cafeMuji 注文管理システム 画面遷移図

1. システム全体の画面構成

1.1 主要画面一覧

```
cafeMuji 注文管理システム
├── 共通画面
│   ├── ログイン画面 (login.html)
│   └── 役割選択画面 (role_select.html)
├── フード注文管理
│   ├── 注文登録画面 (food_register.html)
│   └── キッチン画面 (food_kitchen.html)
├── アイスクリーム注文管理
│   ├── 注文画面 (ice.html)
│   ├── 注文登録画面 (register.html)
│   ├── 注文詳細画面 (detail.html)
│   └── 注文削除画面 (deshap.html)
├── かき氷注文管理
│   ├── 注文画面 (shavedice.html)
│   ├── 注文登録画面 (shavedice_register.html)
│   ├── キッチン画面 (shavedice_kitchen.html)
│   └── 待ち時間画面 (wait_time.html)
└── モバイル対応
    ├── 注文画面 (mobile_order.html)
    └── 完了画面 (mobile_complete.html)
```

2. 画面遷移フロー

2.1 メインフロー

```
[ログイン] → [役割選択] → [各機能画面]
    ↓
[認証成功] → [権限確認] → [機能実行]
    ↓
[セッション管理] → [データ処理] → [画面更新]
```

2.2 詳細画面遷移

2.2.1 フード注文管理フロー

```
[ログイン] → [役割選択: レジ] → [フード注文登録]
    ↓
[メニュー選択] → [数量入力] → [店内/テイクアウト選択]
    ↓
[仮注文追加] → [クリップ情報設定] → [注文確認]
    ↓
[本注文確定] → [データベース保存] → [完了]
    ↓
[キッチン画面] ← [注文状態更新] ← [作業完了]
```

2.2.2 アイスクリーム注文管理フロー

```
[ログイン] → [役割選択: アイス担当] → [アイス注文画面]
    ↓
[サイズ選択] → [容器選択] → [フレーバー選択]
    ↓
```

[数量設定] → [クリップ情報設定] → [注文登録]
↓
[注文詳細確認] → [状態管理] → [完了処理]

2.2.3 かき氷注文管理フロー

[ログイン] → [役割選択: かき氷担当] → [かき氷注文画面]
↓
[フレーバー選択] → [クリップ情報設定] → [注文登録]
↓
[キッチン画面] → [作業管理] → [完了処理]
↓
[待ち時間表示] → [顧客案内]

3. 各画面の詳細仕様

3.1 共通画面

3.1.1 ログイン画面 (login.html)

URL: `/login/` **目的**: ユーザー認証 **機能**: - ユーザーID・パスワード入力 - 認証処理 - セッション開始

画面遷移: - 成功: 役割選択画面へ - 失敗: エラーメッセージ表示

3.1.2 役割選択画面 (role_select.html)

URL: `/role-select/` **目的**: 利用者の役割選択 **機能**: - レジ担当者 - フードキッチン担当者 - アイスクリーム担当者 - かき氷担当者

画面遷移: - レジ: フード注文登録画面へ - キッチン: 各キッチン画面へ - 各担当者: 各注文画面へ

3.2 フード注文管理画面

3.2.1 注文登録画面 (food_register.html)

URL: `/food/register/` **目的**: フード注文の受付・管理 **機能**: - メニュー選択（からあげ丼、ルーロー飯） - 数量入力 - 店内/テイクアウト選択 - 仮注文の追加・削除 - クリップ色・番号設定 - 本注文確定

画面要素:

```
┌──────────────────────────────────────────┐
│ フード注文登録                           │
├──────────────────────────────────────────┤
│ メニュー選択: [からあげ丼] [ルーロー飯]  │
│ 数量: [1] [2] [3] [4] [5]                │
│ 店内/テイクアウト: [店内] [テイクアウト] │
├──────────────────────────────────────────┤
│ [追加] [削除] [確定]                     │
├──────────────────────────────────────────┤
│ 仮注文一覧                               │
│ ・からあげ丼 ×2 (店内)                   │
│ ・ルーロー飯 ×1 (テイクアウト)           │
```

```
┌──────────────────────────────────────────────────────────┐
│ クリップ色: [黄色] [白色]                                │
│ クリップ番号: [1] [2] [3] ... [16]                       │
└──────────────────────────────────────────────────────────┘
```

画面遷移: - 追加ボタン: 仮注文リストに追加 - 削除ボタン: 選択された仮注文を削除 - 確定ボタン: 本注文として確定

3.2.2 キッチン画面 (food_kitchen.html)

URL: `/food/kitchen/` **目的**: キッチンでの作業管理 **機能**: - 注文一覧表示 - 注文状態の更新 - 完了処理 - 作業進捗の管理

