卒業研究報告書

題目

飲食店のオーダーシステムの改善による食品ロスの削減

指導教員

石水 隆　講師

報告者

21-1-037-0157

吉原 大翔

近畿大学理工学部情報学科

令和 7 年 2 月 2 日提出

概要

本研究は,年中無休の焼き鳥屋を対象とし,食品ロス削減と業務効率向上を目的とした発注管理システムを開発するものである.特に,焼き鳥のもも串を対象として,売上データと在庫情報を基に発注点および発注量を計算するシステムを設計した.このシステムはNode.jsとSQLを使用し,「発注点発注」のアルゴリズムに基づいて以下の主要な要素を考慮している.

- 現在の在庫（curStock）:現在店舗内に保管されている在庫数.
- 有効在庫（effStock）:消費期限内で使用可能な在庫数.
- 安全在庫(safStock):突発的な需要増加や納品期間遅延に備えるために確保する最低限の在庫数.
- 最大在庫（maxStock）:店舗が保管できる最大限の在庫数.
- 1日の平均使用量(dailyUsage):過去の売上データを基に計算される1日あたりの平均使用量.
- 消費期限（expirationDays）:食材を使える日数.
- 納品期間（leadTime）:発注から納品されるまでにかかる日数.
- 串1本の重量(stickWeight):もも串1本あたりの平均重量.

発注点は「1日の平均使用量 × 納品期間 + 安全在庫」によって計算され,発注量は「最大在庫 − 有効在庫」で決定される.本システムの導入により,無駄な在庫を削減し,適切なタイミングでの発注が可能となることが期待される.また,在庫管理においては各もも串に一意の番号を付与し,消費期限や入庫順を厳密に管理できる仕組みを採用した.本報告書では,本システムの設計と実装,そしてその効果を検証した結果について述べる.

目次

1. 序論

1.1 本研究の背景

現代の日本には食品ロスという社会的問題がある. 日本の食品ロスは年間約 472 万トンにのぼり, これは世界で飢餓に苦しむ人々に向けた食料支援量とほぼ同等と言われている. また, そのうち約 236 万トンが事業によるロスとなっており, 全体の半分を占めている[1]. 食品ロスは地球温暖化にもつながっており, 我々は食品ロスを減らしていく必要がある. 食品ロスの原因の一つに, 外食産業において飲食店が賞味期限内に消費できない量の食材を発注してしまうことが挙げられる. 多くの店舗では, 店員が食材の減り具合や大体の売上を見て食材を発注するため, 準備した食材が品切れになることもあれば, 食材が余って食品ロスになることもある.

このような食品ロスに対して, 売上や在庫状況, 発注数などをシステムとして管理することにより, 無駄のない発注ができると考えられる. 食品ロスを削減する発注方法として「発注点発注」[2]がある. 発注点は, 「1 日の平均使用量×納品期間+安全在庫」により計算される.

1.2 食品ロスに関する既知の結果

食品ロスに関する研究では, セルフフィードバックや競争情報フィードバックが消費者の意識向上や食品ロス削減に効果を持つことが示されている. 2024 年の伊藤優那・小倉加奈代の研究では, 家庭における食品ロス削減を目的とした記録システムを試作し, 個人が自身の食品廃棄状況を把握できるセルフフィードバック機能を導入した. その結果, 利用者の食品廃棄量が減少し, 食品ロスに対する意識の向上が確認された[3].

さらに, 競争フィードバックを組み合わせることで, 他者との比較が行える環境を作り, より積極的な行動変容を促すことが可能であることが示唆された[4]. これらの結果は, 家庭や企業における食品ロス削減のための情報提供や行動促進の手法として有効であると考えられる.

1.3 本研究の目的

本研究の目的は, 店舗の売上データに基づき発注点を計算するシステムを開発し, 食品ロスを削減すると共に作業効率の向上を図ることにある. 実際のシステムは Node. js と SQL を用い, ウェブ上で食料品の発注をサポートする機能を実現するものである. このシステムの導入により, 無駄な在庫を発生させることなく, 食品ロスの削減を目指す.

1.4 本報告書の構成

本報告書は, 次の構成に基づく. 第 1 章は序論とし, 研究の背景と目的を紹介した. 第 2 章は研究内容として, システム開発の詳細を説明し, 結果を画像やチャートにより描画した. 第 3 章は研究結果とその考察について述べる. 最後に, 第 4 章として結論を織り込み, 今後の課題と謝辞を提示する.

2 発注点発注

本章では,食品ロスを削減する発注方法である「発注点発注」について述べる.

2.1 発注点発注の概要

発注点発注とは,在庫があらかじめ設定した「発注点」に達したタイミングで追加発注を行う在庫管理の方法である.この方式を採用することで,在庫切れを防ぎつつ,過剰な在庫を持たない効率的な管理が可能となる.特に食品業界や医薬品業界など,適正な在庫管理を求められる分野で広く利用されてい

る.

2.2 発注点発注のメリット

発注点発注には以下のメリットがある[5].

1. 在庫切れの防止
   発注点と発注量を適切に設定することにより,在庫の最適化を実現する.在庫が設定された発注点に達するたびに自動的に発注が行われるため,在庫切れを未然に防ぐことができる. これにより,必要な在庫量を常に確保し,商品の供給を途切れさせることなく維持することが可能である.

2. 発注の手間とコストの削減
   発注点に達した時点で発注が行われるため,手動での作業が減少する.また,データにより適切な量を発注するため,発注にかかる手間とコストが削減され,効率的な在庫管理が可能となる.

3. 需要変動に対応
   需要が増加した場合でも,在庫が発注点に達するたびに迅速に発注が行われるため,在庫切れを防ぐことができる.また,需要が減少した場合でも,過剰な在庫を抱えるリスクが低くなる.

2.3 発注点発注の導入例

発注点発注の導入例として,小売業界や衣料品業界では POS システムと連携したものが使われている事例が発見された.しかし,飲食業界においては,オーダーシステムと連携した発注点発注のシステムの事例は見当たらず,まだ開発されていないと考えられる.

3 研究内容

本章では,本研究で開発した発注システムについて述べる.

3.1 システム開発の詳細

本研究の対象とする店舗は年中無休の焼き鳥屋であり,今回は検証対象をももの焼き鳥に限定する.以下はシステムが考慮する主要な要素である.

- 現在の在庫（curStock）：現在店舗内に保管されている在庫数.
- 有効在庫（effStock）：消費期限内で使用可能な在庫数.
- 安全在庫（safStock）：突発的な需要増加や納品期間遅延に備えるために確保する最低限の在庫数.
- 最大在庫（maxStock）：店舗が保管できる最大限の在庫数.
- 1 日の平均使用量（dailyUsage）：過去の売上データを基に計算される 1 日あたりの平均使用量.
- 消費期限（expirationDays）：食材を使える日数.
- 納品期間（leadTime）：発注から納品されるまでにかかる日数.
- 串１本の重量(stickWeight)：もも串１本あたりの平均重量.

なお,在庫管理においては各もも串に一意の番号（1～Spre）が付与されており,これにより消費期限や入庫順を管理する. 本研究で開発したシステムは,Node.js と SQL を用いて設計されており,店舗の売上データと在庫情報をもとに発注点および発注量を計算する機能を持つ.このシステムは「発注点発注」のアルゴリズムに基づき,以下の計算式を採用した.

