WebアプリPOSシステム Lv1 要件定義書

マイクロサービスアーキテクチャによるPOSシステム

Version 1.0 | 作成日: 2024年

目次

1. プロジェクト概要

プロジェクト名

WebアプリPOSシステム Lv1

目的

POSとしての最低限機能を提供するWebアプリケーションの構築

対象レベル

Lv1 - 基本機能実装、モダンフレームワーク学習

納期目標

基本機能完成後、Lv2への改修も検討

2. システム概要

システムの特徴

- カメラ付きデバイス（スマートフォン・タブレット）対応
- ブラウザベースのWebアプリケーション
- Microsoft Azure クラウド環境での運用
- マイクロサービスアーキテクチャ採用
- 商品コード読み取りによる商品情報取得
- リアルタイム購入処理

主要機能

商品コード読み取り　**購入リスト管理**　**購入処理**

3. 技術仕様

フロントエンド

Framework: Next.js
Base: React
Language: JavaScript/TypeScript
UI: Responsive Design
Device: Mobile/Tablet対応

バックエンド

API Framework: FastAPI
Language: Python
Platform: Azure App Service
Protocol: HTTPS
Architecture: Microservices

データベース

DBMS: MySQL
Service: Azure Database for MySQL
Type: FlexibleServer
Charset: UTF-8
Connection: SSL対応

4. 機能要件

4.1 商品コード処理機能

機能ID	機能名	詳細
F001	商品コード入力	13桁の商品コードをテキスト入力で受け付ける
F002	商品コード読み取り	カメラを使用したバーコード読み取り機能
F003	商品情報取得	商品コードから商品名・単価を取得して表示

4.2 購入リスト管理機能

機能ID	機能名	詳細
F004	商品追加	取得した商品情報を購入リストに追加
F005	リスト表示	商品名/数量/単価/小計を一覧表示
F006	合計金額計算	購入リスト内全商品の合計金額を自動計算

4.3 購入処理機能

機能ID	機能名	詳細
F007	購入確定	購入リストの内容でデータベースへ取引情報を登録
F008	取引記録	取引テーブルと取引明細テーブルへのデータ保存
F009	処理結果表示	購入完了メッセージと合計金額の表示

5. システム構成

5.1 物理構成

カメラ付きデバイス
スマホ・タブレット

ブラウザ
Webアプリ実行環境

Azure App Service
Next.js + FastAPI

Azure MySQL
FlexibleServer

5.2 論理構成

Microsoft Azure Cloud Platform

フロントエンド層
- Next.js Application
- Azure App Service
- HTTPS通信

API層
- FastAPI Service
- Azure App Service
- RESTful API

データ層
- MySQL Database
- Azure Database for MySQL
- FlexibleServer

6. アーキテクチャ

6.1 マイクロサービスアーキテクチャ

🖥️ Frontend Service
- 商品購入UI提供
- ユーザーインタラクション
- レスポンシブデザイン

⚡ API Service
- DB操作代替機能
- ビジネスロジック実装
- データ処理・変換

🗄️ Data Service
- 商品マスタ管理
- 購入情報保存
- トランザクション管理

6.2 通信フロー

1. ユーザーが商品コードを入力/読み取り
2. Next.js → FastAPI：商品情報検索リクエスト（HTTPS）
3. FastAPI → MySQL：商品マスタ検索クエリ
4. MySQL → FastAPI：商品情報レスポンス
5. FastAPI → Next.js：商品情報JSON応答
6. Next.js：画面表示更新

7. API仕様

7.1 商品マスタ検索API

リクエスト仕様

```
GET /api/products/search
Content-Type: application/json

Parameters:
- code (string): 商品コード（13桁）
```

レスポンス仕様

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "product_id": 12345,
  "code": "1234567890123",
  "name": "商品名",
  "price": 150
}

※商品が見つからない場合はNULL情報を返す
```

処理ロジック

- 1→1：パラメータのコードに一致する商品コードの商品を1件返す
- 1→1：対象外見つからなかった場合はNULL情報を返す
- 商品マスタテーブルからCODEフィールドで検索
- 商品一意キー、商品コード、商品名称、商品単価を返却

7.2 購入処理API

リクエスト仕様

```
POST /api/purchase
Content-Type: application/json

{
  "receipt_code": "レシート担当者コード",
  "store_code": "店舗コード",
  "pos_id": "POS機ID",
  "items": [
    {
      "product_id": 12345,
      "product_code": "1234567890123",
      "product_name": "商品名",
      "price": 150,
      "quantity": 1
    }
  ]
}
```

レスポンス仕様

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "success": true,
  "total_amount": 150,
  "transaction_id": 98765,
  "message": "購入が完了しました"
}

エラー時:
{
  "success": false,
  "error_message": "エラー内容"
}
```

