

卒業研究報告書

題目

情報倫理教育における e ラーニングのための プラットフォームの開発

指導教員

井口 信和 教授

報告者

17-1-037-0054

栗岡 陽平

近畿大学理工学部情報学科

令和 XX 年 Y 月 Z 日提出

概要

総務省が令和元年に 37182 人に行った調査によると、インターネットの利用率は、その約 9 割にまで増加している [1]. その理由の 1 つに SNS の普及が挙げられる. SNS に関して平成 27 年にみずほ情報総研株式会社が 1178 人に行った調査研究 [2] によると、SNS 上でトラブルの経験があると回答した割合は 15% 程であった. トラブルの内容は、自分自身の発言を他人が異なる意味で受け取ってしまう、自分の意志とは関係なく個人情報などが第三者に公開されてしまうなどである. このようなトラブルを避けるために情報倫理教育は有効な手段の一つである [3].

さらに、新しい情報倫理の問題に対応するために、個人の継続的な学習が要求される [4]. しかし、従来の書籍による学習では、新しい問題への速やかな対応が難しいため、学習の継続が難しい場合がある. 他方、e ラーニング学習は、学習コンテンツの更新が容易なため、学習の継続が期待できる [5].

そこで本研究では、持続的な学習環境を提供することを目的に、情報倫理教育における e ラーニングのためのプラットフォーム（以下、本プラットフォーム）を開発する. まず、本稿ではコンテンツの作成、統計情報の確認、外部アプリケーションの導入を補佐する機能を開発した. 本プラットフォームを用いることで、情報倫理に関するコンテンツを web 上で管理、提供でき、持続的なコンテンツの提供が行える. これらのコンテンツを用いて学習することにより、トラブルの減少やリテラシーの向上が期待できる.

目次

1	序論	1
1.1	本研究の背景	1
1.2	本研究の目的	1
1.3	本報告書の構成	1
2	使用技術	2
2.1	Docker	2
2.2	Python	6
2.3	その他関連技術	9
3	研究内容	11
3.1	概要	11
3.2	開発環境	11
3.3	関連研究	12
3.4	システム概要	13
3.5	コンテンツ提供機能	22
3.6	統計情報提供機能	25
3.7	コンテナ管理機能	27
4	実験・考察	28
4.1	利用評価実験	28
5	結論・今後の課題	30
	謝辞	31

1 序論

1.1 本研究の背景

総務省が令和元年に 37182 人に行った調査によると、インターネットの利用率は、その約 9 割にまで増加している [1]。その理由の 1 つに SNS の普及が挙げられる。SNS に関して平成 27 年にみずほ情報総研株式会社が 1178 人に行った調査研究 [2] によると、SNS 上でトラブルの経験があると回答した割合は 15% 程であった。トラブルの内容は、自分自身の発言を他人が異なる意味で受け取ってしまう、自分の意志とは関係なく個人情報などが第三者に公開されてしまうなどである。このようなトラブルを避けるために情報倫理教育は有効な手段の一つである [3]。

さらに、新しい情報倫理の問題に対応するために、個人の継続的な学習が要求される [4]。しかし、従来の書籍による学習では、新しい問題への速やかな対応が難しいため、学習の継続が難しい場合がある。他方、e ラーニング学習は、学習コンテンツの更新が容易なため、学習の継続が期待できる [5]。

1.2 本研究の目的

本研究では、持続的な学習環境を提供することを目的に、情報倫理教育における e ラーニングのためのプラットフォーム（以下、本プラットフォーム）を開発する。まず、本稿ではコンテンツの作成、統計情報の確認、外部アプリケーションの導入を補佐する機能を開発した。本プラットフォームを用いることで、情報倫理に関するコンテンツを web 上で管理、提供でき、持続的なコンテンツの提供が行える。これらのコンテンツを用いて学習することにより、トラブルの減少やリテラシーの向上が期待できる。

1.3 本報告書の構成

第 2 章では、本研究で使用した技術について述べる。

第 3 章では、本研究の内容について述べる。

第 4 章では、実施した利用評価実験について述べる。

第 5 章では、本研究の結論と今後の課題について述べる。

2 使用技術

本章では，本研究で使用した技術について述べる．

2.1 Docker

2.1.1 概要

Docker[6] は，アプリケーションの開発，導入，実行を行うためのオープンなプラットフォームである．Docker を利用すれば，アプリケーションをインフラストラクチャーから分離できるため，ソフトウェアを素早く提供できる．また，Docker はアプリケーションをパッケージ化して実行するために，ほぼ分離された環境となるコンテナを提供している．

2.1.2 コンテナ

コンテナは Docker がアプリケーションを実行するための，ほぼ分離された環境を提供する．ほぼ分離された環境というものは namespace[7] で実現されている．

コンテナは namespace を用いることによって図 1 のように作成されている．



図 1 コンテナの構成図

ホスト OS 上に namespaceA と namespaceB を作成し、それぞれの namespace の空間で全く独立したファイル構造やプロセス空間を持つことができる。このように分離された環境をコンテナという。コンテナのプロセス空間に関しては、実際にはホスト OS 上で動作しているプロセス ID とコンテナ上で動作しているプロセス ID をマッピングしているテーブルが存在する。これを用いてホスト OS 上では pid2 として認識されているが、コンテナ上では pid1 として認識されていることになる。

このことにより namespace を利用すると、プロセス空間やファイルシステムを一つの OS の中で分離できるので、namespace で区切られた部分であるコンテナは全く異なる OS のようにふるまうことが可能となる。

2.1.3 Docker Engine

2.1.2 節で述べたように Docker はコンテナを用いて動作している。しかし、プロセスをコンテナ上で稼働させるためには、OS の動作に必要なコマンドやライブラリが必要になる。これを実現するためには相当量のコマンドを発行しなければならない。そこで Docker は、Docker Engine を作成した。Docker Engine は主に以下のコンポーネントからなるクライアントサーバ型アプリケーションである。

- サーバ

長時間稼働する種類のプログラムでありデーモン・プロセスと呼ばれる。

- REST API

プログラムとデーモンとの間での通信方法を定義し、何をなすべきかを指示する。

- コマンドライン・インターフェイス (CLI)

クライアント。docker コマンドなど。

デーモンは、Docker オブジェクトを作成、管理する。Docker オブジェクトとは、イメージ、コンテナ、ネットワーク、データ、ボリュームなどを表す。

CLI は Docker REST API を通じて、スクリプトのコマンド実行により、Docker デーモンを制御したり、入出力を行う。Docker アプリケーションの多くが、基本的なところでこの API や CLI を利用している。