- 発注点（reorderPoint）の計算式
  reorderPoint = (dailyUsage × leadTime) + safStock

- 発注量（orderQuantity）の計算式
  orderQuantity = maxStock - effStock

3.2 本研究で開発したシステム

本研究では,Node.js と SQL を用いて在庫管理システムを開発した.付録に本研究で開発したシステムのソースプログラムを示す.

3.3 ディレクトリー構造

本節では,本研究で開発したシステムのディレクトリ構造について述べる.図 1 に本研究で開発したディレクトリ構造を示す.

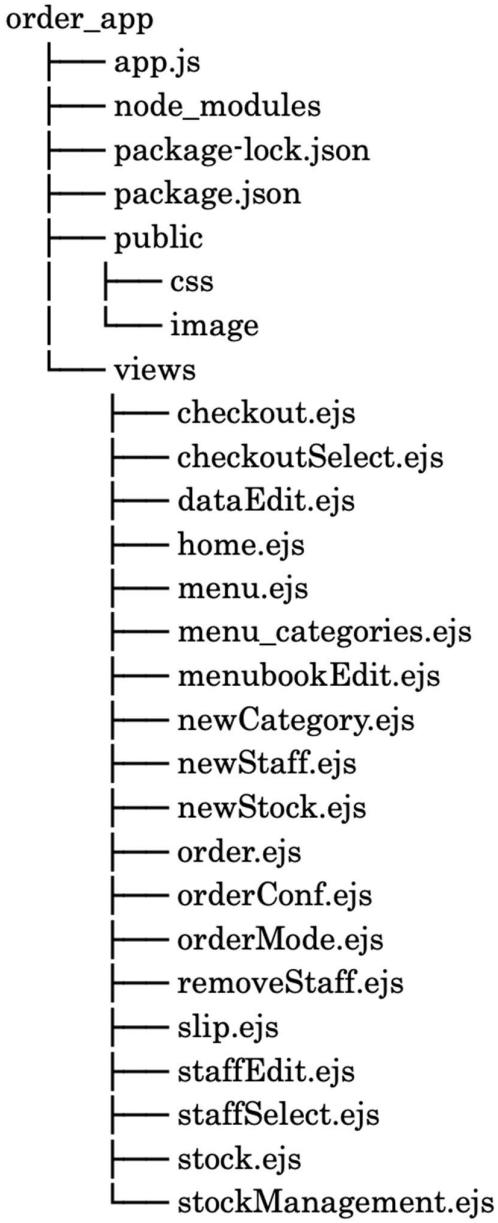

図 1　本研究で開発したシステムのディレクトリ構造

ディレクトリ構造について以下に説明する.

- order_app
  プロジェクトのルートディレクトリで,アプリ全体を管理する.
- app.js
  アプリのエントリーポイントであり,必要なモジュールの読み込みや,フレームワークの設定,ルーティングの設定,サーバーの起動などを行う.また,クライアント（ブラウザ）からのリクエストをルーティングに渡し,適切な処理を実行し,レスポンスとしてEJS をレンダリングしたり,JSON データを返したりする.
- node_module
  Node.js のパッケージが格納されている.

- pakage-lock.json
  インストールしたパッケージのバージョンを固定するためのファイル.
- pakage.json
  プロジェクトの設定ファイルであり,使用するパッケージやスクリプトの情報が含まれている.
- public
  静的ファイル（CSS,画像）を格納するファイル.
  - ✧ css
    CSS ファイルを格納（今回は空）
  - ✧ image
    画像ファイルを格納（今回は空）
- views
  EJS（テンプレートエンジン）を使った画面（HTML）のファイルを格納.それぞれの ejs ファイルが,特定のページに対応している.

3.4 画面遷移図

本節では,本研究で開発したシステムの画面遷移について説明する.図２に本研究で開発したピログラムの画面遷移図を示す.

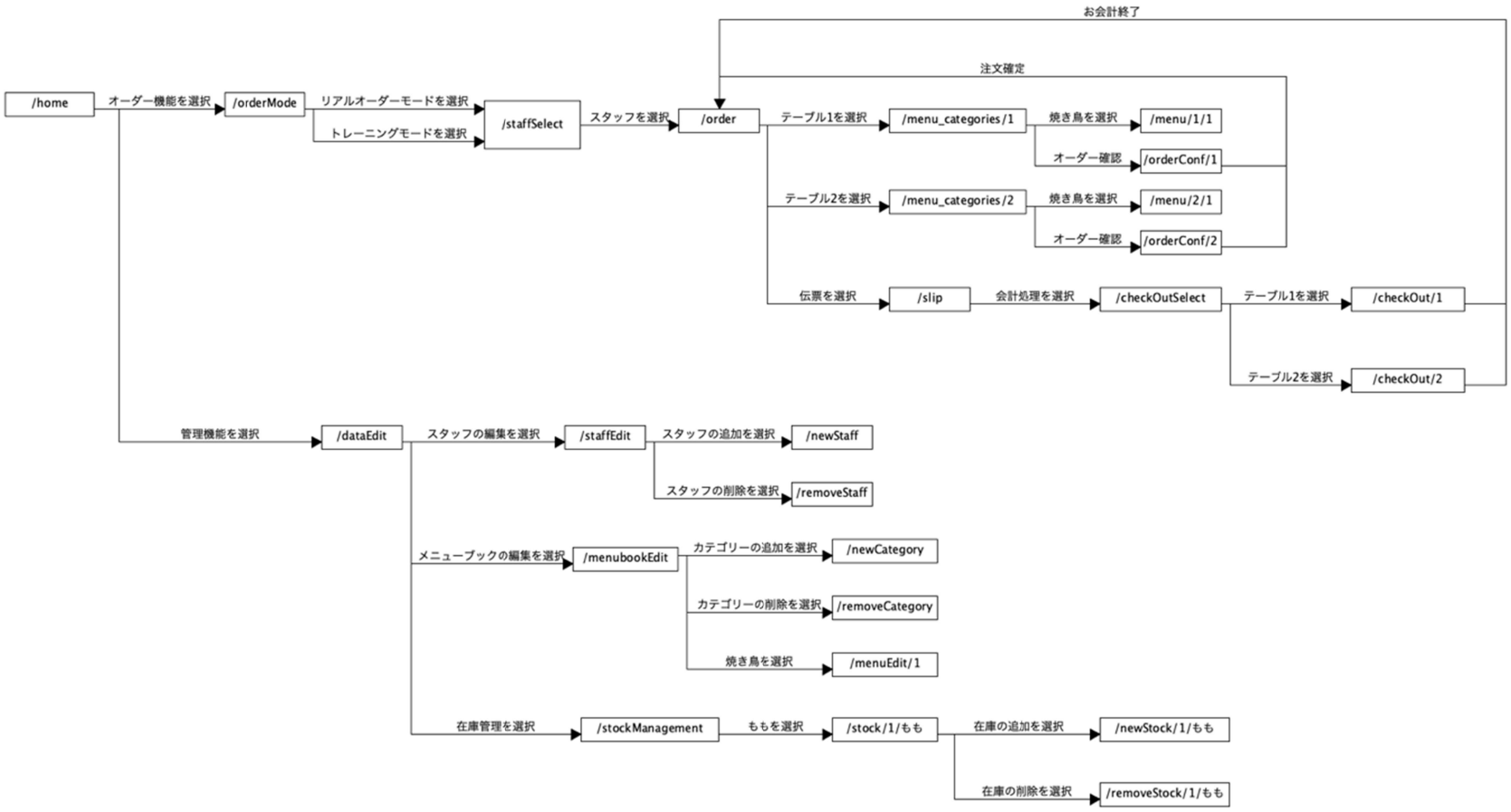

図２　本研究で開発したシステムの画面遷移図

以下に図２に示される画面遷移の詳細について説明する.