画面要素:

```
┌──────────────────────────────────────────────────────────┐
│ フードキッチン                                           │
├──────────────────────────────────────────────────────────┤
│ 黄色クリップ1: からあげ丼 ×2（店内）                    │
│ [作業開始] [完了]                                        │
├──────────────────────────────────────────────────────────┤
│ 白色クリップ3: ルーロー飯 ×1（テイク）                  │
│ [作業開始] [完了]                                        │
├──────────────────────────────────────────────────────────┤
│ 完了済み注文                                             │
│ ・からあげ丼 ×1（店内）- 完了時刻: 14:30                 │
└──────────────────────────────────────────────────────────┘
```

画面遷移: - 作業開始: 注文状態を「作成中」に更新 - 完了: 注文完了処理、完了時刻記録

3.3 アイスクリーム注文管理画面

3.3.1 注文画面 (ice.html)

URL: `/ice/` **目的**: アイスクリーム注文の受付 **機能**: - サイズ選択（シングル/ダブル） - 容器選択（カップ/コーン） - フレーバー選択（12種類） - 注文登録

画面要素:

```
┌──────────────────────────────────────────────────────────┐
│ アイスクリーム注文                                       │
├──────────────────────────────────────────────────────────┤
│ サイズ: [シングル] [ダブル]                              │
│ 容器: [カップ] [コーン]                                  │
├──────────────────────────────────────────────────────────┤
│ フレーバー1: [ジャージー牛乳] [お茶] [マンゴー]...       │
│ フレーバー2: [選択なし] [ジャージー牛乳] [お茶]...       │
├──────────────────────────────────────────────────────────┤
│ クリップ色: [黄色] [白色]                                │
│ クリップ番号: [1] [2] [3] ... [16]                       │
├──────────────────────────────────────────────────────────┤
│ [注文登録]                                               │
└──────────────────────────────────────────────────────────┘
```

3.3.2 注文登録画面 (register.html)

URL: `/ice/register/` **目的**: アイスクリーム注文の詳細登録 **機能**: - 注文内容の確認 - 備考欄入力 - 注文の確定

3.3.3 注文詳細画面 (detail.html)

URL: `/ice/detail/<id>/` **目的**: 注文の詳細表示・編集 **機能**: - 注文内容の表示 - 状態の更新 - 完了処理

3.3.4 注文削除画面 (deshap.html)

URL: `/ice/deshap/<id>/` **目的**: 注文の削除 **機能**: - 削除確認 - 注文の完全削除

3.4 かき氷注文管理画面

3.4.1 注文画面 (shavedice.html)

URL: `/shavedice/` **目的**: かき氷注文の受付 **機能**: - フレーバー選択（抹茶、いちご、ゆず） - 注文登録

3.4.2 注文登録画面 (shavedice_register.html)

URL: `/shavedice/register/` **目的**: かき氷注文の詳細登録 **機能**: - 注文内容の確認 - 備考欄入力 - 注文の確定

3.4.3 キッチン画面 (shavedice_kitchen.html)

URL: `/shavedice/kitchen/` **目的**: かき氷キッチンでの作業管理 **機能**: - 注文一覧表示 - 注文状態の更新 - 完了処理

3.4.4 待ち時間画面 (wait_time.html)

URL: `/shavedice/wait-time/` **目的**: 待ち時間の表示 **機能**: - 現在の待ち時間表示 - 注文状況の確認

3.5 モバイル対応画面

3.5.1 注文画面 (mobile_order.html)

URL: `/mobile/order/` **目的**: モバイル端末での注文 **機能**: - タッチ操作に最適化されたUI - 簡潔な注文フロー

3.5.2 完了画面 (mobile_complete.html)

URL: `/mobile/complete/` **目的**: 注文完了の確認 **機能**: - 注文完了メッセージ - 注文番号の表示

4. ユーザーインターフェース設計

4.1 デザイン原則

- **シンプル性**: 必要最小限の情報表示
- **直感性**: 操作が分かりやすいUI
- **一貫性**: 全画面で統一されたデザイン
- **レスポンシブ**: 様々な画面サイズに対応

4.2 カラーパレット

- **プライマリ**: #007AFF（青）
- **セカンダリ**: #FF9500（オレンジ）
- **アクセント**: #FF3B30（赤）
- **背景**: #F2F2F7（ライトグレー）
- **テキスト**: #000000（黒）