また、Docker の内部には以下の 3 つのコンポーネントが存在する。

- Docker イメージ
- Docker レジストリ
- Docker コンテナ

Docker イメージとは、読み込み専用なテンプレートである。Docker イメージは Docker コンテナ作成時に使用される。各イメージはレイヤの積み重ねで構成される。また、新しいイメージの構築や既存のイメージの更新だけでなく、他人が作成した Docker イメージをダウンロードして使用することも可能である。Docker は UnionFS[8] を用いて各イメージのレイヤを単一のイメージに連結する。これにより、Docker イメージに変更を加えた場合、変更したい部分のレイヤを追加もしくは更新するだけでよいため、構築に時間がかからず早く簡単にイメージを提供できる。

Docker レジストリとは、イメージを保持するものである。パブリックもしくはプライベートに保管されているイメージのアップデートやダウンロードを行うことができる。パブリックな Docker レジストリとして Docker Hub[9] が提供されている。

Docker コンテナとは、ディレクトリと似たようなもので、アプリケーションの実行に必要なすべてが含まれている。各コンテナは Docker イメージによって作成される。Docker コンテナは実行・開始・停止・移動・削

除できる。各コンテナは隔離されているため、安全なアプリケーションのプラットフォームとして動作する。

2.1.4 Docker Compose

Compose とは、複数のコンテナを定義し実行する Docker アプリケーションの為のツールである。Compose においては YAML[10] ファイルを使ってアプリケーションサービスの設定を行う。コマンド一つ実行するだけで、設定内容に基づいたアプリケーションの生成、起動を行う。

Compose を使うには基本的に 3 つのステップを踏む。

1. アプリケーション環境を Dockerfile に定義する。
2. アプリケーションを構成するサービスを docker-compose.yml ファイル内に定義する。
3. docker-compose up を実行したら、Compose はアプリケーション全体を起動・実行する。

2.1.5 コマンド一覧

最後に、Docker と Docker Compose で使用する代表的なコマンド一覧とコマンドの説明を表 1 に示す。

表 1 Docker コマンド一覧

<code>docker run -it [イメージ名] /bin/bash</code>	docker コンテナを起動し、ログイン
<code>docker run -p [外部ポート]:[コンテナポート]</code>	コンテナに外部からアクセスする
<code>docker run -v [ホストボリュームパス]:[コンテナ内パス] [コンテナイメージ名]</code>	コンテナのボリュームをホストにマウントする
<code>docker save [イメージ名] >[ファイル名].tar.gz</code>	docker イメージを tar.gz 形式で保存する
<code>docker load <[ファイル名].tar.gz</code>	tar.gz 形式の docker イメージを取り込む
<code>docker exec -it [コンテナ ID/コンテナ名] /bin/bash</code>	起動中のコンテナにログイン
<code>docker cp [ホストファイルパス] [コンテナ ID]:[コンテナコピー先パス]</code>	起動中コンテナにホストからファイルコピー
<code>docker commit [起動中のコンテナ ID] [イメージ名]</code>	起動中のコンテナをイメージとして保存
<code>docker start [コンテナ名]</code>	コンテナの開始
<code>docker stop [コンテナ名]</code>	コンテナの停止
<code>docker restart [コンテナ名]</code>	コンテナの再起動

表 2 Docker Compose コマンド一覧

<code>docker-compose ps</code>	コンテナの一覧
<code>docker-compose images</code>	イメージの一覧
<code>docker-compose up --build</code>	コンテナのビルドと起動
<code>docker-compose down</code>	docker-compose.yml で起動したコンテナの削除
<code>docker-compose start</code>	docker-compose.yml でビルドしたコンテナの起動
<code>docker-compse stop [サービス名]</code>	docker-compose.yml で起動しているコンテナの停止
<code>docker-compse exec [サービス名] [コマンド]</code>	docker-compose.yml で起動したコンテナにログインしてコマンドを発行

2.2 Python

2.2.1 概要

Python[11] は Windows, Linux/Unix, Mac OS X などの主要なオペレーティングシステムおよび Java や.NET などの仮想環境でも動作するインタプリタ形式の、対話的な、オブジェクト指向プログラミング言語である。この言語には、モジュール、例外、動的な型付け、超高水準の動的データ型、およびクラスが取り入れられている。Python はオブジェクト指向プログラミングを超えて、手続き型プログラミングや関数型プログラミングなど複数のプログラミングパラダイムをサポートしている。また、多くのシステムコールやライブラリだけでなく、様々なウィンドウシステムへのインターフェースがあり、C[12] や C++[13] で拡張することもできる。

2.2.2 Django

Django[14] は、Python で実装された無料オープンソースの Web アプリケーションフレームワークの一つである。Django が作られた時の目的として、複雑なデータベース主体のウェブサイトを簡単に構築するというものがある。これを実現するため、Django ではコンポーネントの再利用性、素早い開発の原則に力を入れている。このような流れから、Django には以下のような特徴がある。

- **高速な動作**

Django には標準で分散型のキャッシュシステムである memcached[15] が備え付けられており、キャッシュ機能が強力である。

- **フルスタック・フレームワーク**

Django には、Web アプリケーションの実装に必要な、ユーザ認証、管理画面、RSS フィードなどの機能があらかじめ含まれている。

- **セキュリティ的に安全な設計**

Django では、デフォルトでパスワードなどはハッシュ化しデータベースに格納する。また、SQL インジェクション、クロスサイトスクリプティング、クロスサイトリクエストフォージェリなどの多くの脆弱性についても保護を有効にしている。

- **自由に選択できるプラットフォーム**

Django は、そのすべてが Python から作成されている。これにより、Python が Linux, Windows, MacOS X などで行えるように Django も多くのプラットフォームで動作可能である。

また、Django の機能構成として MVT(Model・View・Template) というものがある。Django は MVT のもと、ルーティングというリクエストを振り分ける機能を用いてアプリケーションを動作させている。Django がどのように動作するかを図 2 に示す。