- ホーム画面
  - 説明：最初に表示される画面
  - 遷移条件①：オーダー機能を押下する
  - 遷移条件②：管理機能を押下する
  - 遷移先①：モード選択画面
  - 遷移先②：管理機能一覧画面
- モード選択画面
  - 説明：売上データに反映するかしないかを選択する画面(本研究では選択に関わらずデータが変化する)
  - 遷移条件：いずれかのモードボタンを押下する
  - 遷移先：スタッフ選択画面
- スタッフ選択画面
  - 説明：注文を通すスタッフを選択する画面
  - 遷移条件：スタッフを選択し決定ボタンを押下する
  - 遷移先：注文画面
- 注文画面
  - 説明：注文を通すテーブルを選ぶ画面
  - 遷移条件①：いずれかのテーブルボタンを押下する
  - 遷移条件②：伝票ボタンを押下する
  - 遷移先①：カテゴリー画面
  - 遷移先②：伝票機能一覧画面
- カテゴリー画面
  - 説明：メニューのカテゴリーを表示する画面
  - 遷移条件①：いずれかのカテゴリーボタン押下する
  - 遷移条件②：注文確認ボタンを押下する
  - 遷移先①：メニュー画面
  - 遷移先②：注文確認画面
- メニュー画面
  - 説明：選択されたカテゴリー内にあるメニューを表示し,注文を通す画面
  - 遷移条件①：メニューを通す個数を入力し追加ボタンを押下する
  - 遷移条件②：カテゴリー選択に戻るボタンを押下する
  - 遷移先①：メニュー画面
  - 遷移先②：カテゴリー画面
- 注文確認画面
  - 説明：選択したテーブルの注文が確定していないものを表示する画面
  - 遷移条件：注文確定ボタンを押下する
  - 遷移先：注文画面
- 伝票機能一覧画面
  - 説明：会計処理などの機能を選択できる画面
  - 遷移条件①：会計処理ボタンを押下する
  - 遷移条件②：注文ボタンを押下する
  - 遷移先①：お会計テーブル選択画面
  - 遷移先②：注文画面
- お会計テーブル選択画面
  - 説明：会計処理を行うテーブルを選択する画面
  - 遷移条件：いずれかのテーブルボタン押下する
  - 遷移先：注文内容確認画面

- 注文内容確認画面
  - 説明：前回お会計を行ってからこのテーブルに通った注文を表示する画面
  - 遷移条件：お会計終了ボタンを押下する
  - 遷移先：伝票機能一覧画面
- 管理機能一覧画面
  - 説明：管理機能を一覧表示する画面
  - 遷移条件①：スタッフの編集ボタンを押下する
  - 遷移条件②：メニューブックの編集ボタンを押下する
  - 遷移条件③：在庫管理ボタンを押下する
  - 遷移先①：スタッフの編集画面
  - 遷移先②：メニューブックの編集画面
  - 遷移先③：在庫管理画面
- スタッフの編集画面
  - 説明：登録されているスタッフを表示する画面
  - 遷移条件①：スタッフの追加ボタンを押下する
  - 遷移条件②：スタッフの削除を押下する
  - 遷移先①：スタッフの追加画面
  - 遷移先②：スタッフの削除画面
- スタッフの追加画面
  - 説明：スタッフの追加を行う画面
  - 遷移条件：名前を入力し追加ボタンを押下する
  - 遷移先：スタッフの追加画面
- スタッフの削除画面
  - 説明：スタッフの削除を行う画面
  - 遷移条件：削除するスタッフの id を入力し削除ボタンを押下する
  - 遷移先：スタッフの削除画面
- メニューブックの編集画面
  - 説明：編集するカテゴリーを選択,カテゴリーの編集を行う画面
  - 遷移条件①：カテゴリーの追加ボタンを押下する
  - 遷移条件②：カテゴリーの削除を押下する
  - 遷移条件③：いずれかのカテゴリーボタンを押下する
  - 遷移先①：カテゴリーの追加画面
  - 遷移先②：カテゴリーの削除画面
  - 遷移先③：メニューの編集画面（未開発）
- カテゴリーの追加画面
  - 説明：カテゴリーの追加を行う画面
  - 遷移条件：商品名を入力し追加ボタンを押下する
  - 遷移先：カテゴリーの追加画面
- カテゴリーの削除画面
  - 説明：カテゴリーの削除を行う画面
  - 遷移条件：削除する商品の id を入力し削除ボタンを押下する
  - 遷移先：カテゴリーの削除画面
- 在庫管理画面
  - 説明：在庫管理する商品を選択する画面
  - 遷移条件：いずれかの商品名ボタンを押下する
  - 遷移先：在庫画面

- 在庫画面
  - 説明：選択した商品の在庫を表示し, 発注を提案する画面
  - 遷移条件①： 在庫の追加ボタンを押下
  - 遷移条件②： 在庫の削除ボタンを押下
  - 遷移先①：在庫の追加画面
  - 遷移先②：在庫の削除画面
- 在庫の追加画面
  - 説明：発注した商品が到着後, 在庫を追加する画面
  - 遷移条件：追加する個数を入力し追加ボタンを押下する
  - 遷移先：在庫画面
- 在庫の追加画面
  - 説明：賄いや廃棄などで消費した時に在庫から削除する画面
  - 遷移条件：削除する個数を入力し削除ボタンを押下する
  - 遷移先：在庫画面

なお,各状態遷移後は「戻る」ボタンを押すことで１つ前の状態に戻ることができる.

4　結果・考察

本研究で開発したオーダーシステムは,研究内容に示すように,「発注点発注」のアルゴリズムを導入することで,店舗の売上データや在庫状況に基づいた発注提案を行うことができた.システムは,図３に示すように,過去の売上データや消費期限に基づいた発注点を計算し,店舗が必要な食材を効率的に管理することを可能とした.この結果,無駄な在庫を減らし,食品ロスの削減が実現できる可能性が高いと考えられる.

しかし,現在のシステムは,まだ店舗での実装が完了していない段階にある.主な課題としては,値段やスタッフ情報の連携が不足している点や,ユーザーインターフェースのデザイン（CSS によるデザイン）の改善が必要である点が挙げられる.これらを解決することで,実際の店舗運用により適応したシステムになると考える.また,今回の開発はももの焼き鳥に限定したが,他のメニュー（例：ねぎまや唐揚げなど）にも対応できるよう,システムの柔軟性を高める必要がある.これにより,より多彩なメニューに対応した発注提案が可能になると考えられる.

