙	更新日	2023/12/05
基本情報	概要	SessionsController 内の destroy メソッド		
	所属クラス	SessionsController		
	モジュール名	destroy		
	モジュールID	SessionsController.destroy		

処理説明

【処理内容】

ログアウトボタンを押下した時に呼び出す．ログアウト処理をする．

【処理手順】

1. ログアウト処理をする． reset_session を実行する．
2. current_user に nil を代入する．
3. SessionsController.new を呼び出す．

【補足】

reset_session は，セッションを初期化するメソッドで，Rails標準のものである．

入力値説明

なし

出力値説明

なし

他クラス・関数との関係

ApplicationController を継承する．

図 4.29: SessionsController.destroy 定義書

所属クラス	SessionsController	作成日	2023/12/06
アクション名	destroy	更新日	2023/12/08
モジュールID	SessionsController.destroy	作成者	田中諒
使用モジュールID	ScoresController.new		

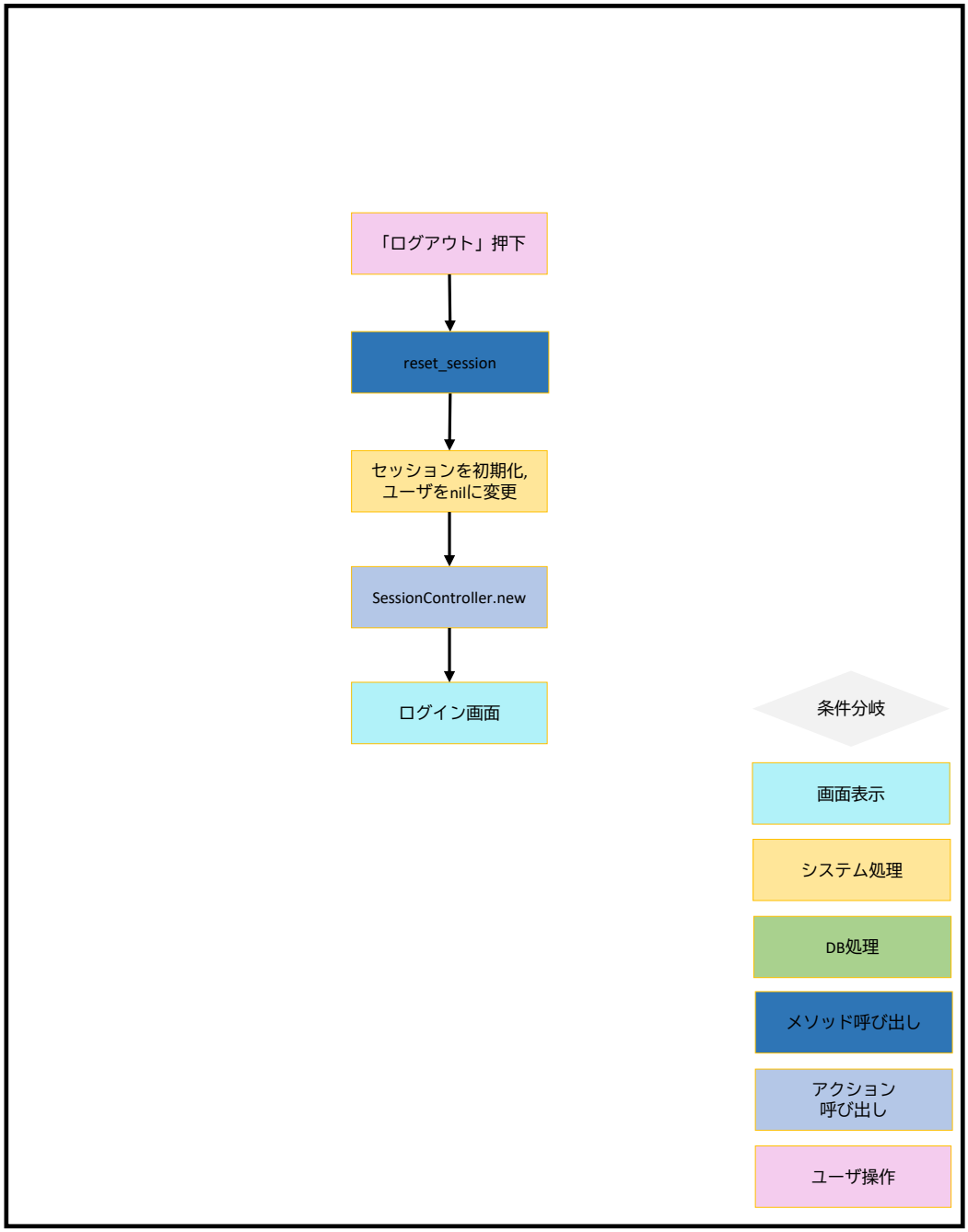

図 4.30: SessionsController.destroy フロー図

第 5 章

DB

DB 構造を以下に示す.


User
int user_id PK ユーザID
text email メールアドレス
text name 組織名など
text password パスワード
boolean admin 管理者であるか
date login 最終ログイン日
timestamp created_at 作成日時
timestamp updated_at 更新日時
1つのユーザは0以上のスコアを持つ
Score
int score_id PK スコアID
references user_id FK 参照(ユーザ)
text name 楽曲名
text composer 作曲者名
text arranger 編曲者名
text grade グレード
int time 演奏時間(秒)
int piccolo
int c_flute
int oboe
int english_horn
int e_clarinet
int b_clarinet
int b_bass_clarinet
int bassoon
int e_alto_saxophone
int b_tenor_saxophone
int b_baritone_saxophone
int b_trumpet
int f_horn
int trombone
int baritone
int tuba
int string_bass
int piano
int harp
int timpani
int drums
int percussion
timestamp created_at 作成日時
timestamp updated_at 更新日時

図 5.1: ER 図

第 6 章

貢献内容

内部設計書第 1 版の作成における，各メンバの貢献内容は以下の通り．

学籍番号	氏名	貢献内容・担当箇所
1250297	奥平 舜理	モジュールの内容，システム実装方法 モジュール担当箇所：UC.index，UC.home
1250341	田中 諒	モジュールの内容，テンプレート作成 モジュール担当箇所：UC.update，SC.update
1250352	中村 祐貴	モジュールの作成，規約（TEX） モジュール担当箇所：UC.show，SC.show
1250372	三上 柊	モジュールの内容，テンプレート作成・更新，進捗管理，TEX 清書 モジュール担当箇所：UC.create，SC.create
1250373	溝口 洸熙	モジュールの内容，規約 モジュール担当箇所：SsC.create，SsC.new，Ssc.destroy
1250382	山田 滉希	モジュールの内容 モジュール担当箇所：UC.edit，SC.edit
1250385	山本 祥弘	モジュールの内容，テンプレート作成 モジュール担当箇所：UC.destroy，SC.destroy，UC.new，SC.new

- UC：UsersController
- SC：ScoresController
- SsC：SessionsController