卒業研究報告題目

医療資源管理のためのスマートマットを用いた在庫管理システムの試作

Prototype Inventory Management System Using SmartMat for Medical Resource Management

指導教員　田島　孝治　准教授

岐阜工業高等専門学校 電気情報工学科

2018E41　山中　鴻晟

令和 5 年（2023 年）2 月 17 日提出

Abstract

In this study, we developed a prototype inventory management system for medical resource management. The objective of this study is to develop an inventory management system for medical resource management to reduce management costs, ordering errors and the number of immovable inventory.

The developed system works as follows.

1. The user authenticates personally and brings in and out inventory.
2. Inventory detection hardware automatically detects inventory counts and records them in an external database.
3. API retrieves inventory counts stored in an external database.
4. Inventory counts and transactions are recorded in an internal inventory management database.
5. The inventory manager checks the inventory and places the appropriate order.

We evaluated the system in two ways. First, we measured the time it took from the time the card was held up to the time the system responded to the inventory change, varying the number of records. We found that increasing the number of records to 14,000 did not change the search time. We then measured the time from when the card was held to when the system responded to the inventory change, varying the frequency of requests per second. The results showed that the system operated normally up to a maximum of 65 requests per second, confirming that the performance was sufficient for practical use.

目次

1　はじめに

1.1　背景

近年、医療技術の発展に伴い、医薬品をはじめとする多くの医療資源が開発され、利用されている。医療資源は、種類や個数が膨大である。医療薬で言えば、医療用医薬品は約 14,000 品目、一般用医薬品だけでも約 11,400 品目存在する [1]。また、医療用機器では、データベースに登録されている件数が 100 万件 [2] を大きく超えている。先に述べたような医療資源の特徴は、不動在庫の発生や時間的・労力的な管理コストの増加の大きな原因となっている。不動在庫とは、長期間販売・出荷・使用されずに残っている在庫のことであり、在庫の維持や処理のために多くの手間や費用がかかるなどのデメリットが存在する。医薬品の廃棄に伴う損失額は全国の薬局だけで年間 100 億円にも達すると推定されており、院内処方の医薬品も加えると、100 億円をはるかに超えると見込まれている [3]。不動在庫が発生する主な原因としては、過剰発注や発注不足といった発注ミスが挙げられる。また、医療資源に対する管理コストの増加は、発注ミスへとつながりやすく、不動在庫発生の可能性を上げてしまう。

不動在庫を減らすためには、発注ミスを減らすことのできる在庫管理システムが必要である。そこで本研究では、医療資源管理のための在庫管理システムを開発することにより、まずは医療資源に対する管理コストの削減を目指す。管理コストを減少させることは発注ミスの抑制を実現し、さらには不動在庫数の減少にもつながる。従来の在庫管理システムでは、ハンディ端末でバーコードを読み取ることなどにより在庫管理を行なっていたが、今回開発するシステムでは、ユーザが職員証をカードリーダにかざしてから在庫を搬入出するだけで自動的に在庫数の更新ができるようにする。いつ誰がいくつの在庫を搬入出したかをリアルタイムに記録することにより、正確な在庫管理を可能にする。ただし、在庫数を検知する部分については既製品を使用する。

2 関連技術

2.1 Docker

2.1.1 Docker

Docker は、コンテナ型の仮想環境を作成、配布、実行するためのプラットフォームである。VirtualBox などの仮想マシンでは、ホストマシン上でハイパーバイザを利用してゲスト OS を動かし、その上でミドルウェアなどを動かすのに対し、Docker コンテナはホストマシンのカーネルを利用し、プロセスやユーザなどを隔離することで、あたかも別のマシンが動いているかのように動作させる。Docker はミドルウェアのインストールや各種環境設定をコード化して管理するため、ファイルを共有することで誰でも同じ環境が作れたり、環境の配布及びスクラップやビルドが容易であったりする。開発工程の中で利用していた環境をそのまま本番環境に利用することも可能なため、環境差分が少なく、環境の問題を削減できる。[4]

2.1.2 Docker Compose

Docker で複数のコンテナを起動させる必要があるときには、各コンテナを起動させるために、それぞれ起動コマンドやオプションを入力する必要がある。それに対し、Docker Compose では、Docker イメージのビルドや各コンテナ起動のオプションなどを含め、複数のコンテナの定義を yml ファイルに書き、1 つのコマンドで複数のコンテナを一度に操作することができる。そのため、Docker Compose では、複数の Docker コンテナに関する複雑な手順を簡単化することができる。[5]

2.2 RFID

2.2.1 RFID

RFID は、電波を用いて RF タグを読み取ることでデータを読み書きするシステムである。電波が届く範囲であれば、離れていても RF タグを読み取ることが可能であり、複数の RF タグを一括でスキャンすることもできる。また、タグの表面が汚れていても読み取ることができる。[6], [7]

2.2.2 FeliCa

FeliCa は、ソニー株式会社が開発した非接触 IC カード技術方式であり、大きなくくりでは RFID の一種である。Suica をはじめとする交通系 IC カードや、楽天 Edy などの電子マネーなどに採用されている。一枚のカードに IC チップとアンテナを搭載しており、約 0.1 秒でデータの読み書きができる。また、ISO/IEC 15408 EAL5+ 以上を取得しており、高レベルのセキュリティでカード内の情報を保護している。[8]

2.3 Python

Python は 1991 年にオランダ人のグイド・ヴァンロッサムというプログラマによって開発されたプログラミング言語である。オープンソースで運営されており、アプリケーションや人工知能の開発、ビッグデータ解析などのさまざまな用途で使用されている。例えば、YouTube や Instagram のようなサービスでも Python が使われている。Python の特徴としては、簡潔で読みやすいプログラムを書けることや、ライブラリが豊富にあることなどが挙げられる。[9], [10]

2.4 Web アプリケーション

2.4.1 HTML

HTML とは、ハイパーテキスト・マークアップ・ランゲージ（Hyper Text Markup Language）の略であり、Web ページを作成するためのマークアップ言語である。マークアップとは、文章の構成や文章の役割を示すという意味の言葉である。HTML を記述することにより、文章構成をコンピュータに指示することができる。[11], [12]

2.4.2 Bootstrap

Bootstrap とは、HTML/CSS/JavaScript から構成されている Web フレームワークの 1 つである。Web ページでよく使われるフォームやボタン、メニューなどがテンプレートとして用意されており、専門的な知識やスキルがなくてもデザイン性の高い Web サイトを効率的に作成できる。また、Bootstrap で開発した Web ページは PC やスマートフォン、タブレットなどの端末ごとに使いやすいレスポンシブデザインへの対応を自動的に行なってくれる。[13], [14]

2.4.3 jQuery

jQuery とは、JavaScript のためのライブラリである。JavaScript で複雑な記述が必要であった部分でも、jQuery を使えば簡単に実装できるため、世界中の Web デザイナやプログラマの間で広く使われている。例えば、クリックすると画像が入れ替わるスライドショーや、マウスオーバーした要素が動くなどギミックが実装できる。また、jQuery はクロスブラウザ設計のため、Google Chrome や Safari など、ブラウザによる処理方法の違いを気にすることなくコードを実行できる。[15], [16]

2.4.4 PHP

PHP はサーバーサイドのスクリプト言語であり、処理は Web サーバ側で行われる。そのため、PHP では動的な Web ページの開発が可能であり、例えば掲示板や問い合わせフォームなどを実装できる。また、MySQL などのデータベースとの連携が容易であることから、Web アプリケーションの開発に使用されることも多い。[17], [18]

2.4.5 データベース

データベースとは、構造化されたデータの集まりのことである。例えば顧客情報を「氏名」や「電話番号」の項目ごとに整理するなど、データベースでは、大量にあるデータを検索・編集・共有しやすいように保存できる。データを管理・参照する際には、操作性の向上のためにデータベース管理システムを介してデータベースにアクセスする。データベース管理システムは、SQL と呼ばれるデータベース言語を用いて操作する。[19], [20]

2.4.6 SQL

SQL は三つの言語に分けられる。一つ目はデータ定義言語である。データ定義言語では、データベースやテーブルを作成したり、オブジェクト同士の関係を定義したりできる。二つ目はデータ操作言語である。データ操作言語では、データベースを検索したり、データの更新や削除などを行うことができる。三つ目はデータ制御言語である。データ制御言語では、トランザクションやデータへのアクセスを制御することができる。[21], [22]