5　結論・今後の課題

本研究の結果,開発したオーダーシステムは,食品ロス削減と業務効率の向上に大きく貢献することが示された.システムは「発注点発注」のアルゴリズムを活用し,店舗の売上データや在庫情報を基にした発注を実現した.この結果,過剰な食材発注を防ぐことができ,食品ロスを減少させるとともに,作業効率の向上やコスト削減にもつながることが確認された.

今後は,より多様な店舗や食材に対応できるようにシステムを拡張し,より広範な適用が可能となるよう改良を加えていく予定である.また,ユーザーインターフェースの改善や,システムの精度向上にも取り組み,実店舗での運用をスムーズに進められるよう努める.

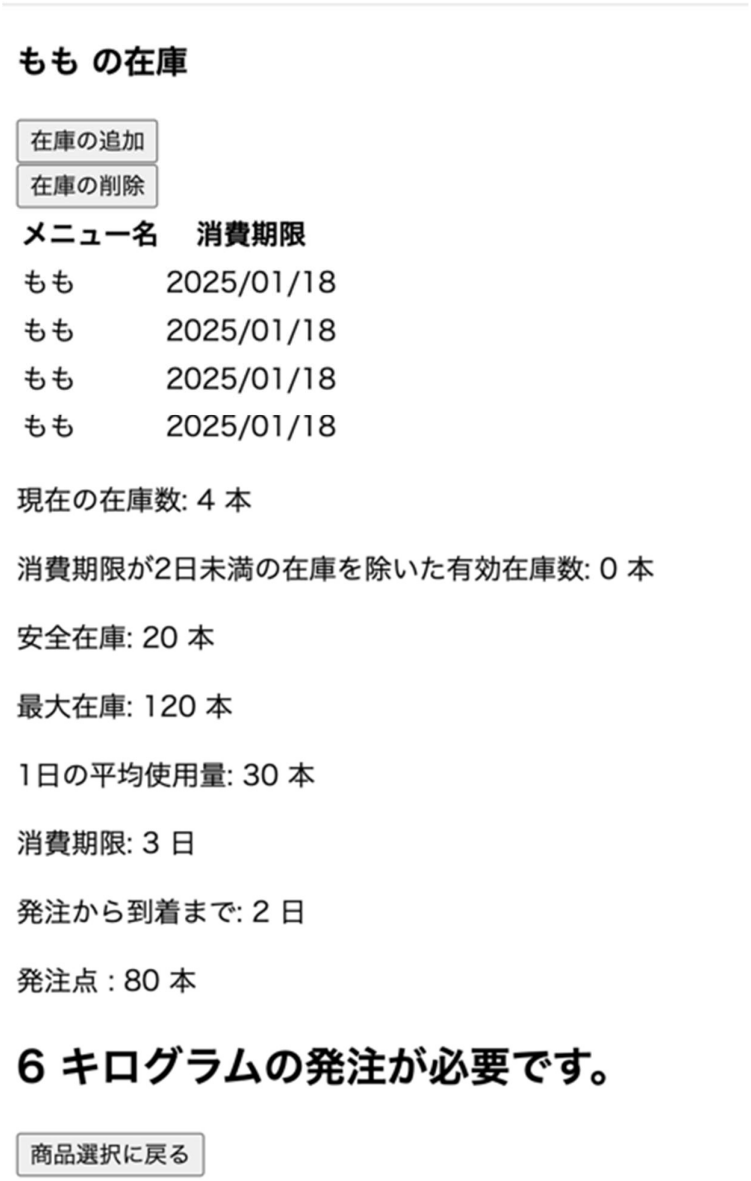

図３　開発したシステムの出力例

謝辞

本研究を進めるにあたり, 貴重なご指導をいただいた石水講師に深く感謝いたします. 講師の専門的なアドバイスと支援により, 本研究を無事に完成させることができました. 心より感謝申し上げます.

参考文献

[1] 食品ロスおよびリサイクルをめぐる情勢<令和 6 年 7 月時点版>,農林水産省(2024) https://www.maff.go.jp/j/shokusan/recycle/syoku_loss/attach/pdf/161227_4-114.pdf

[2] 発注点発注を導入し発注方式を見直して食品ロスを 削減<2023 年 4 月 28 日更新>, https://www.wholesale-vegetable.net/knowledge/order-point.html

[3] 伊藤優那,小倉加奈代:家庭系食品ロス削減に向けたセルフフィードバックを可能とする食品ロス記録システムの試作,情報処理学会 研究報告ヒューマンコンピュータインタラクション,Vol.2024-HCI-206,No.35,pp.1-6,(2024), http://id.nii.ac.jp/1001/00231526/

[4] 伊藤優那,小倉加奈代:セルフフィードバックと競争情報フィードバックによる食品ロスに対する意識向上と食品ロス量削減に関する効果の検証, Vol.2024-HCI-207,No.16,pp.1-8,(2024), http://id.nii.ac.jp/1001/00232875/

[5] 発注点方式・発注点法とは？発注点方式の特徴と発注点や発注量の決定方法,クラウド管理システム zaico,(2024 年 12 月 3 日更新),https://www.zaico.co.jp/zaico_blog/order-point-method/

[6] 嶋田滉平, 山本雄平, 須山敬之:物体の誤検出傾向を用いた画像認識による在庫管理システムの開発, マルチメディア, 分散, 協調とモバイルシンポジウム 2022 論文集, Vol.2022, pp.2-8, 情報処理学会 (2022), http://id.nii.ac.jp/1001/00219467/

[7] 金子格, 湯田恵美, 岡田仁志:オープンデータからのフードロス回収経路産出における巡回エリアサイズの検討, 情報処理学会 研究報告電子知的財産・社会基盤, Vol.2022-EIP-97, No.16, pp.1-6, (2022), http://id.nii.ac.jp/1001/00219464/

[8] 王孟琪, 陳俊文, 柳井啓司:画像認識技術を活用した冷蔵庫内食材自動判別システムの開発, 情報処理学会 第 86 回全国大会講演論文集, Vol.2024, No.1, pp.591-592, (2024), http://id.nii.ac.jp/1001/00236012/

付録

本研究で作成したプログラムのソースプログラムを以下に示す.