4.3 フォント・サイズ

- **見出し**: 24px, 太字
- **本文**: 16px, 通常
- **ラベル**: 14px, 通常
- **注釈**: 12px, 細字

5. 画面遷移の実装

5.1 URL設計

```
# config/urls.py
urlpatterns = [
    path('admin/', admin.site.urls),
    path('food/', include('food.urls')),
    path('ice/', include('ice.urls')),
    path('shavedice/', include('shavedice.urls')),
    path('mobile/', include('mobile.urls')),
]

# food/urls.py
urlpatterns = [
    path('register/', views.food_register, name='food_register'),
    path('kitchen/', views.food_kitchen, name='food_kitchen'),
    path('add-temp/', views.add_temp_food, name='add_temp_food'),
    path('remove-temp/', views.remove_temp_food, name='remove_temp_food'),
    path('complete/', views.complete_food, name='complete_food'),
    path('confirm/', views.confirm_food, name='confirm_food'),
]
```

5.2 ビュー関数

```
def food_register(request):
    """フード注文登録画面"""
    temp_food = request.session.get('temp_food', [])
    context = {
        'temp_food': temp_food,
        'clip_color': request.session.get('clip_color', ''),
        'clip_number': request.session.get('clip_number', ''),
    }
    return render(request, 'food/food_register.html', context)

def food_kitchen(request):
    """フードキッチン画面"""
    active_orders = FoodOrder.objects.filter(is_completed=False)
    completed_orders = FoodOrder.objects.filter(is_completed=True).order_by('-completed_at')[:10]

    context = {
        'active_orders': active_orders,
        'completed_orders': completed_orders,
    }
    return render(request, 'food/food_kitchen.html', context)
```

5.3 テンプレート継承

```
<!-- base.html -->
<!DOCTYPE html>
<html>
<head>
    <title>{% block title %}cafeMuji{% endblock %}</title>
    <link rel="stylesheet" href="{% static 'common/base_cafe.css' %}">
</head>
<body>
    <header>
        <h1>cafeMuji 注文管理システム</h1>
    </header>

    <main>
        {% block content %}{% endblock %}
    </main>

    <footer>
        <p>&copy; 2025 cafeMuji</p>
    </footer>
</body>
</html>

<!-- food_register.html -->
{% extends 'base.html' %}
{% block title %}フード注文登録{% endblock %}
{% block content %}
    <!-- 注文登録フォーム -->
{% endblock %}
```

6. セッション管理

6.1 セッション構造

```
# セッションに保存されるデータ
request.session = {
    'temp_food': [
        {
            'menu': 'からあげ丼',
            'quantity': 2,
            'eat_in': True
        },
        {
            'menu': 'ルーロー飯',
            'quantity': 1,
            'eat_in': False
        }
```

```
    ],
    'clip_color': 'yellow',
    'clip_number': 3
}
```

6.2 セッション操作

```
# 仮注文の追加
def add_temp_food(request):
    if request.method == "POST":
        menu = request.POST.get('menu')
        quantity = int(request.POST.get('quantity', 1))
        eat_in = request.POST.get('eat_in') == '1'

        temp_food = request.session.get('temp_food', [])
        temp_food.append({
            'menu': menu,
            'quantity': quantity,
            'eat_in': eat_in
        })
        request.session['temp_food'] = temp_food

    return redirect('food_register')
```

7. エラーハンドリング

7.1 エラー画面

- **404エラー**: ページが見つからない
- **500エラー**: サーバー内部エラー
- **権限エラー**: アクセス権限がない

7.2 バリデーション

```
def validate_order_data(menu, quantity, eat_in):
    """注文データの妥当性チェック"""
    errors = []

    if not menu:
        errors.append("メニューを選択してください")

    if quantity < 1:
        errors.append("数量は1以上で入力してください")

    if eat_in not in [True, False]:
        errors.append("店内/テイクアウトを選択してください")

    return errors
```

8. レスポンシブ対応

8.1 ブレークポイント

```
/* モバイル */
@media (max-width: 768px) {
    .container {
        padding: 10px;
    }

    .menu-grid {
```

```
        grid-template-columns: 1fr;
    }
}

/* タブレット */
@media (min-width: 769px) and (max-width: 1024px) {
    .menu-grid {
        grid-template-columns: repeat(2, 1fr);
    }
}

/* デスクトップ */
@media (min-width: 1025px) {
    .menu-grid {
        grid-template-columns: repeat(3, 1fr);
    }
}
```

8.2 タッチ操作最適化

```
/* タッチ操作に最適化されたボタン */
.btn {
    min-height: 44px;
    min-width: 44px;
    padding: 12px 24px;
    border-radius: 8px;
    touch-action: manipulation;
}
```

作成日: 2025年8月 **作成者**: 村岡 優次郎 **バージョン**: 1.0