2.4.7 API

API は、ユーザからのリクエストに対して何らかのレスポンスをすることで、ソフトウェアやプログラム同士をつなぐインタフェースとして機能している。API を利用することのメリットの 1 つとして、他社のデータを利用できることが挙げられる。API に適切なリクエストを送ることによって必要な情報を取得することができるため、さまざまな分野へのデータの活用が見込まれる。[23]

2.4.8 JSON

JSON はもともと JavaScript 上で値を扱うためのフォーマットであったが、バックエンドの普及に伴い Python や PHP などのさまざまな言語でサポートされるようになった。JSON は値と項目のペアで定義するため、階層が深くなっても構造が直感的にわかりやすいという特徴がある。また、JSON はテキスト量が少ないためデータを軽くしやすく、高速な通信を実現しやすい。[24], [25]

3　研究概要

3.1　目的

本研究では、医療資源管理のための在庫管理システムの開発により、管理コストの削減、発注ミスの抑制、不動在庫数の減少の実現を目指す。具体的には、在庫数をリアルタイムに取得し、搬入出の記録を残すことにより、いつ誰がその医療資源を持ち出したのかを把握できるシステムを目指す。在庫数の検知にはスマートマットという製品の利用を想定している。スマートマットは重量センサを搭載した IoT 機器であり、上に物を置くだけで残量を自動計測することができる。検知した在庫数はスマートマットクラウドに自動的に保存される仕組みとなっている。スマートマットは製品の性質上、外部ネットワークに接続する必要がある。

ここで課題となるのが、システム導入の容易化とデータの安全性の確保である。システム導入の容易化では、可能な限り普段の業務を妨げないようにすることが大切である。また、既存の製品であるスマートマットとどのように連携するかが重要となってくる。データの安全性の確保では、個人情報や薬品の情報など、外部に流出してはいけない情報を扱うため、データの安全性を高める必要がある。

3.2　システム概要

本研究で作成するシステムの概要を Fig. 3.1 に示す。

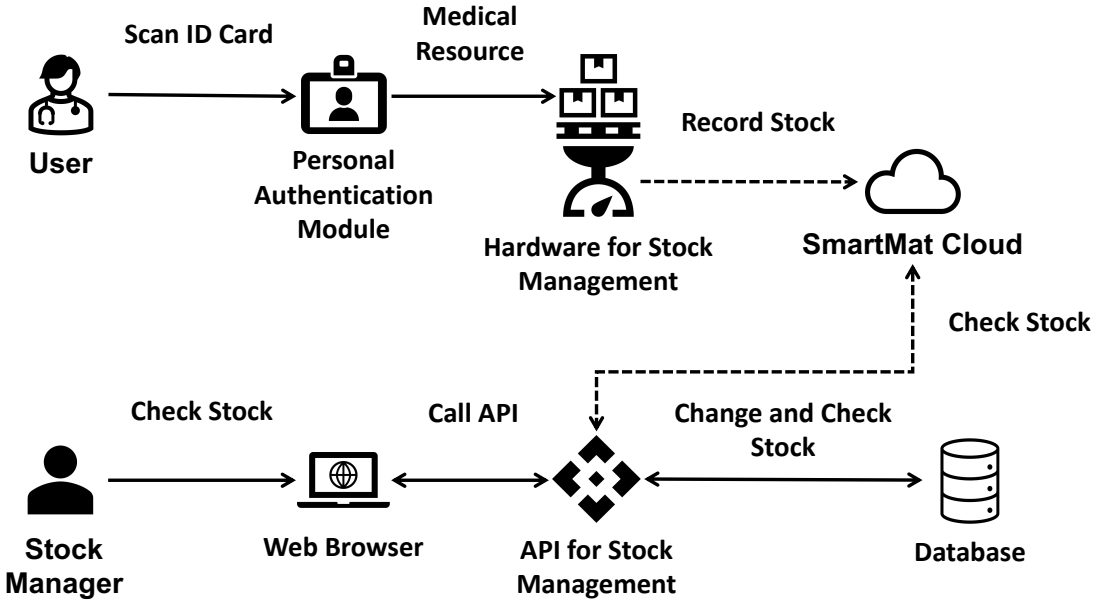

Fig. 3.1. The system overview.

システムは、具体的には、次のような流れで動作する。

1. ユーザ（看護師や医師、納入業者）が個人認証と入出庫をする。
2. 在庫検知用ハードウェアが在庫数を自動検知し外部データベース（スマートマットクラウド）に記録する。
3. API が外部データベースに自動保存された在庫数を取得する。
4. 在庫管理用データベースに在庫数やトランザクションを記録する。
5. 在庫管理者が在庫を確認し、適切な発注をする。

4　設計

4.1　設計方針

本システムは「ユーザが普段の業務の流れの中で在庫を取り出す際に職員証をカードリーダにかざすだけで適切な在庫管理を実現する」という設計方針のもと設計を行った。システム全体は、次の 5 つの要素から構成される。

1. 医療資源の在庫を管理するためのデータベース（在庫管理用データベース）
2. 在庫管理用データベースを操作するための API（在庫管理用データベース操作 API）
3. ユーザを識別するための仕組み（個人認証モジュール）
4. 自動検知された在庫数を記録するためのデータベース（スマートマット用データベース）
5. スマートマット用データベースを操作するための API（スマートマット API）

さらに、システムを内部ネットワークと外部ネットワークに分ける。内部ネットワークには、在庫管理用データベースと在庫管理用データベース操作 API が、外部ネットワークにはスマートマット用データベースとスマートマット API を接続する。今回は、Docker を利用して仮想環境を構築し、その仮想環境の中に在庫管理用データベース及び在庫管理用データベース操作 API を置く。それにより、在庫管理用データベース及び在庫管理用データベース操作 API をネットワーク的に独立させ、データの漏洩を防ぎ、安全性を高める。また、Docker による内部ネットワークの仮想化によって、物理的なサーバやネットワークの設置・維持にかかる費用を削減することができるため、資金力のない小規模の病院での利用も実現する。スマートマット API はグローバル IP を持たせた仮想マシン上に設置し、内部ネットワークとは別の独立した環境として動作させる。

4.2　制限事項

今回のシステム開発では、スマートマットの代わりに、スマートマットクラウドを模したデータベース（スマートマット用データベース）と、スマートマットクラウドから在庫数を取得するスマートマット API を開発する。つまり、スマートマット API は内部ネットワークと外部ネットワーク間のインタフェースとして実装する。このようにすることで、スマートマットクラウドの仕様に変更があった時でも、インタフェースであるスマートマット API の実装を変更するだけで対応可能となる。

また、本システムでは、個人認証のためにカードキーを利用する。今回は、岐阜工業高等専門学校の学生証（FeliCa 対応）を利用し、学生証の中に格納されている学籍番号のデータを用いて個人を特定する。カードキーは職員証として利用されており、ユーザ側の動作はカードをかざすだけであるため、使用上の負担が少ないことがメリットである。

4.3 在庫管理用データベース

在庫管理用データベースは、医療資源の名前や在庫数、部屋名、ユーザ、スマートマットの情報を持たせる。在庫管理用データベースは、Items、Rooms、Stocks、Users、Smartmats、Transactionsというテーブルを持たせる。Items テーブルの構造を Table. 4.1 に示す。Items テーブルには、医薬品などの医療資源の情報を保存する。Rooms テーブルの構造を Table. 4.2 に示す。Rooms テーブルには、部屋の名前を保存する。Stocks テーブルの構造を Table. 4.3 に示す。Stocks テーブルには、スマートマットの ID と在庫数の情報を保存する。Users テーブルの構造を Table. 4.4 に示す。Users テーブルには、ユーザ名を保存する。内部データベース内の Smartmats テーブルの構造を Table. 4.5 に示す。Smartmats テーブルには、smartmat_id、item_id、room_id の情報を保存する。Transactions テーブルの構造を Table. 4.6 に示す。Transactions テーブルには、トランザクションの情報を保存する。具体的には、ユーザ名と行なわれた操作のタイプに加え、医療資源の ID、部屋 ID、在庫数を保存することで、操作した人と内容を保存する。在庫管理用データベース操作 API は内部ネットワーク内に置き、外部からはアクセスできないようにすることでデータの安全性を高める。

Table 4.1. The data structure of Items table.