app. js

```
const express = require('express');// Express フレームワークをインポート
const mysql = require('mysql2');// MySQL データベースとの接続を行うためのライブラリをインポート

const app = express(); // Express アプリケーションを作成

//静的ファイルを提供
app.use(express.static('public'));
app.use(express.urlencoded({extended: false}));

//データベースの接続設定
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'order_app'
});

//データベースに接続
connection.connect((err) => {
    if (err) {
        console.error('データベースに接続できませんでした: ' + err.stack);
        return;
    }
    console.log('データベースに接続成功！ ID: ' + connection.threadId);
});

//ホーム画面を表示
app.get('/home', (req, res) => {
    res.render('home.ejs');
});

//モード選択画面を表示
app.get('/orderMode', (req, res) => {
    res.render('orderMode.ejs');
});

//スタッフ選択画面を表示
app.get('/staffSelect',(req,res) => {
    connection.query(
        'SELECT * FROM users',
        (error, results) => {
            res.render('staffSelect.ejs', {users: results});
        }
    );
```

```
});

//注文画面を表示
app.get('/order',(req,res) =>{
    connection.query(
        'SELECT * FROM order_tables ORDER BY id ASC',
        (error,results) =>{
            res.render('order.ejs', {tables: results});
        }
    );
});

//カテゴリー画面を表示
app.get('/menu_categories/:tableId',(req,res) =>{
    const tableId = req.params.tableId;
    connection.query(
        'SELECT * FROM menu_categories ORDER BY id ASC',
        (error,results) =>{
            res.render('menu_categories.ejs', {categories: results,tableId: tableId});
        }
    );
});

//メニュー画面を表示
app.get('/menu/:tableId/:categoryId',(req,res) =>{
    const tableId = req.params.tableId;
    const categoryId = req.params.categoryId;

    connection.query(
        'SELECT * FROM menu_items WHERE category_id = ?',
        [categoryId],
        (error,results) =>{
            res.render('menu.ejs', {items: results,tableId: tableId,categoryId: categoryId});
        }
    );
});

//注文を追加するアクション
app.post('/addquantity', (req, res) => {
    const { itemId, quantity, tableId, categoryId} = req.body;

    connection.query(
        'INSERT INTO orders(menu_id, table_id, quantity, status) VALUES(?, ?, ?, ?)',
        [itemId, tableId, quantity, 0],
        (error, result) => {

            console.log(`Item ID: ${itemId}, Table ID: ${tableId}, Quantity: ${quantity}`);
```

```
                res.redirect(`/menu/${tableId}/${categoryId}`);
            }
        );
});

//注文確認画面の表示
app.get('/orderConf/:tableId', (req, res) => {
    const tableId = req.params.tableId;

    connection.query(
        `SELECT o.quantity, m.name
        FROM orders o
        JOIN menu_items m ON o.menu_id = m.id
        WHERE o.table_id = ? AND o.status = 0`,
        [tableId],
        (error, results) => {

            res.render('orderConf.ejs', { orders: results, tableId: tableId });
        }
    );
});

//注文を確定する時のアクション
app.post('/confirmOrder/:tableId', (req, res) => {
    const tableId = req.params.tableId;

    // 注文されたアイテムを取得
    connection.query(
        'SELECT * FROM orders WHERE table_id = ? AND status = 0',
        [tableId],
        (error, orders) => {
            if (error) {
                return res.status(500).send('注文の取得に失敗しました');
            }

            // 注文がない場合
            if (orders.length === 0) {
                return res.redirect(`/orderConf/${tableId}`);
            }

            // 注文の各アイテムに対して在庫処理を行う
            let deleteStockPromise = orders.map((order) => {
                const menuId = order.menu_id;
                const quantity = order.quantity;

                return new Promise((resolve, reject) => {
                    // 在庫を消費期限の早い順に並べ、必要な数量分を減らす
                    connection.query(
```

```
                        'SELECT * FROM stock WHERE menu_id = ? ORDER BY expiration_date ASC
LIMIT ?',
                        [menuId, quantity],
                        (error, stocks) => {
                            if (error) {
                                return reject('在庫の取得に失敗しました');
                            }

                            // 在庫が不足している場合
                            if (stocks.length < quantity) {
                                return reject('在庫が不足しています');
                            }

                            // 在庫を削除する
                            let deletePromises = stocks.map((stock) => {
                                return new Promise((resolve, reject) => {
                                    connection.query(
                                        'DELETE FROM stock WHERE id = ?',
                                        [stock.id],
                                        (error) => {
                                            if (error) {
                                                return reject('在庫の削除に失敗しました');
                                            }
                                            resolve();
                                        }
                                    );
                                });
                            });

                            // 全ての在庫削除が完了したら注文ステータスを更新
                            Promise.all(deletePromises)
                                .then(() => {
                                    connection.query(
                                        'UPDATE orders SET status = 1 WHERE table_id = ? AND status =
0',
                                        [tableId],
                                        (error) => {
                                            if (error) {
                                                return reject('注文のステータス更新に失敗しました');
                                            }
                                            resolve();
                                        }
                                    );
                                })
                                .catch(reject);
                        }
                    );
                });
```

```
                });

                // すべての在庫削除とステータス更新が完了したらリダイレクト
                Promise.all(deleteStockPromise)
                    .then(() => {
                        // 在庫削除後に ID の再連番
                        connection.query(
                            'SELECT MAX(id) AS maxId FROM stock',
                            (error, result) => {
                                if (error) {
                                    return res.status(500).send('最大 ID 取得に失敗しました');
                                }

                                const maxId = result[0].maxId;
                                connection.query(
                                    'ALTER TABLE stock AUTO_INCREMENT = ?',
                                    [maxId + 1],
                                    (error) => {
                                        if (error) {
                                            return res.status(500).send('在庫の ID 再設定に失敗しました');
                                        }

                                        // 最後に注文画面へリダイレクト
                                        res.redirect('/order');
                                    }
                                );
                            }
                        );
                    })
                    .catch((err) => {
                        res.status(500).send(err);
                    });
            }
        );
});

//伝票機能一覧画面を表示
app.get('/slip',(req,res) => {
    res.render('slip.ejs');
});

//お会計テーブル選択画面を表示
app.get('/checkOutSelect',(req,res) => {
    connection.query(
        'SELECT * FROM order_tables',
        (error, results) => {
            res.render('checkOutSelect.ejs', {tables: results});
        }
```

```
    );
});

//注文内容確認画面を表示
app.get('/checkOut/:tableId', (req, res) => {
    const tableId = req.params.tableId;

    connection.query(
        'SELECT orders.menu_id, menu_items.name, SUM(orders.quantity) AS total_quantity FROM orders
JOIN menu_items ON orders.menu_id = menu_items.id WHERE orders.table_id = ? GROUP BY orders.menu_id,
menu_items.name',
        [tableId],
        (error, results) => {
            res.render('checkOut.ejs', { orders: results, tableId: tableId });
        }
    );
});

//お会計を確定するアクション
app.post('/completeCheckout', (req, res) => {
    const tableId = req.body.tableId;

    // orders テーブルから該当テーブル番号のデータを削除
    connection.query(
        'DELETE FROM orders WHERE table_id = ?',
        [tableId],
        (error, results) => {
            res.redirect('/slip');
        }
    );
});

//管理機能一覧画面を表示
app.get('/dataEdit',(req,res) => {
    res.render('dataEdit.ejs');
});

//スタッフの編集画面を表示
app.get('/staffEdit',(req,res) => {
    connection.query(
        'SELECT * FROM users',
        (error, results) => {
          res.render('staffEdit.ejs', {users: results});
        }
    );
});

//スタッフ追加画面を表示
```

```
app.get('/newStaff',(req,res) =>{
    connection.query(
        'SELECT * FROM users',
        (error, results) => {
          res.render('newStaff.ejs', {users: results});
        }
    );
});

//スタッフ追加のアクション
app.post('/addStaff',(req,res) =>{
    connection.query(
        'insert into users(name) values(?)',
        [req.body.userName],
        (error, results) => {
          connection.query(
            'SELECT * FROM users',
            (error, results) => {
              res.render('newStaff.ejs', {users: results});
            }
          );
        }
      );
});

//スタッフ削除画面の表示
app.get('/removeStaff',(req,res) =>{
    connection.query(
        'SELECT * FROM users',
        (error, results) => {
          res.render('removeStaff.ejs', {users: results});
        }
    );
});

//スタッフ削除のアクション
app.post('/deleteStaff',(req,res) =>{
    connection.query(
        'DELETE FROM users WHERE id = ?',
        [req.body.userId],
        (error, results) => {

        }
      );

    connection.query(
        'SET @new_id = 0; UPDATE users SET id = (@new_id := @new_id + 1) ORDER BY id;',
        (error, results) => {
```

```
            }
          );

    connection.query(
        'ALTER TABLE users AUTO_INCREMENT = 1',
        (error, results) => {
            connection.query(
            'SELECT * FROM users',
            (error, results) => {
              res.render('removeStaff.ejs', {users: results});
            }
          );
        }
      );
});

//メニューブックの編集画面の表示
app.get('/menubookEdit',(req,res) =>{
    connection.query(
        'SELECT * FROM menu_categories ORDER BY id ASC',
        (error,results) =>{
            res.render('menubookEdit.ejs', {categories: results});
        }
    );
});

//カテゴリー追加画面の表示
app.get('/newCategory',(req,res) =>{
    connection.query(
        'SELECT * FROM menu_categories ORDER BY id ASC',
        (error, results) => {
          res.render('newCategory.ejs', {categories: results});
        }
    );
});

//カテゴリー追加のアクション
app.post('/addCategory',(req,res) =>{
    connection.query(
        'insert into menu_categories(id,name) values(?,?)',
        [req.body.categoryId,req.body.categoryName],
        (error, results) => {
          connection.query(
            'SELECT * FROM menu_categories ORDER BY id ASC',
            (error, results) => {
              res.render('newCategory.ejs', {categories: results});
            }
```

```
            );
        }
    );
});

//在庫管理画面の表示
app.get('/stockManagement',(req,res) =>{
    connection.query(
        'SELECT * FROM menu_items ORDER BY id ASC',
        (error,results) =>{
            res.render('stockManagement.ejs', {items: results});
        }
    );
});

//在庫画面の表示
app.get('/stock/:itemId/:itemName', (req, res) => {
    const itemId = req.params.itemId;
    const itemName = req.params.itemName;

    // 現在の日付を取得してフォーマット
    const todayFormatted = new Date().toLocaleDateString('ja-JP', {
        year: 'numeric',
        month: '2-digit',
        day: '2-digit',
    });

    connection.query(
        'SELECT * FROM stock WHERE menu_id = ?',
        [itemId],
        (error, results) => {

            // 消費期限を年/月/日形式にフォーマット
            const formattedResults = results.map(stock => {
                const formattedDate = new Date(stock.expiration_date).toLocaleDateString('ja-JP', {
                    year: 'numeric',
                    month: '2-digit',
                    day: '2-digit',
                });
                return {
                    ...stock,
                    expiration_date: formattedDate,
                };
            });

            // 現在の在庫数(数量)を計算
            const currentStock = formattedResults.length;
```

```
            // 消費期限が 2 日未満の在庫を除外して現在の有効在庫数を計算
            const effectiveStock = formattedResults.filter(stock => {
                const expirationDate = stock.expiration_date;
                const dateDifference = (new Date(expirationDate) - new Date(todayFormatted)) / (1000 * 3600 *
24); // 日数差を計算
                return dateDifference >= 2; // 2 日以上の在庫
            }).length;

            // 安全在庫やその他の情報
            const safetyStock = 20;
            const maxStock = 120;
            const averageDailyUsage = 30;
            const expirationDays = 3;
            const leadTime = 2;
            const stickWeightInGrams = 50; // 1 本の重さ

            // 発注点の計算
            let reorderPoint = (averageDailyUsage * leadTime) + safetyStock;

            // 発注が必要かどうかを判定
            let orderMessage = '';
            if (effectiveStock < reorderPoint) {
                // 発注が必要な場合
                let orderQuantityInSticks = maxStock - effectiveStock;
                let orderQuantityInGrams = orderQuantityInSticks * stickWeightInGrams;
                let orderQuantityInKg = orderQuantityInGrams / 1000; // キログラムに変換
                let orderQuantityInKgFloored = Math.floor(orderQuantityInKg); // 1 キロ単位で切り下げ

                orderMessage = `${orderQuantityInKgFloored} キログラムの発注が必要です。`; // 発注メッセージ
            } else {
                orderMessage = '発注の必要はありません。'; // 発注の必要なし
            }

            res.render('stock.ejs', {
                stocks: formattedResults,
                itemId: itemId,
                itemName: itemName,
                currentStock,
                effectiveStock,
                safetyStock,
                maxStock,
                averageDailyUsage,
                expirationDays,
                leadTime,
                reorderPoint,
                orderMessage // 発注メッセージをビューに渡す
            });
        }
```

```
    );
});




app.listen(3000);
```