処理ロジック

1. 取引テーブルへ基本情報を登録（取引日時、担当者、店舗、POS機）
2. 取引明細テーブルへ商品情報を登録
3. 合計金額を算出（商品単価 × 数量の合計）
4. 取引テーブルの合計金額を更新
5. 成功・失敗状態と合計金額をフロントへ返却

8. データベース設計

8.1 商品マスタテーブル（products）

カラム名	データ型	制約	説明
PRD_ID	INTEGER	PRIMARY KEY, AUTO_INCREMENT	商品一意キー
CODE	CHAR(13)	UNIQUE, NOT NULL	商品コード（13桁）
NAME	VARCHAR(50)	NOT NULL	商品名称
PRICE	INTEGER	NOT NULL	商品単価

インデックス: CODE フィールドにユニークインデックス設定

8.2 取引テーブル（transactions）

カラム名	データ型	制約	説明
TRD_ID	INTEGER	PRIMARY KEY, AUTO_INCREMENT	取引一意キー
DATETIME	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	取引日時
EMP_CD	CHAR(10)	NOT NULL	レシート担当者コード
STORE_CD	CHAR(5)	NOT NULL	店舗コード
POS_NO	CHAR(3)	NOT NULL	POS機ID
TOTAL_AMT	INTEGER	NOT NULL, DEFAULT 0	合計金額

8.3 取引明細テーブル（transaction_details）

カラム名	データ型	制約	説明
TRD_ID	INTEGER	PRIMARY KEY, FOREIGN KEY	取引一意キー
DTL_ID	INTEGER	PRIMARY KEY	取引明細一意キー
PRD_ID	INTEGER	NOT NULL, FOREIGN KEY	商品一意キー
PRD_CODE	CHAR(13)	NOT NULL	商品コード
PRD_NAME	VARCHAR(50)	NOT NULL	商品名称
PRD_PRICE	INTEGER	NOT NULL	商品単価

外部キー: TRD_ID → transactions.TRD_ID, PRD_ID → products.PRD_ID

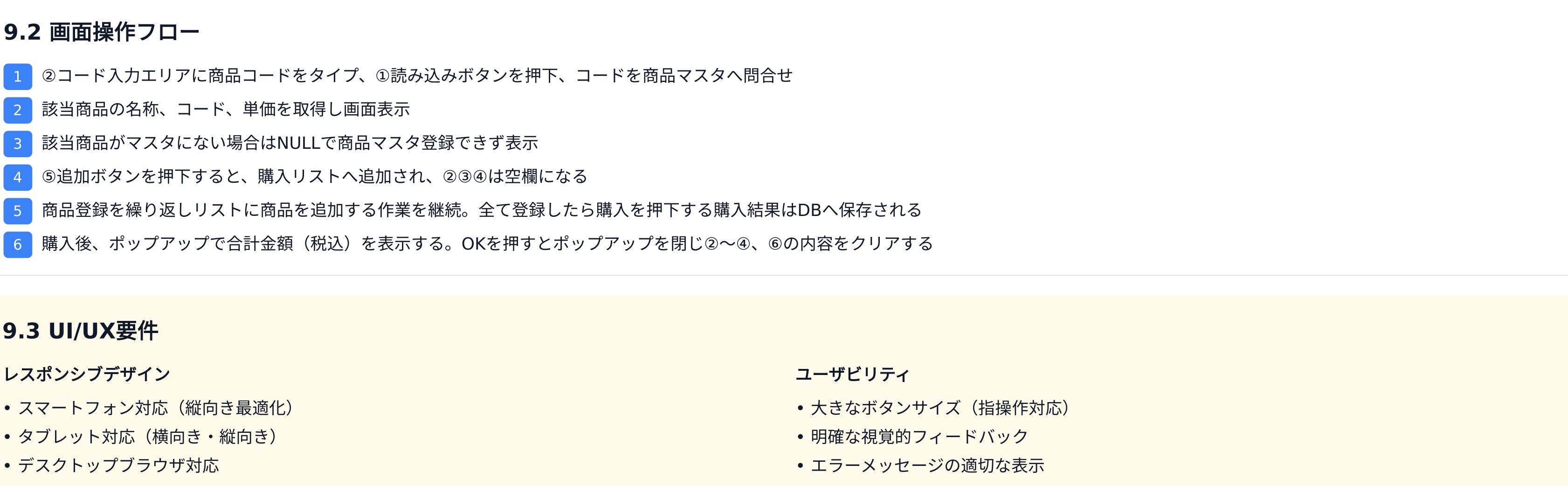
8.4 テーブル関連図

```
products (1) ←——→ (0..*) transaction_details
                                           ↑
                                           |
transactions (1) ←————————————————————————┘
```

products.PRD_ID = transaction_details.PRD_ID（商品情報参照）
transactions.TRD_ID = transaction_details.TRD_ID（取引詳細関連）