Data	Type	Description
item_id	bigint	item id
item_name	varchar	item name
created_at	datetime	created time
updated_at	datetime	updated time

Table 4.2. The data structure of Rooms table.

Data	Type	Description
room_id	bigint	room id
room_name	varchar	room name
created_at	datetime	created time
updated_at	datetime	updated time

Table 4.3. The data structure of Stocks table.

Data	Type	Description
smartmat_id	bigint	smartmat id
stock	smallint	number of stocks
created_at	datetime	created time
updated_at	datetime	updated time

Table 4.4. The data structure of Users table.

Data	Type	Description
user_id	bigint	user id
user_name	varchar	user name
registered_at	datetime	registered time

Table 4.5. The data structure of Smartmats table in the internal network.

Data	Type	Description
id	bigint	id
smartmat_id	bigint	smartmat id
item_id	bigint	item id
room_id	bigint	room id
created_at	datetime	created time
updated_at	datetime	updated time

Table 4.6. The data structure of Transactions table.

Data	Type	Description
transaction_id	bigint	transaction id
user_id	varchar	user id (student id)
type	varchar	transaction type
item_id	bigint	item id
room_id	bigint	room id
stock	smallint	number of stocks
transacted_date	datetime	transacted date
memo	text	memo

4.4 在庫管理用データベース操作 API

在庫管理用データベース操作 API は、在庫管理用データベースを操作するための API であり、病院内の医療資源の在庫や、搬入出の記録のために利用される。Items、Rooms、Stocks テーブル用の API には、データの追加、変更、確認、削除の機能を持たせる。Users、Smartmats テーブル用の API には、データの追加、確認、削除の機能を持たせる。Transactions テーブル用の API には、データの追加と確認の機能を持たせる。また、item_id や room_id、stock の情報を持たせることで、どの医薬品がどの部屋から何個持ち出されたのかを特定できるようにした。

在庫管理用データベース操作 API の動作の流れを次に示す。

- 個人認証モジュールからユーザ情報を受け取る。
- スマートマット API にアクセスし、スマートマット用データベース（スマートマットクラウド）から在庫数を取得する。
- 在庫管理用データベースに在庫数とトランザクション（誰が何をいくつ入出庫したのか）を記録する。

在庫管理用データベース操作 API のフローチャートを Fig. 4.1 に示す。

4.5 個人認証モジュール

個人認証モジュールはユーザを識別するためのモジュールである。ユーザの識別により、誰が搬入出したかを記録し特定できるようにする。本研究では RFID により個人認証を行うことを想定しており、IC カードリーダを用いてカード内の情報を読み取る。今回は、岐阜工業高等専門学校の学生証をカードキーとして用いる。学生証の中には一意な値として学籍番号が格納されているため、ユーザの識別には学籍番号を利用する。個人認証モジュールは内部ネットワーク内に置く。

個人認証モジュールの動作の流れを次に示す。

- 学生証がカードリーダにかざされるのを待つ。
- 学生証からユーザ情報（学籍番号）を読み取る。
- 在庫管理用データベース操作 API にユーザ情報を渡す。

個人認証モジュールのフローチャートを Fig. 4.2 に示す。

4.6 スマートマット用データベース

外部データベース内の Smartmats テーブルの構造を Table. 4.7 に示す。スマートマット用データベースは、スマートマットクラウドを模したデータベースであり、Smartmats テーブルに smartmat_id と stock のカラムを持っている。このデータベースは、スマートマットにより自動検知されたと想定される在庫数を記録するために利用する。今回は、Web ページ上から手動で在

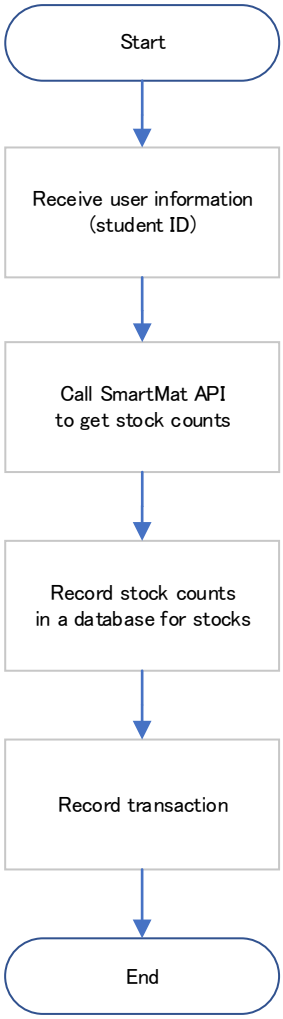

Fig. 4.1. The flowchart of database manipulation API for stock management.

庫数を変更できるようにする。スマートマット API は外部ネットワークと繋がっているスマートマットから内部ネットワーク内に存在しているデータベースへと在庫数を伝えるための役割を担っている。具体的には、スマートマットによってスマートマットクラウド上に記録された残量を取得した後、データベースを操作するための API にアクセスすることで内部データベースの情報を書き換える。スマートマット API も外部ネットワーク上に置く。スマートマット API は在庫管理用データベース操作 API から呼び出されたときのみ、内部ネットワークと通信を行う。

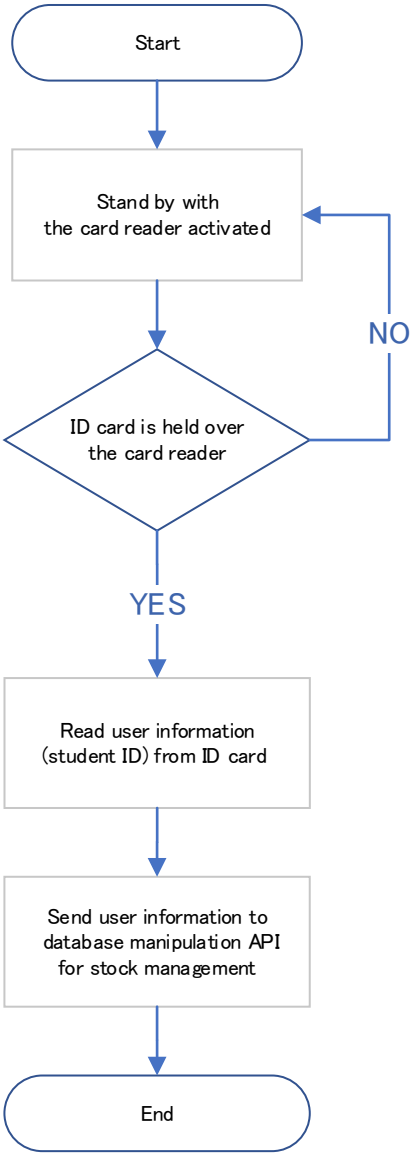

Fig. 4.2. The flowchart of personal authentication module.

Table 4.7. The data structure of Smartmats table in the external network.

Data	Type	Description
smartmat_id	bigint	smartmat id
stock	smallint	number of stocks
created_at	datetime	created time
updated_at	datetime	updated time

4.7　スマートマット API

スマートマット API は、スマートマット用データベースを操作するための API であり、スマートマット用データベース上の在庫数を更新したり、確認したりするために利用する。Web ページ上から在庫数を変更するリクエストが送られてきた際には、スマートマット用データベースの在庫数を変更する。また、在庫数の確認では、在庫数をスマートマット用データベースから取得し、内部ネットワーク内のデータベースに伝える役割を担う。この API にはデータの追加、変更、確認、削除の機能を持たせる。

在庫確認リクエストを受け取った際のスマートマット API の動作の流れを次に示す。

- 在庫管理用データベース操作 API からのリクエストを受け取る。
- スマートマット用データベースから在庫数を取得する。
- 在庫管理用データベース操作 API へ在庫数を返信する。

スマートマット API のフローチャートを Fig. 4.3 に示す。

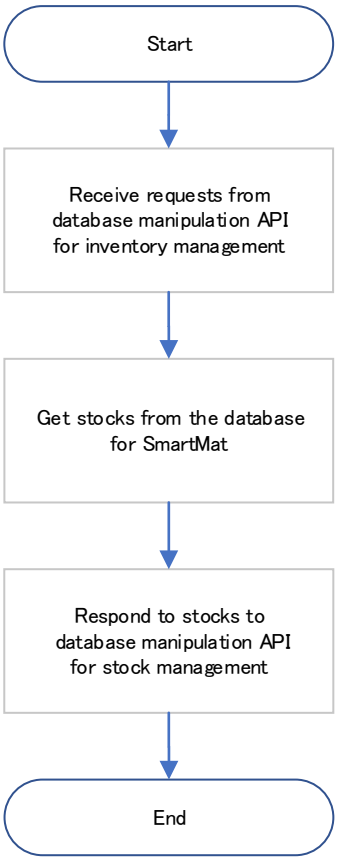

Fig. 4.3. The flowchart of SmartMat API.