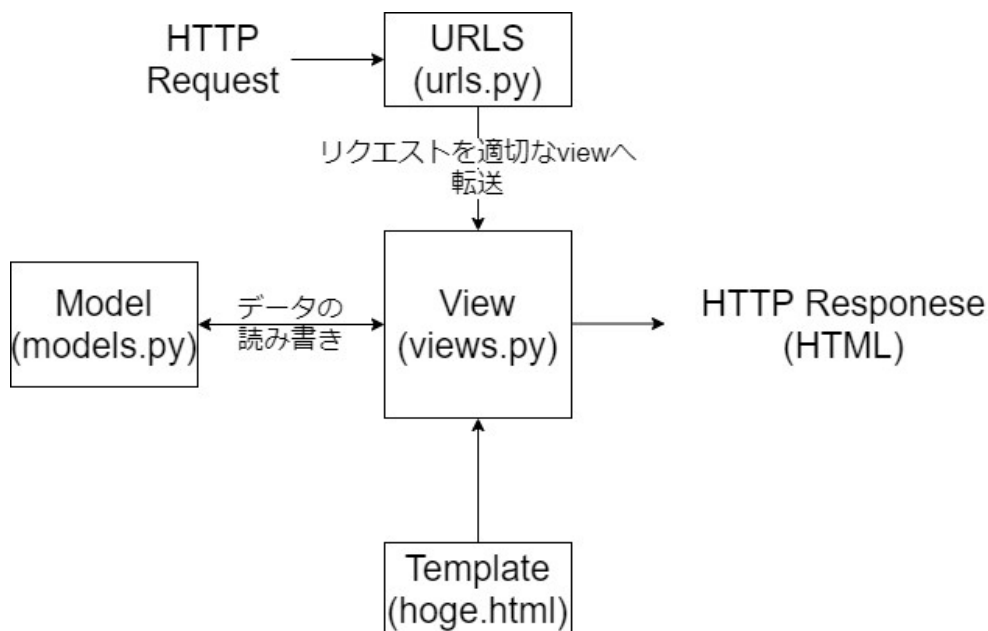


図 2 Django の動作の概要図

URLS は単一 URL から単一の関数を介してすべてのリクエストを処理することは可能だが、各リソースを処理する別々の View 関数を作成する方がメンテナンスが容易なため、URL をマッピングしている。これにより、URL に含まれる特定のパターンの文字列もしくは数字を照合し、これらをデータとして View 関数に渡して処理を行っている。

View は、HTTP リクエストを受け取り、HTTP レスポンスを返す関数である。View はモデルを介して要求を満たすために必要なデータにアクセスし、レスポンスのフォーマットを Template に委任する。

Model は、アプリケーションのデータ構造を定義し、データベース内のレコードを追加、変更、削除、および照会するための機能を提供する Python オブジェクトである。

Template は、ファイル構造やレイアウトを定義するテキストファイルで、プレースホルダを使用して実際のコンテンツを表示する。View は、モデルから取得したデータと HTML テンプレートを使用して動的に HTML ページを作成できる。また、Template を利用して、HTML だけでなくあらゆる種類のファイルの構造を定義できる。

続いて、Django は複雑なデータベース主体のウェブサイトを簡単に構築するため、データベースを操作する 3 つの機能がある。

- Django シェル

Django アプリケーションの環境を有効にしたままコマンドで操作が可能。

- マイグレーション

データベースの操作を一括に実行・取り消しを行うことが可能。

- 管理サイト

Web ブラウザで、データベースを操作することが可能。

これらの機能を用いることにより、複雑なデータベース主体のウェブサイトを簡単に構築できる。

最後に Django で主に使用するコマンド一覧とその説明を表 3 に示す。

表 3 Docker コマンド一覧

manage.py createsuperuser	管理者ユーザを作成する。
manage.py changepassword	ユーザのパスワードを変更する。
manage.py check	プロジェクトの構成チェック。
manage.py dbshell	データベース管理用のシェルを起動する。
manage.py dumpdata	データベースの内容を JSON 等の形式でエクスポートする。
manage.py loaddata	データベースに dump ファイルをインポートする。
manage.py makemigrations	マイグレーションファイルの作成。
manage.py migrate	データベースにマイグレーションを摘録する。
manage.py shell	管理者用のコマンドシェルを起動したいわモードに入る。
manage.py startapp	新規アプリケーションの作成。
manage.py startproject	プロジェクトディレクトリの作成。
manage.py test	テストフレームワークの実行。
manage.py testserver	検証用サーバを起動する。
manage.py collectstatic	static ファイルの収集を行う。
manage.py runserver	アプリケーションの起動を行う。

2.2.3 Django REST framework

Django REST framework[16] とは Django を使って RESTful な API を開発するために利用されるライブラリである。Django REST framework を動作させるには、以下の要件を満たさなければならない。

1. Python (3.5, 3.6, 3.7, 3.8, 3.9)
2. Django (2.2, 3.0, 3.1)

2.3 その他関連技術

2.3.1 Go

Go[17] は静的型付けの効率と安全性、動的型付けのプログラミングの容易さを兼ね備えたプログラミング言語である。また、Go のコンパイラ、ツール、およびソースコードはすべてフリーなオープンソースである。

2.3.2 PostgreSQL

PostgreSQL[18] とは、Linux, Windows, MacOS X に対応した、オープンソースのリレーショナルデータベース管理システム (RDBMS) である。特徴として以下のようなものがある。

拡張性

PostgreSQL では、アーキテクチャが拡張可能である。このことにより、データベース内の手順のカスタマイズや自動化に使用できるユーザ定義関数とサードパーティライブラリへのサポートが可能となる。

同時実行性

PostgreSQL では、マルチバージョン同時実行制御があり、ユーザは同じデータベースの書き込みと読み取りが可能となっている。

標準 SQL への準拠

PostgreSQL では、ISO/IEC の標準 SQL にならった実装となっている。

2.3.3 Chart.js

Chart.js[19] とは、HTML5 の Canvas を使用したグラフ描画ライブラリである。Chart.js はオープンソースなプロジェクトで、8 つのチャートタイプと優れたレスポンスを有している。8 つのチャートタイプは以下の通りである。

- 線グラフ
- 棒グラフ
- レーダーチャート
- 円グラフ
- 鶏頭図
- バブルチャート
- 散布図
- エリアチャート

また、2 種類以上のチャートを複合した複合グラフも作成できる。

2.3.4 Json

Json[20](Javascript Object Notation) は、軽量なデータ交換フォーマットである。人間にとって読み書きが容易で、マシンにとってもパースや生成を行える。

2.3.5 NGINX

NGINX[21] は、フリーでオープンソースな Web サーバである。HTTP, HTTPS, SMTP, POP3, IMAP のリバースプロキシ機能や、ロードバランサ、HTTP キャッシュなどの機能を持つ。

2.3.6 Gunicorn

Gunicorn[22] は、UNIX のための Python WSGI HTTP Server である。WSGI[23] とは、Python において Web サーバと Web アプリケーションを接続するための、標準化されたインターフェース定義のことである。