checkout.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>checkout</title>
        <link rel = "stylesheet" type = "text/css" herf = "checkout.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">>注文確認</h3>
            </header>

            <h4>注文内容:</h4>
                <table>
                    <thead>
                        <tr>
                            <th>メニュー名</th>
                            <th>数量</th>
                        </tr>
                    </thead>
                    <tbody>
                        <% orders.forEach((order) => { %>
                            <tr>
                                <td><%= order.name %></td>
                                <td><%= order.total_quantity %></td>
                            </tr>
                        <% }); %>
                    </tbody>
                </table>

            <a href="/checkOutSelect" style="text-decoration: none;">
                <button id="checkOutSelect_button">戻る</button>
            </a>

            <form action="/completeCheckout" method="POST" style="margin-top: 20px;">
                <input type="hidden" name="tableId" value="<%= tableId %>">
```

```
                    <button type="submit" id="completeCheckout_button">お会計終了</button>
                </form>



            </div>
        </body>
</html>
```

checkoutSelect.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>checkoutSelect</title>
        <link rel = "stylesheet" type = "text/css" herf = "checkoutSelect.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">>お会計テーブル選択</h3>
            </header>

            <p>お会計をするテーブルを選択してください</p>

                <% tables.forEach((table) => { %>
                    <a href="/checkOut/<%= table.id %>" style="text-decoration: none;">
                        <button id="table_button"><%= table.table_name %></button>
                    </a>
                    <br>
            <% }) %>

            <a href="/slip" style="text-decoration: none;">
                <button id="staffEdit_button">戻る</button>
            </a>



        </div>
    </body>
</html>
```

dataEdit.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>dataEdit</title>
        <link rel = "stylesheet" type = "text/css" herf = "dataEdit.css">
```

```
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">管理機能一覧</h3>
            </header>

            <a href="/staffEdit" style="text-decoration: none;">
                <button id="staffEdit_button">スタッフの編集</button>
            </a>

            <br>

            <a href="/menubookEdit" style="text-decoration: none;">
                <button id="menubookEdit_button">メニューブックの編集</button>
            </a>

            <br>

            <a href="/stockManagement" style="text-decoration: none;">
                <button id="menubookEdit_button">在庫管理</button>
            </a>

            <br>



            <a href="/home" style="text-decoration: none;">
                <button id="home_button">ホームに戻る</button>
            </a>