5　実装

5.1　開発環境

本システムの開発環境を以下に示す。仮想環境の構築のために Docker を使用した。また、NFC タグを読み込むための Python ライブラリとして、nfcpy を利用した。

- OS：macOS Monterey 12.1
- 言語：HTML, CSS, JavaScript, Python, PHP
- 仮想化ソフトウェア：Docker 20.10.14

5.2　在庫管理用データベース

在庫管理用データベースに実装した Items テーブルの構造を Fig. 5.1 に、Rooms テーブルの構造を Fig. 5.2 に、Stocks テーブルの構造を Fig. 5.3 に、Users テーブルの構造を Fig. 5.4 に、Smartmats テーブルの構造を Fig. 5.5 に、Transactions テーブルの構造を Fig. 5.6 に示す。ここで、Stocks テーブルと Smartmats テーブルの smartmat_id カラムが AUTO_INCREMENT になっていないのは、スマートマットクラウド上に登録されているスマートマット ID を、内部ネットワーク上のデータベースに反映させるためである。

	#	名前	タイプ	照合順序	属性	NULL	デフォルト値	コメント	その他
☐	1	item_id	bigint(20)			いいえ	なし		AUTO_INCREMENT
☐	2	item_name	text	utf8mb4_general_ci		はい	NULL		
☐	3	created_at	datetime			はい	current_timestamp()		
☐	4	updated_at	timestamp			いいえ	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

Fig. 5.1. The data structure of Items table.

	#	名前	タイプ	照合順序	属性	NULL	デフォルト値	コメント	その他
☐	1	room_id	bigint(20)			いいえ	なし		AUTO_INCREMENT
☐	2	room_name	text	utf8mb4_general_ci		はい	NULL		
☐	3	created_at	datetime			はい	current_timestamp()		
☐	4	updated_at	timestamp			いいえ	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

Fig. 5.2. The data structure of Rooms table.

	#	名前	タイプ	照合順序	属性	NULL	デフォルト値	コメント	その他
☐	1	smartmat_id	bigint(20)			はい	*NULL*		
☐	2	stock	smallint(6)			はい	*NULL*		
☐	3	created_at	datetime			はい	current_timestamp()		
☐	4	updated_at	timestamp			いいえ	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

Fig. 5.3. The data structure of Stocks table.

	#	名前	タイプ	照合順序	属性	NULL	デフォルト値	コメント	その他
☐	1	user_id	bigint(20)			いいえ	*なし*		AUTO_INCREMENT
☐	2	user_name	varchar(16)	utf8mb4_general_ci		はい	*NULL*		
☐	3	registered_at	datetime			はい	current_timestamp()		

Fig. 5.4. The data structure of Users table.

	#	名前	タイプ	照合順序	属性	NULL	デフォルト値	コメント	その他
☐	1	smartmat_id	bigint(20)			はい	*NULL*		
☐	2	item_id	bigint(20)			はい	*NULL*		
☐	3	room_id	bigint(20)			はい	*NULL*		
☐	4	created_at	datetime			はい	current_timestamp()		
☐	5	updated_at	timestamp			いいえ	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

Fig. 5.5. The data structure of Smartmats table in the internal network.

	#	名前	タイプ	照合順序	属性	NULL	デフォルト値	コメント	その他
☐	1	transaction_id	bigint(20)			いいえ	*なし*		AUTO_INCREMENT
☐	2	user_id	varchar(50)	utf8mb4_general_ci		はい	*NULL*		
☐	3	type	varchar(50)	utf8mb4_general_ci		はい	*NULL*		
☐	4	item_id	bigint(20)			はい	*NULL*		
☐	5	room_id	bigint(20)			はい	*NULL*		
☐	6	stock	smallint(6)			はい	*NULL*		
☐	7	transacted_date	datetime			はい	current_timestamp()		
☐	8	memo	text	utf8mb4_general_ci		はい	*NULL*		

Fig. 5.6. The data structure of Transactions table.

5.3 在庫管理用データベース操作 API

http://localhost/stock/change に HTTP リクエストが来ると、在庫管理用データベース操作 API が動作する。stock/change.php のフローチャートを Fig. 5.7 に示す。リクエストを受け取ると、初めに必要な情報が全て送られているかを確認する。データが 1 つでも欠けていた場合は、リクエスト元にエラーを返す。次に、データベースへと接続し、データベース上に該当するスマートマットとユーザが存在しているかを確認する。スマートマットまたはユーザが存在しなかった場合は、リクエスト元にエラーを返す。データベース上に該当するスマートマットとユーザの存在を確認した後は、データベース上の在庫数を更新し、その旨をリクエスト元に返信する。最後に、誰がいつ在庫を更新したかをログとして記録し、処理を終了する。

5.4 個人認証モジュール

非接触 IC カードリーダ/ライタである PaSoRi RC-S380 を用いて ID カード内の情報を読み取る。card_reader.py を実行し、カードリーダに学生証をかざすことで、学籍番号を取得することができる。card_reader.py のフローチャートを Fig. 5.8 に示す。カードがかざされると、card_reader.py は http://localhost/main に POST でリクエストを行い、レスポンスを受け取ったらその結果をコンソールに出力する。リクエストの際には、取得した学籍番号のデータを main.php へと渡す。main.php では、スマートマット API と在庫管理用データベース操作 API に HTTP リクエストを行うことで、在庫数の取得と、取得した在庫数の反映を実現させている。main.php のフローチャートを Fig. 5.9 に示す。main.php では、card_reader.py から受け取った学籍番号を利用し、smartmat/check に POST リクエストを行う。smartmat/check からのレスポンスで在庫数を取得すると、stock/change に POST リクエストを行う。stock/change からのレスポンスを受け取ると、card_reader.py にレスポンスを返す。

5.5 スマートマット用データベース

スマートマットは既製品を使用することを想定しているため、実装していない。今回は、Web ページ上から手動で在庫数を変更することによって、スマートマットにより在庫が自動計測されたとみなす。実装した Web ページを Fig. 5.10 に示す。在庫の変更をするためには、「在庫数の変更」というラジオボタンを選択し、smartmat_id と change_to にそれぞれ値を入力してから「CALL API」ボタンを押す。処理結果は「response」の欄に表示される。また、スマートマット用データベースに実装した Smartmats テーブルの構造を Fig. 5.11 に示す。