3 研究内容

本章では，概要，開発環境，関連研究，システム概要を述べた後に，開発した本プラットフォームの詳細について述べる．

3.1 概要

本研究の目的は，情報倫理教育に関する学習を支援することである．そのため，学習者が情報倫理に関して学ぶ際，手軽に学習する環境を構築する必要がある．これを解決するために本研究では学習をインターネット上で手軽に学習するために web アプリケーションを用いた e ラーニングプラットフォームを開発した．本プラットフォームは，情報倫理に関して学習する学習者と情報倫理に関するコンテンツを提供する教材提供者を対象としたシステムである．教材提供者に対して本プラットフォームではコンテンツ提供機能，統計情報提供機能，コンテナ管理機能を用意している．教材提供者はこれらの機能を用いることにより情報倫理に関するコンテンツを web アプリケーション上に投稿でき，学習者はそれらのコンテンツを用いて情報倫理に関する学習を行える．

3.2 開発環境

本プラットフォームを作成するにあたって使用した PC のスペックと開発環境を表 4 に示す．

表 4 PC のスペックと開発環境

CPU	Intel Core i7 @ 3.70GHz
Memory	24.0GB
OS	Windows 10 Education 64-bit
開発環境	Docker version 20.10.0, build 7287ab3 docker-compose version 1.27.4, build 40524192

3.2.1 クライアントシステムとサーバシステムの開発環境

クライアントシステムとサーバシステムの開発には Docker のコンテナを用いた．表 5，表 6 にそれぞれの開発環境を示す．

表 5 クライアントシステムの開発環境

OS	Linux
使用言語	Python 3.7.9 Django 3.0.2

表 6 サーバシステムの開発環境

OS	GNU/Linux
使用言語	go version go1.15.6 linux/amd64 djangorestframework 3.12.1

3.3 関連研究

関連研究として、上田氏らの「倫倫姫プロジェクト-学人連携 Moodle による多言語情報倫理 e ラーニング-」[24] がある。

倫倫姫プロジェクトでは大学の情報倫理教育における以下 3 つの問題を解決している。

- (i) 標準化と可視化がなされていない
- (ii) 留学生への教育が困難
- (iii) 持続可能性が低い

(i) を解決するために、倫倫姫ではサンプル規程集「A3301 教育テキスト作成ガイドライン (一般利用者向け)」に準拠することで、内容を標準化した。また、受講履歴を閲覧でき受講者の学習状況の可視化も実現した。

(ii) を解決するために、倫倫姫では英語、中国語、韓国版を作成しており、各言語圏の文化の違いも考慮しコンテンツを作成することで解決した。

(iii) を解決するために、倫倫姫では SCORM が規定する e ラーニングコンテンツパッケージを利用した。これは、パッケージ構造とリソースを記述するマニフェストファイル (imsmanifest.xml) とそれから参照される物理ファイル (HTML,swf など) をファイル単位で修正可能なため継続的な改訂を可能としている。

3.4 システム概要

3.4.1 システム構成

本システムの構成を図3に示す。本システムは Django で構成されるアプリケーション部 (以下, クライアント), および golang と djangorestframework で構成されるサーバ部 (以下, サーバ) から構築される。サーバは API を用いて統計情報提供機能とコンテナ管理機能を動作させる。

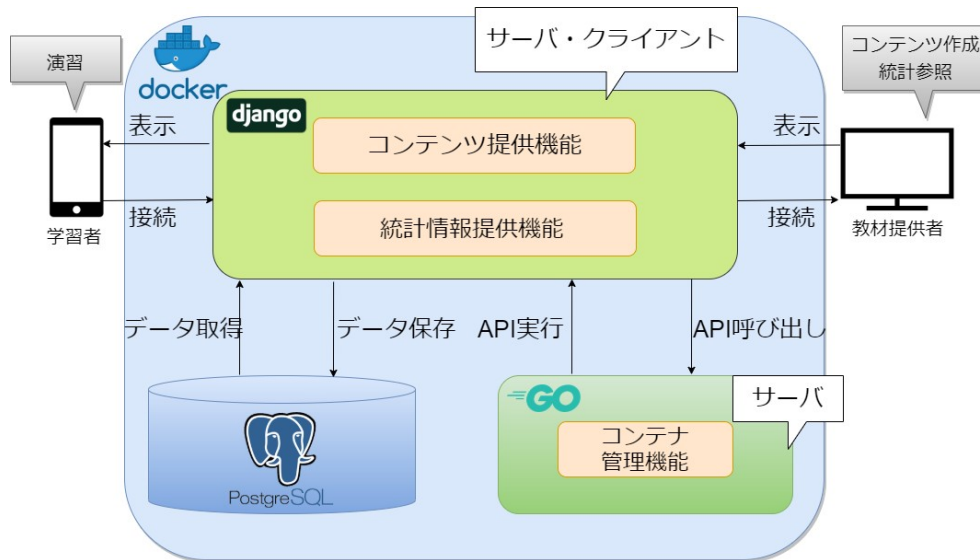


図3 システム構成

コンテンツ提供機能の GUI を図4に、統計情報提供機能の GUI を図5に、コンテナ管理機能の GUI を図6にそれぞれ示す。

教材提供者は図 4 のコンテンツ提供機能を用いて情報倫理に関するコンテンツを提供する。

admin | ユーザー情報更新 パスワードの変更 ログアウト コンテンツ関連 ▾ 統計情報

タイトル:
ネット詐欺などに巻き込まれないようにするために

タグ:
ネット依存
コンテンツ依存
つながり依存
ネット詐欺

説明文:
ネット詐欺
高校1年生のゆきさんとめく みさんは、インターネットでアイドルグループのコンサート チケットを購入しましたが、行ってみるとそれが偽物だったことが判明しました。母親のクレジットカードには、50 万円 もの決済書が送られてきました。

<iframe width="560" height="315" src="https://www.youtube.com/embed/0P26mr9zkVs" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>

ネット詐欺
高校1年生のゆきさんとめく みさんは、インターネットでアイドルグループのコンサート チケットを購入しましたが、行ってみるとそれが偽物だったことが判明しました。母親のクレジットカードには、50 万円 もの決済書が送られてきました。

🕒 後で見る

🔗 共有

図 4 コンテンツ提供機能の GUI

図 5 の統計情報提供機能では，教材提供者が学習者の回答情報等をグラフとして確認できる．



図 5 統計情報提供機能の GUI

図 6 のコンテナ管理機能では教材提供者が Docker を用いて作成した他の教育アプリケーションを本プラットフォームでも利用することが可能となる。

The screenshot displays a web interface for container management. At the top, a navigation bar includes links for 'admin!', 'ユーザー情報更新', 'パスワードの変更', 'ログアウト', 'コンテンツ関連', and '統計情報'. The main form contains three sections: a 'URL:' field with a text input box; a '使用可能ポート一覧' (List of available ports) section showing a scrollable list of ports 49152, 49153, 49154, and 49155; and a '実行手順:' (Execution steps) section with a large text area. A blue '送信' (Send) button is located at the bottom right of the form.