9. 画面設計

9.1 メイン画面（POSレジ画面）



商品入力エリア

②コード入力エリア
1234567890

①読み込みボタン

③名称表示エリア
商品名

④単価表示エリア
150円

⑤購入リストへ追加ボタン

購入リスト

⑥購入品目リスト
リフレッシュ x1 300円 300円
無糖要保存する人 x1 188円 188円
タイガー電気プラン新 x1 200円 200円
四つ谷サイダー x1 160円 160円

⑦購入ボタン

9.2 画面操作フロー

1. ②コード入力エリアに商品コードをタイプ、①読み込みボタンを押下、コードを商品マスタへ問合せ
2. 該当商品の名称、コード、単価を取得し画面表示
3. 該当商品がマスタにない場合はNULLで商品マスタ登録できず表示
4. ⑤追加ボタンを押下すると、購入リストへ追加され、②③④は空欄になる
5. 商品登録を繰り返しリストに商品を追加する作業を継続。全て登録したら購入を押下する購入結果はDBへ保存される
6. 購入後、ポップアップで合計金額（税込）を表示する。OKを押すとポップアップを閉じ②〜④、⑥の内容をクリアする

9.3 UI/UX要件

レスポンシブデザイン
- スマートフォン対応（縦向き最適化）
- タブレット対応（横向き・縦向き）
- デスクトップブラウザ対応
- タッチ操作最適化

ユーザビリティ
- 大きなボタンサイズ（指操作対応）
- 明確な視覚的フィードバック
- エラーメッセージの適切な表示
- 操作完了時の確認表示

10. 非機能要件

性能要件

レスポンス時間	商品検索: 2秒以内
購入処理時間	5秒以内
同時接続数	最大10セッション
可用性	営業時間中99%

セキュリティ要件

- HTTPS通信の必須化
- Azure AD認証（将来対応）
- SQLインジェクション対策
- XSS攻撃対策
- データベース接続の暗号化
- ログの適切な管理

拡張性要件

- 商品マスタ：1,000件まで対応
- 1日の取引件数：最大500件
- Azure App Service スケールアップ対応
- データベース容量拡張可能
- マイクロサービス追加容易性

互換性要件

ブラウザ	Chrome, Safari, Edge
モバイルOS	iOS 12+, Android 8+
画面解像度	375px〜1920px
カメラ	フロント・リアカメラ対応

11. 開発・運用要件

11.1 開発環境

IDE: Visual Studio Code推奨
Node.js: v18以上
Python: v3.9以上
データベース: MySQL 8.0
バージョン管理: Git

11.2 デプロイメント

- Azure App Service自動デプロイ
- GitHub Actions CI/CD
- 環境変数による設定管理
- ブルーグリーンデプロイメント
- ロールバック機能

11.3 監視・ログ

- Azure Application Insights
- エラーログの自動収集
- パフォーマンス監視
- ユーザーアクセスログ
- データベース監視

11.4 テスト要件

単体テスト: Jest（Frontend）, pytest（Backend）
結合テスト: API連携テスト
E2Eテスト: Playwright推奨
パフォーマンステスト: Azure Load Testing

12. リスク・課題

高リスク

🔴 **カメラAPI互換性:** 異なるデバイス・ブラウザでのカメラアクセス動作が不安定になる可能性
🔴 **Azure課金:** 想定を超えるリクエスト数による課金額増加のリスク
🔴 **同時接続制限:** 複数ユーザー同時使用時のパフォーマンス低下

中リスク

🟡 **商品マスタ管理:** 商品データの登録・更新機能が未実装（手動SQL操作が必要）
🟡 **エラーハンドリング:** ネットワーク断絶時の挙動が未定義
🟡 **データバックアップ:** 自動バックアップ機能の設定が必要

低リスク・将来課題

🔵 **認証機能:** Lv1では未実装、Lv2以降で検討
🔵 **レシート印刷:** 印刷機能は将来実装
🔵 **多言語対応:** 国際化は将来検討事項
🔵 **在庫管理:** 在庫数管理機能は将来実装

実装まとめ

Lv1 POSシステム 実装スコープ

✅ 必須実装
- 商品コード入力機能
- 商品情報検索API
- 購入リスト管理
- 購入処理・DB保存
- 基本的なレスポンシブUI

🔄 推奨実装
- カメラによるコード読み取り
- エラーハンドリング強化
- ログ機能
- 簡単な認証機能
- テストコード作成

⏭️ 将来実装
- 商品マスタ管理画面
- 売上レポート機能
- 在庫管理機能
- レシート印刷
- 多店舗対応

この要件定義書に基づいて、段階的な開発を進めてください

Lv1完成後は、要求の曖昧さが追加されるLv2、実用性を重視するLv3への発展が可能です

WebアプリPOSシステム Lv1 要件定義書 | Next.js + FastAPI + MySQL | Azure Cloud Platform
本文書は実装の指針として作成されています。詳細な技術仕様は開発過程で調整してください。