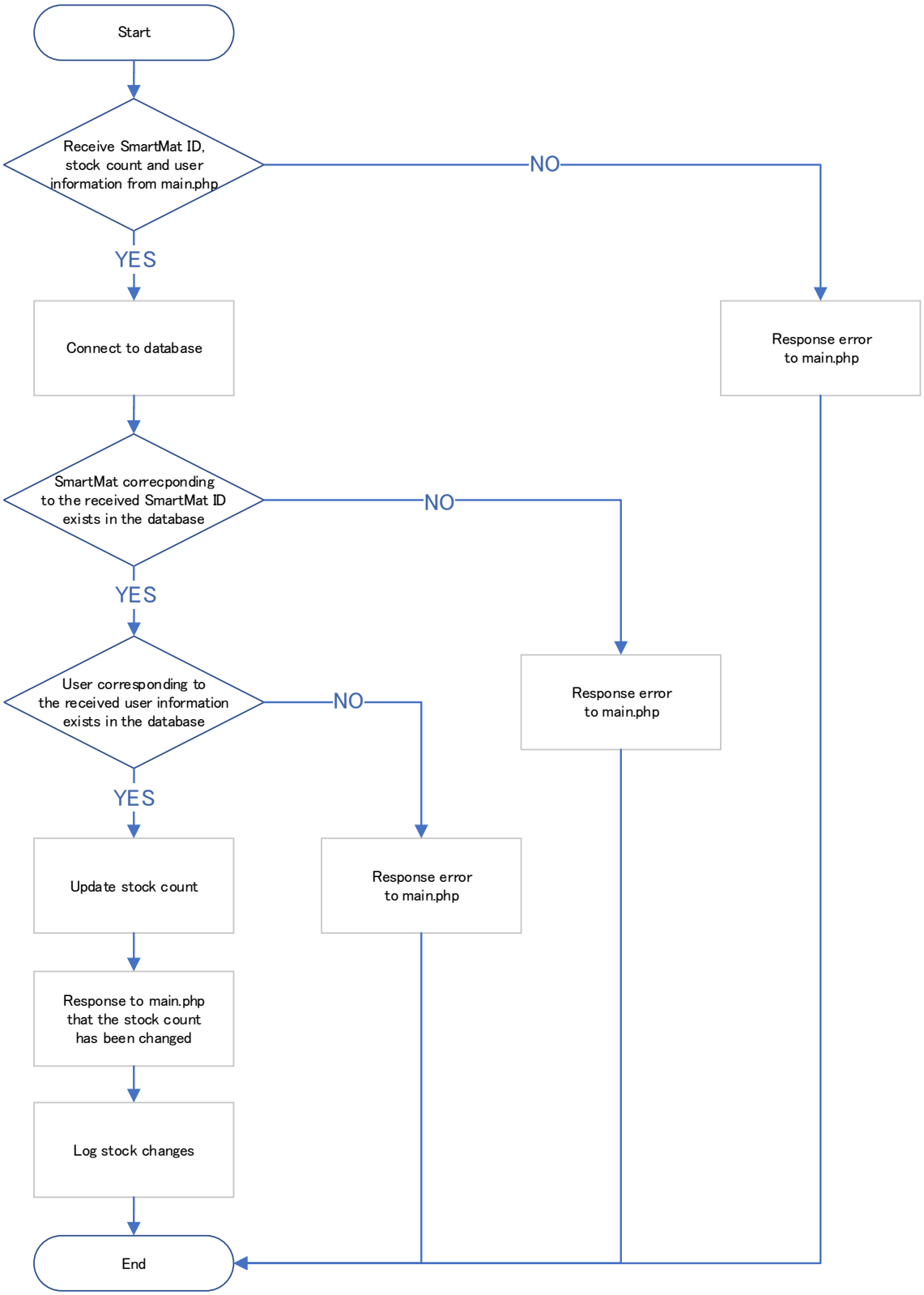

Fig. 5.7. The flowchart of when stock/change.php is accessed.

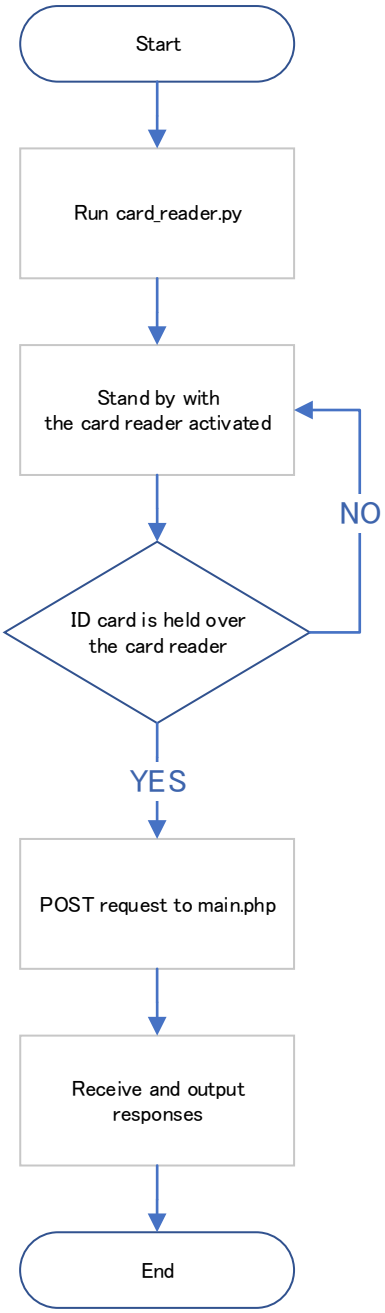

Fig. 5.8. The flowchart of card_reader.py.

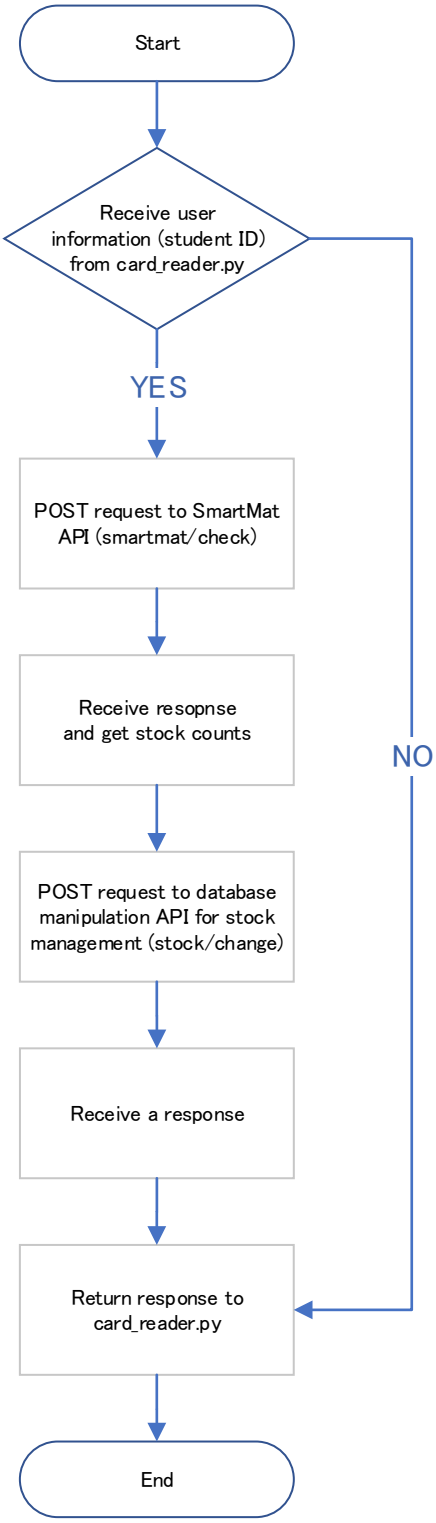

Fig. 5.9. The flowchart of main.php.