図 6 コンテナ管理機能の GUI

学習者は図 4 のコンテンツ提供機能を用いて作成されたコンテンツを図 7 のようにして閲覧，学習することが可能である。



図 7 コンテンツ閲覧時の GUI

3.4.2 システムの内部構成

クライアントの内部構成を図 8 に示す。はじめに、教材提供者および学習者が各々ネットワークに接続できる環境を用意し、Web ブラウザを立ち上げ、特定の IP アドレスを入力しログインする。ただし、教材提供者が本プラットフォームの機能を利用するにはログインは必須であるが、学習者は必須ではない。

Web ブラウザで本プラットフォームに接続しログインした後、教材提供者は GUI のコンテンツ提供機能、統計情報提供機能、コンテナ管理機能を使用できる。コンテンツ提供機能は教材提供者が入力した内容をサーバを介しデータベースに保存する。統計情報提供機能はデータベースからサーバを介しデータを取得し Web ブラウザ上に情報を表示する。コンテナ管理機能は教材提供者が入力した内容を golang で作成した API で実行し、Docker のコマンドを用いてコンテナが建ち上がっていることを確認し、建ち上がっていた場合それにアクセスする URL を画面上に発行する。

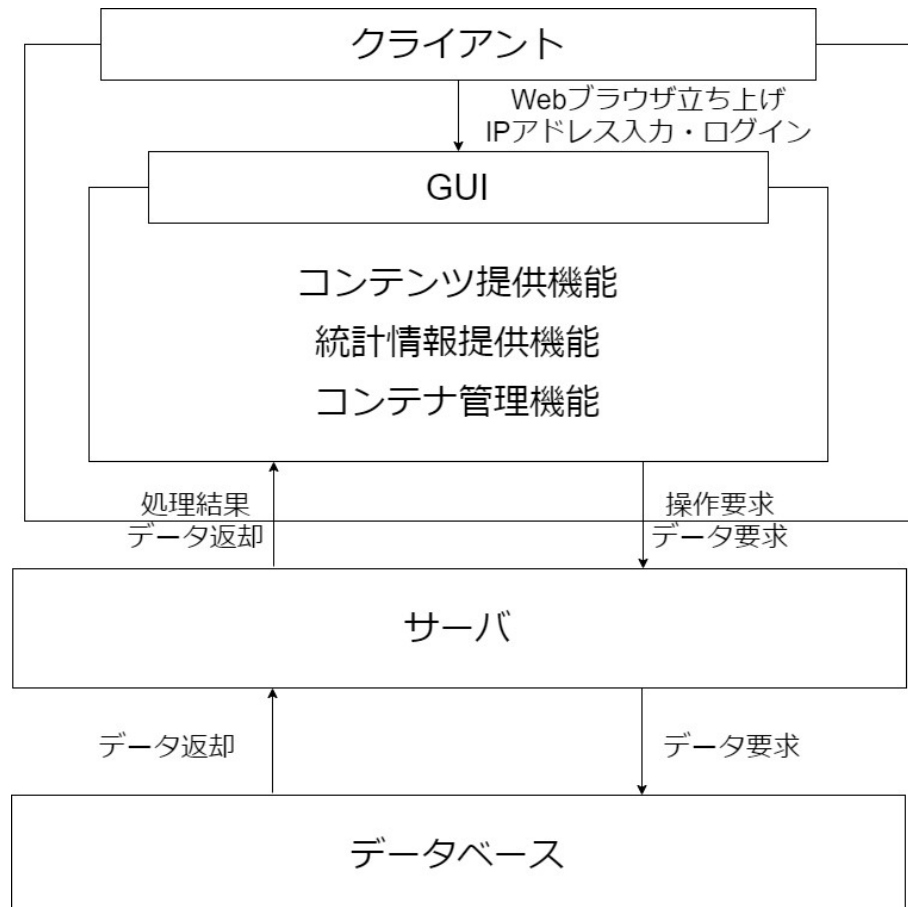


図 8 クライアントの内部構成

続いて、サーバの内部構成を図9に示す。サーバは Django を用いて作成された Django 処理部と golang を用いて作成された golang 処理部、データを保存するためのデータベース処理部がある。サーバはクライアントからの通信が行われた場合に動作する。それぞれの処理部の内容について以下に示す。

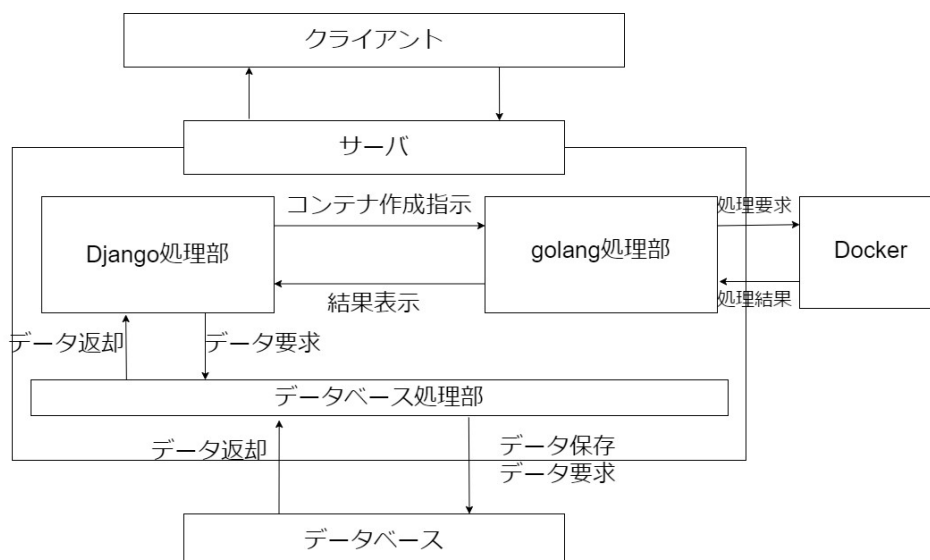


図9 サーバの内部構成

• Django 処理部

Django 処理部では、クライアントからフォームに従って入力されたデータをデータベースに登録、抽出する処理を行っている。具体的には、ユーザの新規登録、ログイン処理、パスワードや登録情報の変更、コンテンツ提供機能とコンテナ管理機能で入力された情報の登録と修正、タグの検索処理、統計情報提供機能のためのデータの検索が挙げられる。

• golang 処理部

golang 処理部では、コンテナ管理機能により入力された情報を API を用いて Docker に実行、処理させコンテナを建ち上げている。

• データベース処理部

データベース処理部では、Django 処理部においてデータベースのやり取りが必要な場合に動作する。Django 処理部から sql コマンドが発行され、それを実行処理している。

3.4.3 システムのフローチャート

教材提供者のフローチャートを図 10 に、学習者のフローチャートを図 11 に示す. 本システムはまず、教材提供者はログインが必須、学習者は任意となっている. そのため今回は簡単のため教材提供者と学習者がともにログインした場合のフローチャートとする.