        </div>
    </body>
</html>
```

home.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>home</title>
        <link rel="stylesheet" type="text/css" href="home.css"/>
    </head>
    <body>
        <header>
            <div class="container">
                <h3>オーダーシステム</h3>
```

```
            </div>
        </header>

        <div class="main">
            <div class="container">
                <div class="mode">
                    <p>実行する機能を選択してください。</p>
                    <a class="btn ordermode" href="/orderMode">オーダー機能</a>
                    <a   class="btn dataediting" href="/dataEdit">管理機能</a>
                </div>
            </div>
        </div>

    </body>
</html>
```

menu_categories.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>home</title>
        <link rel="stylesheet" type="text/css" href="home.css"/>
    </head>
    <body>
        <header>
            <div class="container">
                <h3>オーダーシステム</h3>
            </div>
        </header>

        <div class="main">
            <div class="container">
                <div class="mode">
                    <p>実行する機能を選択してください。</p>
                    <a class="btn ordermode" href="/orderMode">オーダー機能</a>
                    <a   class="btn dataediting" href="/dataEdit">管理機能</a>
                </div>
            </div>
        </div>

    </body>
</html>
```

menu.ejs

```
<!DOCTYPE html>
<html>
    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>menu</title>
        <link rel = "stylesheet" type = "text/css" herf = "menu.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">メニュー</h3>
            </header>


            <% items.forEach(item => { %>
                <p><%= item.name %></p>
                <form action="/addquantity" method="post">
                    <input name="quantity" type="number" min="1" placeholder="数量">
                    <input type="hidden" name="itemId" value="<%= item.id %>">
                    <input type="hidden" name="tableId" value="<%= tableId %>">
                    <input type="hidden" name="categoryId" value="<%= item.category_id%>">
                    <input type="submit" value="追加">
                  </form>
                <br>
              <% }); %>


            <a href="/menu_categories/<%= tableId %>" style="text-decoration: none;">
                <button id="menu_button">カテゴリー選択に戻る</button>
            </a>


        </div>
    </body>
</html>
```

menubookEdit.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>menubookEdit</title>
        <link rel = "stylesheet" type = "text/css" herf = "menubookEdit.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">メニューブックの編集</h3>
            </header>
```

```
            <a href="/newCategory" style="text-decoration: none;">
                <button id="categoryEdit_button">カテゴリーの追加</button>
            </a>

            <br>

            <a href="/removeCategory" style="text-decoration: none;">
                <button id="staffEdit_button">カテゴリーの削除</button>
            </a>

            <p>メニューを編集するカテゴリーを選択してください</p>

            <% categories.forEach(category => { %>

                <a href="/menuEdit/<%= category.id %>" style="text-decoration: none;">
                    <button id="menu_button">id:<%= category.id %>/<%= category.name %></button>
                </a>
                <br>

            <% }); %>



            <a href="/dataEdit" style="text-decoration: none;">
                <button id="back_button">戻る</button>

        </div>
    </body>
</html>
```

newCategory.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>newCategory</title>
        <link rel = "stylesheet" type = "text/css" herf = "newCategory.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">カテゴリーの追加</h3>
            </header>

            <div class="item-form-wrapper">
                <form action="/addcategory" method="post">
```

```
                    <span class="form-label">id</span>
                    <input name="categoryId" type="text">
                    <span class="form-label">カテゴリー名</span>
                    <input name="categoryName" type="text">
                  <input href="/newCategory" type="submit" value="追加する">
                </form>
            </div>

            <table>
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>カテゴリー名</th>
                    </tr>
                </thead>

                <tbody>
                    <% categories.forEach((category) => { %>
                        <tr>
                            <td><%= category.id %></td>
                            <td><%= category.name %></td>
                        </tr>
                <% }) %>
                </tbody>
            </table>

            <a href="/menubookEdit" style="text-decoration: none;">
                <button id="back_button">戻る</button>



        </div>
    </body>
</html>
```

newSteff.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>newStaff</title>
        <link rel = "stylesheet" type = "text/css" herf = "newStaff.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">スタッフの追加</h3>
```

```
            </header>

            <div class="item-form-wrapper">
                <p class="form-label">名前を入力</p>
                <form action="/addStaff" method="post">
                  <input name="userName" type="text">
                  <input href="/newStaff" type="submit" value="追加する">
                </form>
            </div>

            <table>
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>名前</th>
                    </tr>
                </thead>

                <tbody>
                    <% users.forEach((user) => { %>
                        <tr>
                            <td><%= user.id %></td>
                            <td><%= user.name %></td>
                        </tr>
                <% }) %>
                </tbody>
            </table>

            <a href="/staffEdit" style="text-decoration: none;">
                <button id="back_button">戻る</button>



        </div>
    </body>
</html>
```

newStock. ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>newStock</title>
        <link rel = "stylesheet" type = "text/css" herf = "newStock.css">
    </head>

    <body>
        <div id="page" style="display: block;">
```

```
            <header>
                <h3 class="title1">スタッフの追加</h3>
            </header>

            <div class="item-form-wrapper">
                <p class="form-label">名前を入力</p>
                <form action="/addStaff" method="post">
                  <input name="userName" type="text">
                  <input href="/newStaff" type="submit" value="追加する">
                </form>
            </div>

            <table>
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>名前</th>
                    </tr>
                </thead>

                <tbody>
                    <% users.forEach((user) => { %>
                        <tr>
                            <td><%= user.id %></td>
                            <td><%= user.name %></td>
                        </tr>
                <% }) %>
                </tbody>
            </table>

            <a href="/staffEdit" style="text-decoration: none;">
                <button id="back_button">戻る</button>



        </div>
    </body>
</html>
```

order.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>order</title>
        <link rel = "stylesheet" type = "text/css" herf = "order.css">
    </head>
```

```
    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">注文</h3>
            </header>

            <a href="/order" style="text-decoration: none;">
                <button id="order_button">注文</button>
            </a>

            <a href="/slip" style="text-decoration: none;">
                <button id="ticket_button">伝票</button>
            </a>

            <a href="/order" style="text-decoration: none;">
                <button id="other_button">その他</button>
            </a>

            <br>

            <p>オーダを受けるテーブルを選択してください</p>

                <% tables.forEach((table) => { %>
                    <a href="/menu_categories/<%= table.id %>" style="text-decoration: none;">
                        <button id="table_button"><%= table.table_name %></button>
                    </a>
                    <br>
            <% }) %>

            <a href="/orderMode" style="text-decoration: none;">
                <button id="staffEdit_button">モード選択へ戻る</button>
            </a>


        </div>
    </body>
</html>
```

orderConf.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>orderConf</title>
        <link rel = "stylesheet" type = "text/css" href = "orderConf.css">
    </head>

    <body>
```

```
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">テーブル  <%= tableId %>  の注文確認</h3>
            </header>

            <h4>注文内容:</h4>
            <% if (orders.length > 0) { %>
                <table>
                    <thead>
                        <tr>
                            <th>メニュー名</th>
                            <th>数量</th>
                        </tr>
                    </thead>
                    <tbody>
                        <% orders.forEach(order => { %>
                            <tr>
                                <td><%= order.name %></td>
                                <td><%= order.quantity %></td>
                            </tr>
                        <% }); %>
                    </tbody>
                </table>
            <% } else { %>
                <p>まだ注文がありません。</p>
            <% } %>

            <a href="/menu_categories/<%= tableId %>" style="text-decoration: none;">
                <button id="back_button">カテゴリー選択に戻る</button>
            </a>