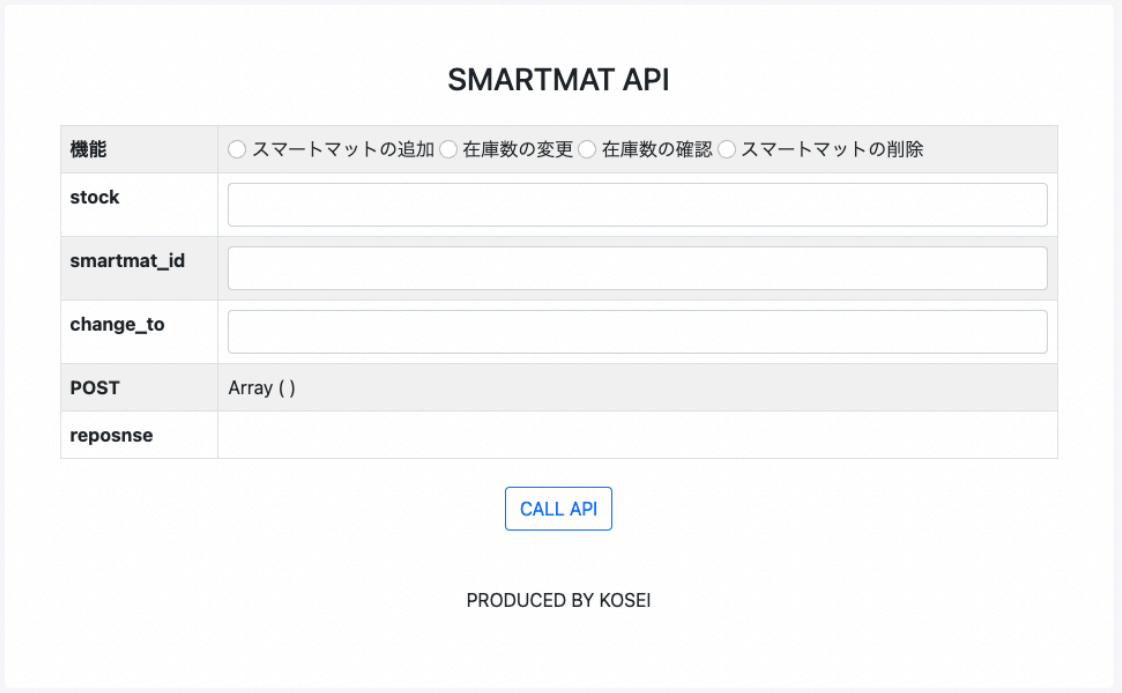

Fig. 5.10. The web page for changing stocks.

	#	名前	タイプ	照合順序	属性	NULL	デフォルト値	コメント	その他
☐	1	id	bigint(20)			いいえ	なし		AUTO_INCREMENT
☐	2	stock	smallint(6)			はい	NULL		
☐	3	created_at	datetime			はい	current_timestamp()		
☐	4	updated_at	timestamp			いいえ	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

Fig. 5.11. The data structure of Smartmats table in the external network.

5.6 スマートマット API

スマートマット API では、https://smartmat.db0.jp/smartmat/check にリクエストが来ると、送られた smartmat_id に対応する在庫数をレスポンスする。smartmat/check.php のフローチャートを Fig. 5.12 に示す。リクエストが来ると、初めにスマートマット ID と新規在庫数と学籍番号が送られてきたかを確認する。もし送られてきていない場合には、リクエスト元にエラーを返す。全ての値を受け取っていたらデータベースに接続し、受け取ったスマートマット ID に該当するスマートマットがデータベース上に存在しているかどうかを確認する。もし該当するスマートマットが存在していない場合には、その旨を返信する。受け取った学籍番号に該当するユーザがデータベース上に存在しているか確認し、存在していたらデータベースの在庫数を更新する。もし

ユーザが存在していない場合には、その旨を返信する。その後、リクエスト元にレスポンスを返し、在庫変更のログを記録し、処理を終了する。

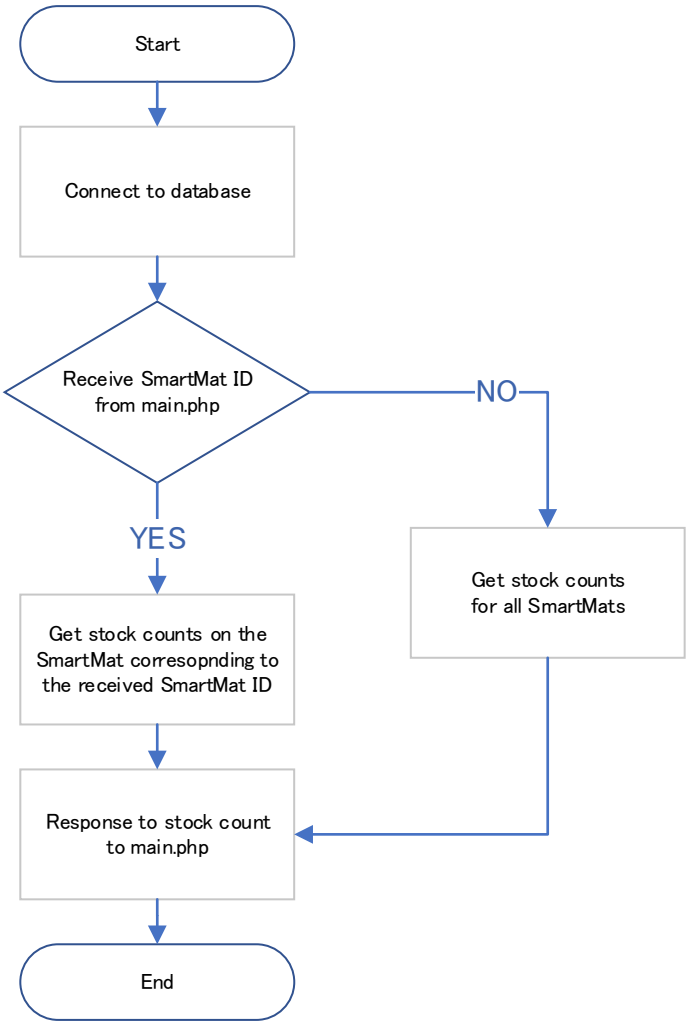

Fig. 5.12. The flowchart of when smartmat/check.php is accessed.

6 評価

6.1 実験方法

今回作成したシステムの性能評価として、カードがかざされてから在庫の変更をレスポンスするまで（main.php で行う処理）にかかる時間の変化を計測する。今回の評価実験では、在庫管理用データベース操作 API の性能評価を主とするが、在庫管理用データベース操作 API にはスマートマット API のリクエスト処理が含まれているため、実験結果は、スマートマット API の処理性能による影響を受ける可能性がある。なお、カードをかざしたときに取得するユーザ情報である職員番号（学籍番号）は固定とする。変化させるパラメータを Table. 6.1 に示す。また、実験環境を Table. 6.2 に示す。

Table 6.1. Parameters used in evaluation experiments.

Experiment No.	Number of records	Frequency of requests
1	500 to 14,000, Increase by 500	10
2	14,000	5 to TIMEOUT, Increase by 5

Table 6.2. The experimental environment.

OS	macOS Monterey 12.1
CPU	Apple M1
Memory	16GB
Load Testing Tool	Vegeta 12.8.4

1 つ目の実験では、外部データベースの Smartmats テーブルと、内部データベースの Stocks テーブルのレコード数を変化させながら、一連の処理にかかる時間を計測する。レコード数は、500 から 14,000 まで 500 ずつ変化させる。なお、1 秒間のリクエスト数は 10 とする。ここで、14,000 という数は、文献 [1] で示された医療用医薬品の品目数を参考にしている。この実験では、レコード数が多くなるほど、つまり扱う薬品数が多くなるほど、在庫取得のための検索スピードが遅くなるのではないかという予想を立てた。

2 つ目の実験では、1 秒間のリクエスト数を変化させながら、一連の処理にかかる時間を計測する。1 秒間のリクエスト数は、5 から 5 ずつ増やしていく。なお、レコード数は 14,000 とする。この実験では、どれだけの数の処理を同時に行うことができるのかを検証する。

6.2 実験結果・考察

6.2.1 評価実験の結果と考察

1 つ目の実験の結果を Table. 6.3 と Fig. 6.1 に示す。Fig. 6.1 は、Table. 6.3 の結果からレコードの数と待ち時間の値を取り出し、グラフにしたものである。Fig. 6.1 を見ると、レコードの数を増やしていったとしても、待ち時間は 100ms から 150ms の間に収まっていることがわかる。つまり、14,000 品目の医療用医薬品を扱うデータベースでは、レコードの数による待ち時間の変化はあまりないといえる。レコードを変化させたときの待ち時間の最大値は、129.152ms であった。この実験では 1 秒間のリクエスト数を 10 としているため、待ち時間を約 130ms とすると、1 リクエストあたり 13ms かかることになる。したがって、1 秒間に行える最大リクエスト数は 76 リクエストとなる。

2 つ目の実験の結果を Table. 6.4 と Fig. 6.2 に示す。Fig. 6.2 は、Table. 6.4 の結果からリクエストの数と待ち時間の値を取り出し、グラフにしたものである。Fig. 6.2 を見ると、どの点も 100ms から 150ms の間に収まっていることがわかる。リクエスト数が 65 を超えると、タイムアウトした。この結果から、理論的な最大リクエスト数は 76 リクエストとなるが、実際には 65 が最大リクエスト数となった。理論値と実測値のギャップについては、実際の通信状況により発生したものであると考えられる。1 つの病院内で 1 秒間に行われるリクエスト数はどちらの値よりも少ないと考えられるため、実用には十分耐えうる性能であるといえる。

6.2.2 テーブル構造についての考察

今回は実装の順番上、設計で示したようなテーブル構造となったが、処理スピードの最大化のためにはテーブル構造を見直す必要があると考えられる。データの整合性を取りやすくなったり、SQL 文の用途がわかりやすくなったりすると考えられるテーブル構造を Fig. 6.3 に示す。Fig. 6.3 では、外部キーを利用して親子関係を明確化したり、各テーブルの要素名をわかりやすくしたりした。特に、データの整合性が取りやすくなることは、データの冗長化を排除するために有効であり、より良い処理スピードを実現できると考えられる。

Table 6.3. The relationship between the number of records and time.