はじめに教材提供者について説明を行う. 教材提供者は事前にユーザ登録を行ったことを管理者に通知し、管理者からアカウントを一般のものから教材提供者のアカウントに設定する必要がある. 教材提供者アカウントに設定した後、ログイン処理を行う. ログインが完了したら教材提供者はコンテンツ提供機能を用いてコンテンツの必要情報を入力し、サーバに対して登録を行う. 統計情報提供機能を用いる場合は、コンテンツに対する統計情報をサーバに対しリクエストすると、その結果が画面上に表示される. コンテナ管理機能を用いる場合は、動作させたいアプリケーションの情報をクライアント上で入力することにより、サーバに対しその情報がリクエストされ、実行される. 実行が正常に完了した場合、接続するための URL が画面上に表示される.

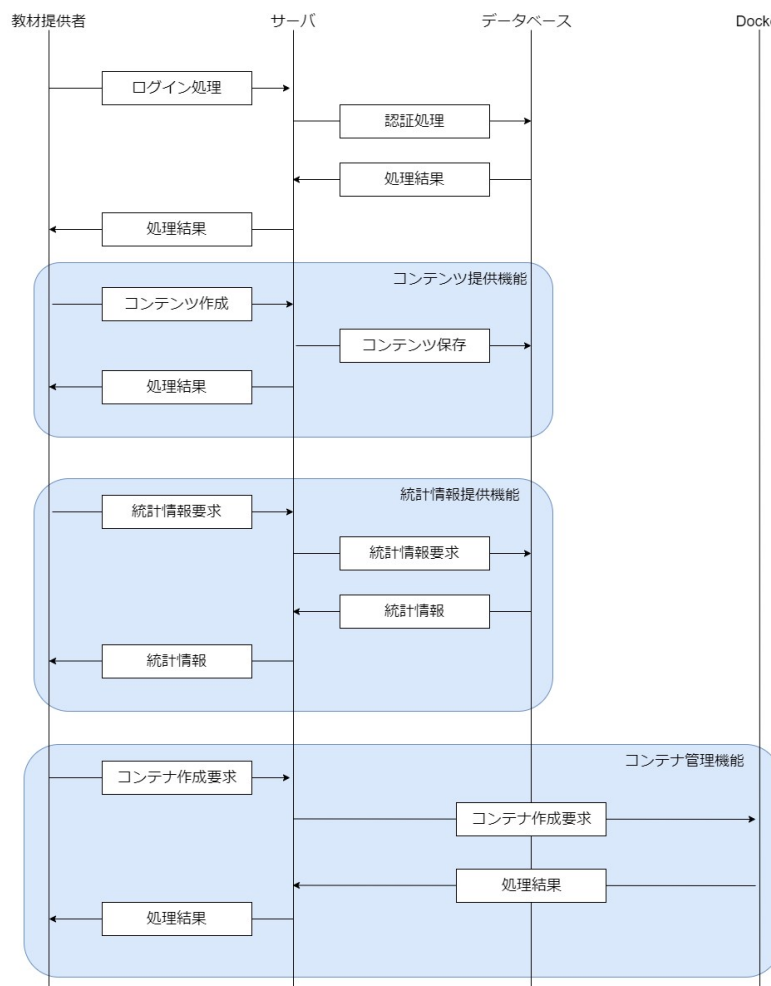


図 10 教材提供者のフローチャート

続いて学習者について説明を行う。学習者は Web ブラウザで本プラットフォームに接続した後、本プラットフォームでアカウント作成を行うことができる。アカウント作成には、本プラットフォーム上で表示されるユーザ名、性別、年齢、パスワードと確認用パスワードが必要である。アカウント作成後、トップページにて教材提供者が作成したコンテンツを選択、閲覧できる。