            <form action="/confirmOrder/<%= tableId %>" method="post">
                <button type="submit" id="confirmOrder_button">注文確定</button>
            </form>

        </div>
    </body>
</html>
```

oederMode.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>orderConf</title>
        <link rel = "stylesheet" type = "text/css" href = "orderConf.css">
```

```
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">テーブル <%= tableId %> の注文確認</h3>
            </header>

            <h4>注文内容:</h4>
            <% if (orders.length > 0) { %>
                <table>
                    <thead>
                        <tr>
                            <th>メニュー名</th>
                            <th>数量</th>
                        </tr>
                    </thead>
                    <tbody>
                        <% orders.forEach(order => { %>
                            <tr>
                                <td><%= order.name %></td>
                                <td><%= order.quantity %></td>
                            </tr>
                        <% }); %>
                    </tbody>
                </table>
            <% } else { %>
                <p>まだ注文がありません。</p>
            <% } %>

            <a href="/menu_categories/<%= tableId %>" style="text-decoration: none;">
                <button id="back_button">カテゴリー選択に戻る</button>
            </a>

            <form action="/confirmOrder/<%= tableId %>" method="post">
                <button type="submit" id="confirmOrder_button">注文確定</button>
            </form>

        </div>
    </body>
</html>
```

removeStaff.ejs

```
<!DOCTYPE html>
<html>
    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>removeStaff</title>
        <link rel = "stylesheet" type = "text/css" herf = "removeStaff.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">スタッフの追加</h3>
            </header>

            <div class="item-form-wrapper">
                <p class="form-label">id を入力</p>
                <form action="/deleteStaff" method="post">
                  <input name="userId" type="text">
                  <input href="/removeStaff" type="submit" value="削除する">
                </form>
            </div>

            <table>
                <thead>
                    <th>ID</th>
                    <th>名前</th>
                </thead>

                <tbody>
                    <% users.forEach((user) => { %>
                        <td><%= user.id %></td>
                        <td><%= user.name %></td>
                <% }) %>
                </tbody>
            </table>

            <a href="/staffEdit" style="text-decoration: none;">
                <button id="back_button">戻る</button>



        </div>
    </body>
</html>
```

slip.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>slip</title>
```

```
        <link rel = "stylesheet" type = "text/css" herf = "slip.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">伝票</h3>
            </header>

            <a href="/order" style="text-decoration: none;">
                <button id="order_button">注文</button>
            </a>

            <a href="/slip" style="text-decoration: none;">
                <button id="ticket_button">伝票</button>
            </a>

            <a href="/order" style="text-decoration: none;">
                <button id="other_button">その他</button>
            </a>

            <br>

            <a href="/checkOutSelect" style="text-decoration: none;">
                <button id="checkOut_button">会計処理</button>
            </a>

            <br>

            <a href="/orderMode" style="text-decoration: none;">
                <button id="staffEdit_button">モード選択へ戻る</button>
            </a>

        </div>
    </body>
</html>
```

staffEdit.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>staffEdit</title>
        <link rel = "stylesheet" type = "text/css" herf = "staffEdit.css">
    </head>

    <body>
        <div id="page" style="display: block;">
```

```
            <header>
                <h3 class="title1">スタッフの編集</h3>
            </header>

            <a href="/newStaff" style="text-decoration: none;">
                <button id="staffEdit_button">スタッフの追加</button>
            </a>

            <br>

            <a href="/removeStaff" style="text-decoration: none;">
                <button id="staffEdit_button">スタッフの削除</button>
            </a>

            <table>
                <thead>
                    <th>ID</th>
                    <th>名前</th>
                </thead>

                <tbody>
                    <% users.forEach((user) => { %>
                        <td><%= user.id %></td>
                        <td><%= user.name %></td>
                <% }) %>
                </tbody>
            </table>



            <a href="/dataEdit" style="text-decoration: none;">
                <button id="back_button">戻る</button>

        </div>
    </body>
</html>
```

staffSelect.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>staffSelect</title>
        <link rel="stylesheet" type="text/css" href="staffSelect.css"/>
    </head>
    <body>
        <div id="page" style="display: block;">
            <header>
```

```
                <h3 class="title1">スタッフ選択</h3>
            </header>

            <div class="table-body">
                <% users.forEach((user) => { %>
                    <div class="item-data">
                        <input type="radio" class="name-column" ><%= user.name %></button>
                    </div>
                <% }) %>
            </div>

            <a href="/orderMode" style="text-decoration: none;">
                <button id="back_button">戻る</button>

            <a href="/order" style="text-decoration: none;">
                 <button id="confirm_button">決定</button>
            </a>
        </div>
    </body>
</html>
```

stock.ejs

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>stock</title>
        <link rel = "stylesheet" type = "text/css" href = "stock.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1"><%= itemName %>  の在庫</h3>
            </header>

            <a href="/newStock/<%= itemId %>/<%= itemName %>" style="text-decoration: none;">
                <button id="newStock_button">在庫の追加</button>
            </a>

            <br>

            <a href="/newStock/<%= itemId %>/<%= itemName %>" style="text-decoration: none;">
                <button id="newStock_button">在庫の削除</button>
```

```
            </a>

            <table>
                <thead>
                    <tr>
                        <th>メニュー名</th>
                        <th>消費期限</th>
                    </tr>
                </thead>
                <tbody>
                    <% stocks.forEach((stock) => { %>
                        <tr>
                            <td><%= itemName %></td>
                            <td><%= stock.expiration_date %></td>
                        </tr>
                    <% }); %>
                </tbody>
            </table>

            <p>現在の在庫数: <%= currentStock %> 本</p>
            <p>消費期限が 2 日未満の在庫を除いた有効在庫数: <%= effectiveStock %> 本</p>

            <!-- 安全在庫、最大在庫などの情報 -->
            <p>安全在庫: <%= safetyStock %> 本</p>
            <p>最大在庫: <%= maxStock %> 本</p>
            <p>1 日の平均使用量: <%= averageDailyUsage %> 本</p>
            <p>消費期限: <%= expirationDays %> 日</p>
            <p>発注から到着まで: <%= leadTime %> 日</p>

            <p>発注点 : <%= reorderPoint %> 本</p>

            <!-- 発注量 -->
            <h2><%= orderMessage %></h2> <!-- 発注メッセージを表示 -->


            <a href="/stockManagement" style="text-decoration: none;">
                <button id="stockManagement_button">商品選択に戻る</button>
            </a>


        </div>
    </body>
</html>
```

stockManagement.ejs

```
<!DOCTYPE html>
<html>
    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>stockManagement</title>
        <link rel = "stylesheet" type = "text/css" herf = "stockManagement.css">
    </head>

    <body>
        <div id="page" style="display: block;">
            <header>
                <h3 class="title1">在庫管理</h3>
            </header>

            <p>商品名を選択してください</p>

                <% items.forEach((item) => { %>
                    <a href="/stock/<%= item.id %>/<%= item.name %>" style="text-decoration: none;">
                        <button id="item_button"><%= item.name %></button>
                    </a>
                    <br>
            <% }) %>

            <a href="/dataEdit" style="text-decoration: none;">
                <button id="staffEdit_button">戻る</button>
            </a>


        </div>
    </body>
</html>
```