	Duration		
Number of records	Total [ms]	Attack [ms]	Wait [ms]
500	5025	4899	126.545
1000	5009	4898	110.792
1500	5011	4900	111.352
2000	5022	4899	123.219
2500	5029	4900	129.152
3000	5029	4900	128.857
3500	5006	4900	106.404
4000	5013	4899	113.314
4500	5020	4900	119.757
5000	5028	4899	129.076
5500	5029	4900	128.662
6000	5009	4899	109.775
6500	5010	4900	110.459
7000	5006	4900	106.197
7500	5005	4900	105.477
8000	5020	4900	120.743
8500	5013	4899	114.437
9000	5021	4899	121.052
9500	5007	4900	107.304
10000	5029	4900	129.125
10500	5024	4900	124.323
11000	5023	4900	123.386
11500	5017	4901	116.108
12000	5011	4900	110.465
12500	5022	4900	122.639
13000	5029	4901	128.258
13500	5027	4899	127.161
14,000	5012	4899	112.253

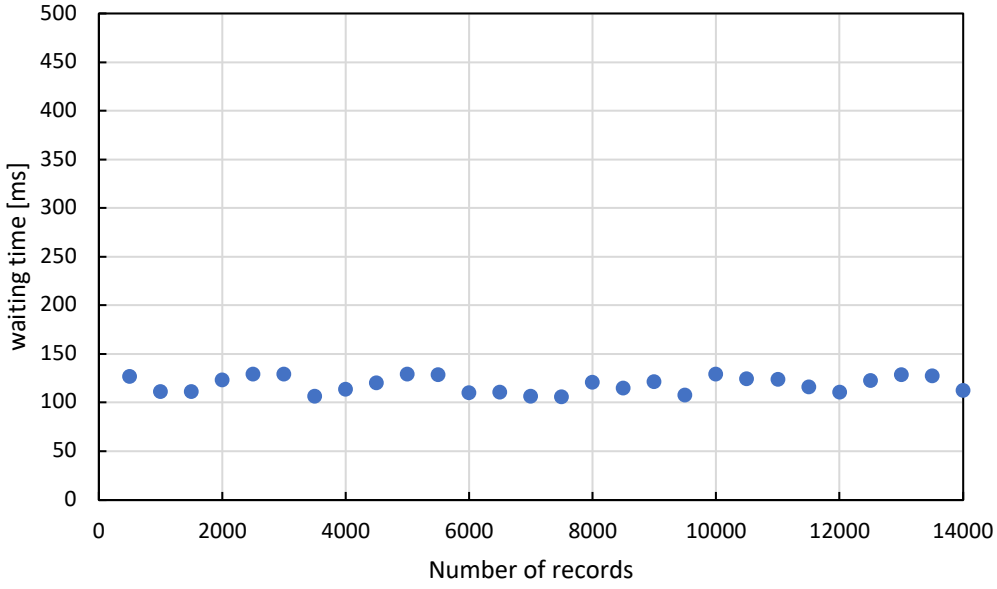

Fig. 6.1. The relationship between the number of records and time.

Table 6.4. The relationship between the frequency of requests and time.

	Duration		
Frequency of requests	Total [ms]	Attack [ms]	Wait [ms]
5	4920	4800	120.362
10	5016	4900	116.338
15	5051	4933	117.989
20	5059	4950	109.124
25	5072	4959	112.866
30	5075	4966	109.048
35	5086	4971	115.371
40	5088	4974	113.303
45	5080	4978	101.801
50	5092	4980	112.803
55	5094	4982	112.371
60	5101	4983	118.308
65	5109	4985	124.749

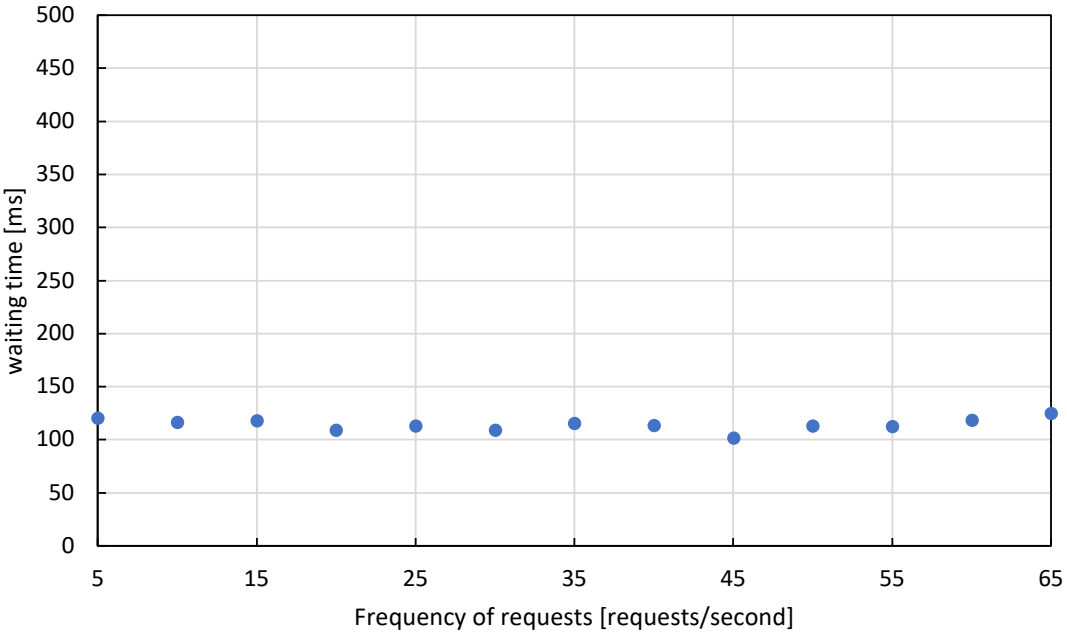

Fig. 6.2. The relationship between the frequency of requests and time.

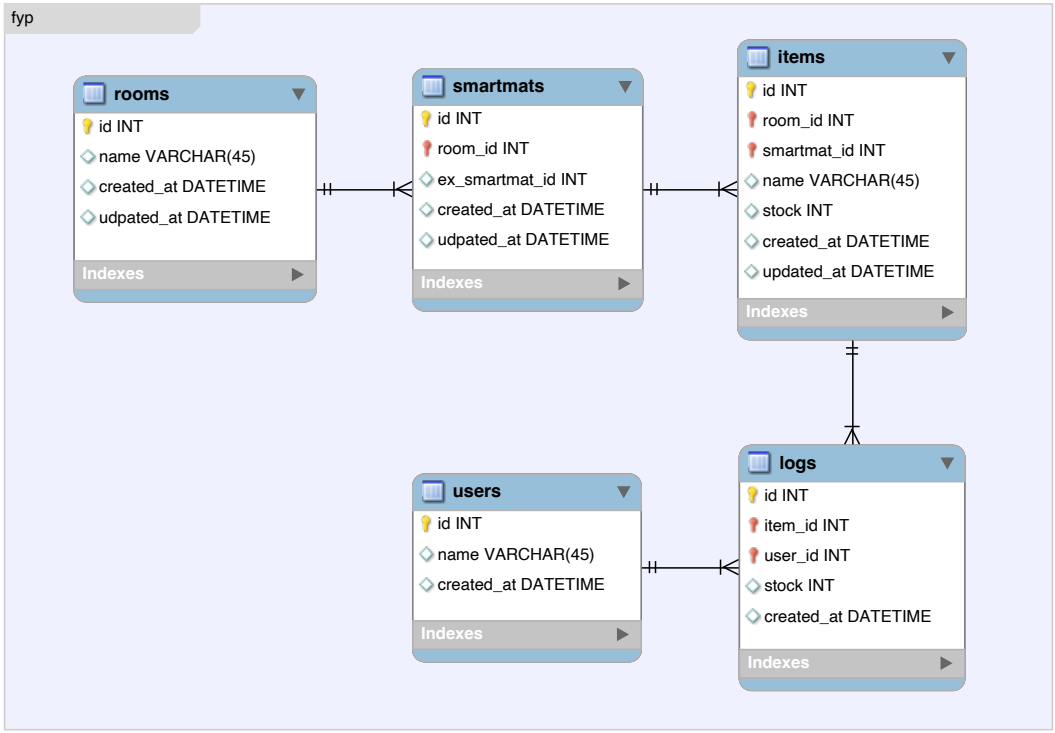

Fig. 6.3. The ideal table structure.