コンテンツ提供機能、統計情報提供機能、コンテナ管理機能についての詳細は 3.5 節，3.6 節，3.7 節にて述べる。

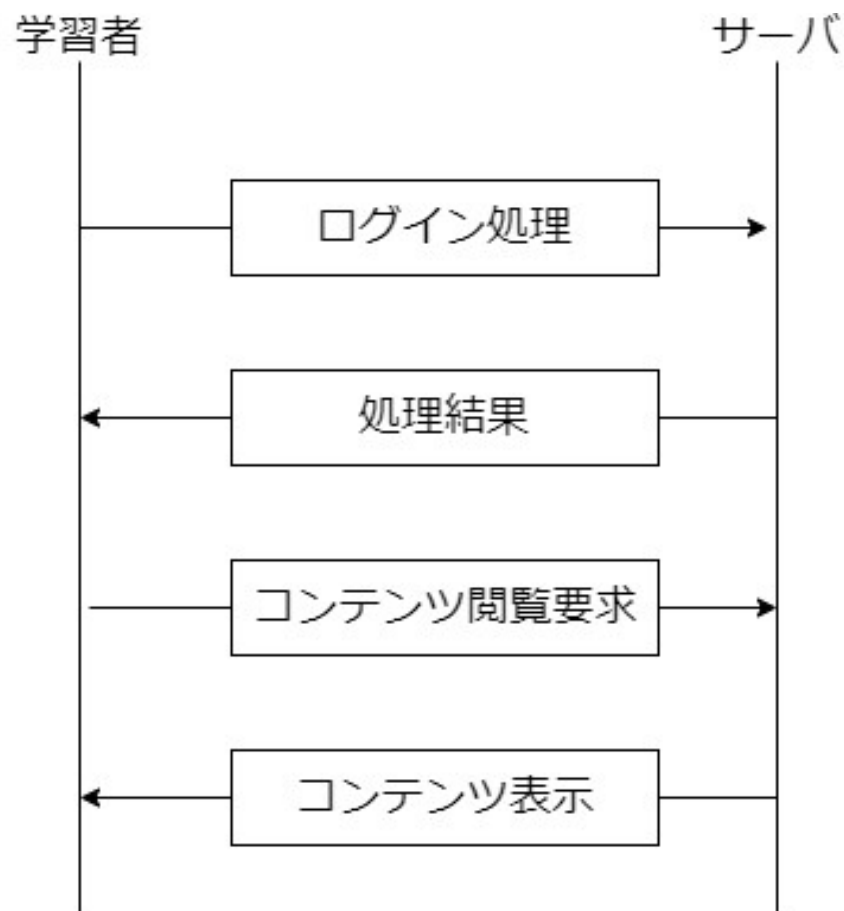


図 11 学習者のフローチャート

3.5 コンテンツ提供機能

コンテンツ提供機能は、教材提供者が情報倫理に関するコンテンツを提供するための機能である。本機能では Web 上で教材提供者のみがコンテンツの投稿と管理ができる。コンテンツを投稿する際の GUI を図 12 に示す。図 12 に示したとおり、コンテンツを投稿する際に必要な情報は、コンテンツのタイトルとコンテンツのタグ、本文である。本文はマークダウン形式で記入可能であり、画像や動画の挿入が容易にできる。またプレビュー機能もあるため投稿する前にどのような見で投稿されるのかを確認できる。

The screenshot displays the user interface for creating new content. It includes a 'タイトル:' (Title) field with the placeholder text 'このサイトについて' (About this site). Below it is a 'タグ:' (Tags) section with a list of tags: 'ネットゲーム', 'ゲーム依存', 'ネット', 'タグ追加ボタン' (Tag add button), and 'コ'. A plus icon is visible to the left of the tag list. The main area is a '説明文:' (Description) field containing a sample text in Japanese about internet usage and SNS. The text is formatted with a heading '# 序論' (Introduction) and several paragraphs. The text is rendered in a light gray font, indicating it is a preview or template text.

図 12 コンテンツ提供機能の GUI

コンテンツのタグは図 12 に示すプラスボタンを押下することにより，別画面で新たなタグを登録できる．タグを登録する際の GUI を図 13 に示す．

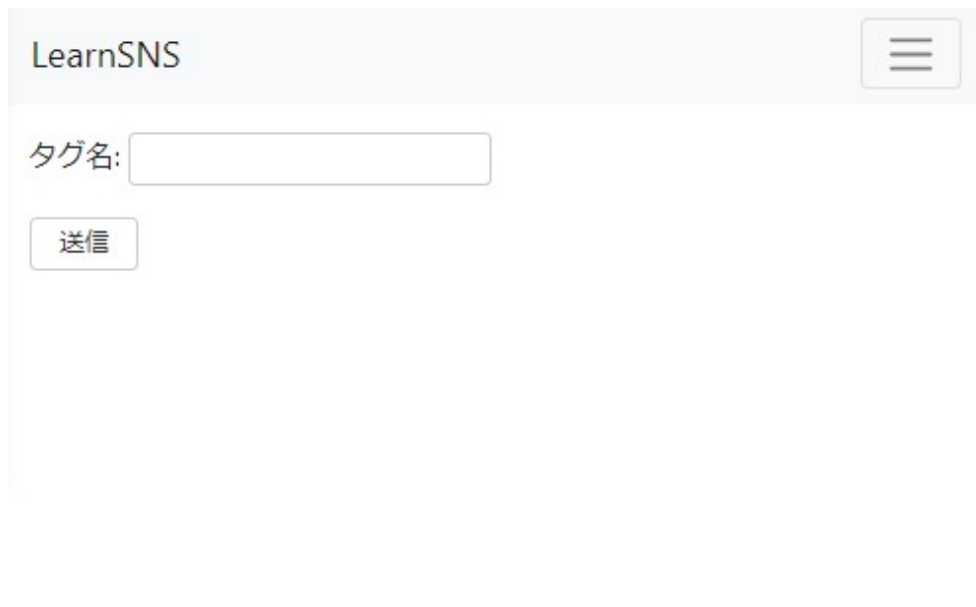


図 13 タグ登録の GUI

また，教材提供者はコンテンツを管理するためのグループに参加する．そのグループはすべてのコンテンツに対して公開または非公開を選択することでコンテンツを管理する．グループに参加している教材提供者全員が認可したコンテンツのみが学習者に公開される．これにより，コンテンツの正当性を担保できる．教材提供者をグループに参加させるには，図 14 のようにして教材提供者のアカウントを選択し教材提供者のグループに登録する．



図 14 グループ登録の GUI

続いて本機能ではコンテンツ作成後にコンテンツの 4 択問題を作成することができる。問題の作成には、問題のタイトル、問題文、選択肢 1~4 および正解の選択肢をフォームに従って入力する。問題作成の際の GUI を図 15 に示す。

最後に、Django の機能を用いることにより本機能で投稿したコンテンツを json ファイルに変換することができる。json ファイルに変換するには、以下のような手順を経てコマンドを発行すればよい。これにより、教材提供者はコンテンツを互いに json ファイルを介して共有することが可能となる。

1. 「docker exec -it コンテナ名 /bin/ash」などとして、本プラットフォームを動作させているコンテナにログインする。
2. cd コマンドを用いて Django によって自動的に生成された manage.py ファイルと同階層に移動する。
3. 「python manage.py dumpdata json ファイルに変換したいデータの DB のテーブル名 >保存する json ファイル名.json」を実行後、json ファイルが生成。

LearnSNS ようこそ, admin! ユーザー情報更新 パスワードの変更 ログアウト コンテンツ関連 統計情報

問題のタイトル:

問題文:

選択肢1:

選択肢2:

選択肢3:

選択肢4:

答え:

保存

図 15 問題作成の GUI

3.6 統計情報提供機能

統計情報提供機能は、教材提供者が作成した問題を学習者が解いた際の回答情報を基に、グラフで統計情報を提供する機能である。これにより、教材提供者は作成したコンテンツの質を向上させることができる。提示する統計情報の内容としては、問題の各選択肢における割合や回答者の年齢層、性別である。統計情報提供機能の GUI は図 16 の通りである。



図 16 統計情報提供機能の GUI

本機能は Chart.js という javascript で作成されたグラフ描画ライブラリを作成している．Chart.js では線グラフ，棒グラフ，レーダーチャート，鶏頭図，ドーナツチャート，円グラフ，バブルチャートを作成できる．本機能では，棒グラフと円グラフを使用している．