7　まとめ

7.1　研究の成果

本研究では、医療資源管理のためのスマートマットを用いた在庫管理システムを試作した。このシステムは病院内での利用を想定しており、医療資源の数である 14000 品目を扱うデータベースが正常に動作することを確認できた。また、同時アクセスについても、問題のないレベルでの運用ができることを確認できた。

7.2　今後の課題

本システムの課題としては、自ホストに対して HTTP リクエストを行っていることが挙げられる。実際にシステムを利用する際には、そのシステムのネットワークにファイアウォールが設置されているなどする場合がある。このとき、自ホストへの HTTP リクエストは、正常に処理されない可能性がある。このため、自ホストへリクエストをおこなっている部分を、クラスとして定義するなどの対策が必要であると考えられる。

参考文献

[1] 病院の薬、薬局の薬｜おくすり Q&A｜すこやかコンパス｜住友ファーマ株式会社
https://www.sumitomo-pharma.co.jp/sukoyaka/qanda/vol5.html
(Accessed on 2022/12/3)

[2] 医療機器データベース　トップページ
https://www.kikidb.jp/index.cfm
(Accessed on 2022/12/3)

[3] 意外と大きい!?薬剤廃棄ロスが経営に与える影響｜薬局経営 NAVI
https://yk-navi.jp/column/480/
(Accessed on 2023/2/12)

[4] Docker 入門（第一回）〜Docker とは何か、何が良いのか〜｜さくらのナレッジ
https://knowledge.sakura.ad.jp/13265/?gclid=CjwKCAiApvebBhAvEiwAe7mHSArv6l2-YiThhFsgp1emf2pgkAlkrwSrWo0JO6hlYeS7v4hQN7mclhoCIE0QAvD_BwE
(Accessed on 2022/11/23)

[5] Docker 入門（第六回）〜Docker Compose〜｜さくらのナレッジ
https://knowledge.sakura.ad.jp/16862/
(Accessed on 2022/11/27)

[6] RFID とは？ ｜自動認識の技術情報｜デンソーウェーブ
https://www.denso-wave.com/ja/adcd/fundamental/rfid/rfid/index.html
(Accessed on 2023/2/13)

[7] RFID とは？ 基礎から応用までわかりやすく解説｜ RFID Room
https://rfid.tss21.co.jp/knowledge/whatsrfid/
(Accessed on 2023/2/13)

[8] ソニー株式会社｜FeliCa｜FeliCa とは｜FeliCa ってなに？
https://www.sony.co.jp/Products/felica/about/
(Accessed on 2023/2/13)

[9] Python（パイソン）とは？ ｜注目のプログラミング言語を紹介｜コエテコキャンパス
https://coeteco.jp/articles/10661
(Accessed on 2023/2/13)

[10] Python とは？ 大人気プログラミング言語のメリットや活用事例をご紹介
https://www.internetacademy.jp/it/programming/programming-basic/what-is-python.html
(Accessed on 2023/2/13)

[11] 今さら聞けない！ HTML とは【初心者向け】| TechAcademy マガジン
https://magazine.techacademy.jp/magazine/4843

(Accessed on 2022/11/22)
[12] HTML とは？ 初心者向けにタグの種類と使い方の基本を解説！ | Udemy メディア
https://udemy.benesse.co.jp/design/web-design/what-is-html.html
(Accessed on 2022/11/22)
[13] Bootstrap とは？ 特徴や種類、メリット・デメリットを解説 | レンタルサーバーナレッジ
https://knowledge.cpi.ad.jp/cms/bootstrap/
(Accessed on 2023/2/13)
[14] Bootstrap とは？ 意味や特徴、種類を徹底解説 | 侍エンジニアブログ
https://www.sejuku.net/blog/7407
(Accessed on 2023/2/13)
[15] jQuery とは | Web デザイン・Web デザイナー専攻 | デジタルハリウッドの専門スクール(学校)
https://school.dhw.co.jp/course/web/contents/w_jQuery.html
(Accessed on 2023/2/13)
[16]【初心者向け】jQuery とは | メリット・デメリットから記述方法まで解説 | 株式会社パソナ(旧パソナテック) | IT エンジニア・ものづくりエンジニアの求人情報・転職情報
https://www.pasonatech.co.jp/workstyle/column/detail.html?p=2570
(Accessed on 2023/2/13)
[17] PHP とは？ 基礎知識、できることを初心者にもわかりやすく解説します - カゴヤのサーバー研究室
https://www.kagoya.jp/howto/it-glossary/web/php/
(Accessed on 2023/2/13)
[18] PHP とは何かわかりやすく解説！ できることや需要・将来性も紹介 | 侍エンジニアブログ
https://www.sejuku.net/blog/4097
(Accessed on 2023/2/13)
[19] データベースとは | クラウド・データセンター用語集／ IDC フロンティア
https://www.idcf.jp/words/database.html
(Accessed on 2023/2/13)
[20] データベースとは？ 基礎知識を初心者にわかりやすく解説！ | IT トレンド
https://it-trend.jp/database/article/89-0065
(Accessed on 2023/2/13)
[21] SQL とは？ データベース言語の基礎知識をわかりやすく解説！ - システム開発のプロが発注成功を手助けする【発注ラウンジ】
https://hnavi.co.jp/knowledge/blog/sql/
(Accessed on 2023/2/13)
[22]【SQL 入門】データベース言語の基礎知識を学んで MySQ… | Udemy メディア
https://udemy.benesse.co.jp/development/system/intro-sql.html

(Accessed on 2023/2/13)

[23] API とは？ 意味やメリット、使い方を世界一わかりやすく解説 | 侍エンジニアブログ
https://www.sejuku.net/blog/7087
(Accessed on 2023/2/13)

[24] 今さら聞けない JSON とは？ 表記形式や使い方をサンプル付きで解説！ | プログラミングを学ぶならトレノキャンプ（TRAINOCAMP）
https://camp.trainocate.co.jp/magazine/whats-json/
(Accessed on 2023/2/13)

[25] JSON とは？ データフォーマット（データ形式）について学ぼう！
https://products.sint.co.jp/topsic/blog/json
(Accessed on 2023/2/13)

謝辞

本論文は筆者である山中が国立岐阜工業高等専門学校電気情報工学科に在籍中の研究成果をまとめたものである。本研究は多くの方々のご指導、ご協力のもと行われており、その方々の助力無くして本研究は成立しなかった。ここに深く感謝申し上げる。

本研究の指導教員である田島孝治准教授には、研究の提案からその詳細、執筆活動に渡るまで、多くの助言と細やかなご指導をいただいた。また、研究、開発に必要な設備、環境を提供していただき、より良い方向へと進むよう、多くの助言をいただいた。ここに心から感謝の意を示す。

本論文の副査である出口利憲教授には論文を精読していただき、細やかなご指導をいただいた。ここに深く感謝の意を示す。

同研究室に配属された黒﨑椎真氏、戸松準貴氏、西倉有晟氏の 3 人に加え、本研究室の先輩である苫米地康太氏、西口丈二氏、堀壮吾氏には本研究に関して様々な助言をいただいた。ここに深く感謝の意を示す。