また，本機能で取得可能な情報を表 7 に示す．これらの取得できる情報を使って教材提供者は Chart.js で新たなグラフを挿入することができる．

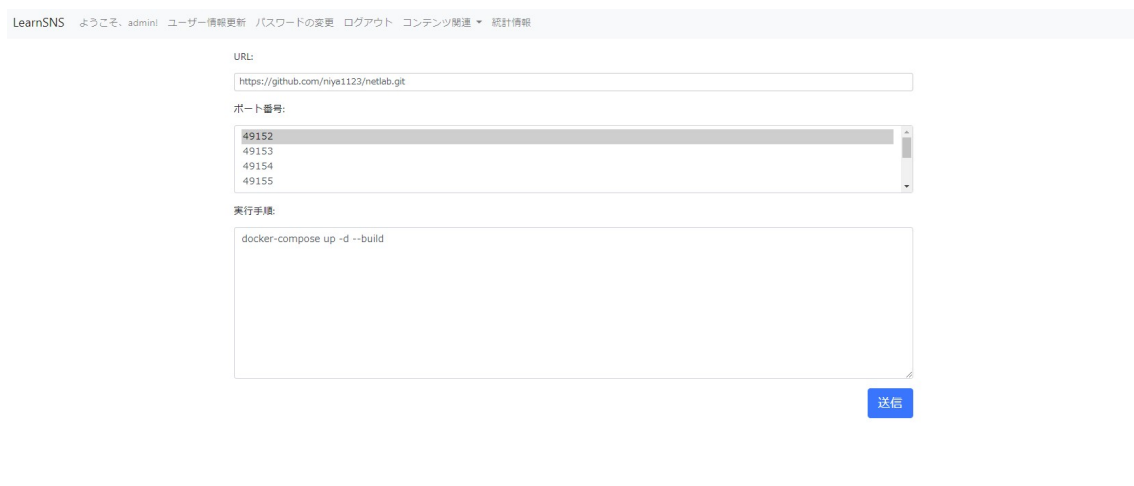
表 7 取得可能情報一覧

取得可能情報	詳細
学習者情報	年齢
	性別
	回答した問題・選択肢
コンテンツ情報	問題に対応したコンテンツ

3.7 コンテナ管理機能

コンテナ管理機能とは、教材提供者が Docker を用いて作成したアプリケーションを本プラットフォームでも利用可能とするための機能である。本機能を利用するには、本プラットフォームで動作させたいコンテナ情報が記載された Dockerfile と docker-compose.yml を事前に Github などのリポジトリに用意する。外部通信が必要なアプリケーションの場合、本機能が提示するポート番号のみが利用可能となっている。以上の情報を、本機能が提示するフォームに入力することにより、本プラットフォームで Docker のコンテナを作成し、アクセスするための URL が画面上に表示される。教材提供者は発行された URL にアクセスすることで動作を確認できる。また、発行された URL をコンテンツ提供機能の本文に貼り付けることにより、学習者はそのコンテンツを利用できる。

本機能の入力画面と入力が完了し、コンテナが作成された後の画面を図 17、図 18 に示す。



LearnSNS ようこそ、admin! ユーザー情報更新 パスワードの変更 ログアウト コンテンツ関連 ▾ 統計情報

URL:

ポート番号:

49152
49153
49154
49155

実行手続:

docker-compose up -d --build

送信

図 17 コンテナ管理機能の入力画面



LearnSNS ようこそ、admin! ユーザー情報更新 パスワードの変更 ログアウト コンテンツ関連 ▾ 統計情報

コンテナの作成が完了しました。

<http://localhost:49152>

図 18 コンテナ管理機能のコンテナ作成後の画面

4 実験・考察

本章では、実施した利用評価実験の内容と結果および考察について述べる。

4.1 利用評価実験

本プラットフォームが、学習者に対し情報倫理に関するコンテンツを提供でき、そのコンテンツを見て学ぶことにより情報倫理に関するトラブルの減少とリテラシーの向上が期待できるかどうかを確認するために、利用評価実験を行った。実験では 20 代男性 10 人に対して、本プラットフォームのコンテンツを事前に学習したコンテンツ利用者群と、事前に学習していないコンテンツ非利用者群の 5 人ずつに分かれて実験を行った。そして各々の群に対し、GoogleForm を用いて全 5 問の情報倫理に関する問題を解いてもらった。ただし、コンテンツ非利用者群に対して GoogleForm で問題を解くときに、わからない単語等があれば調べてもよいとした。

このような条件で実施したコンテンツ利用者群とコンテンツ非利用者群の実験結果を図 19、図 20 に示す。図 19 はコンテンツ利用者群とコンテンツ非利用者群の獲得点数を表している。図 20 はコンテンツ利用者群とコンテンツ非利用者群の獲得点数の平均と標準偏差を表している。

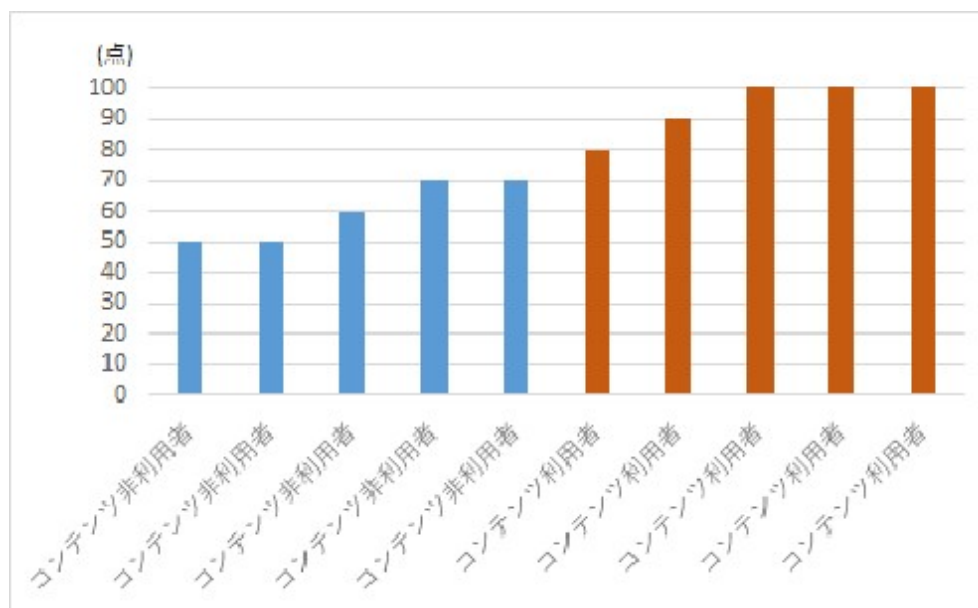


図 19 コンテンツ利用者群とコンテンツ非利用者群の獲得点数

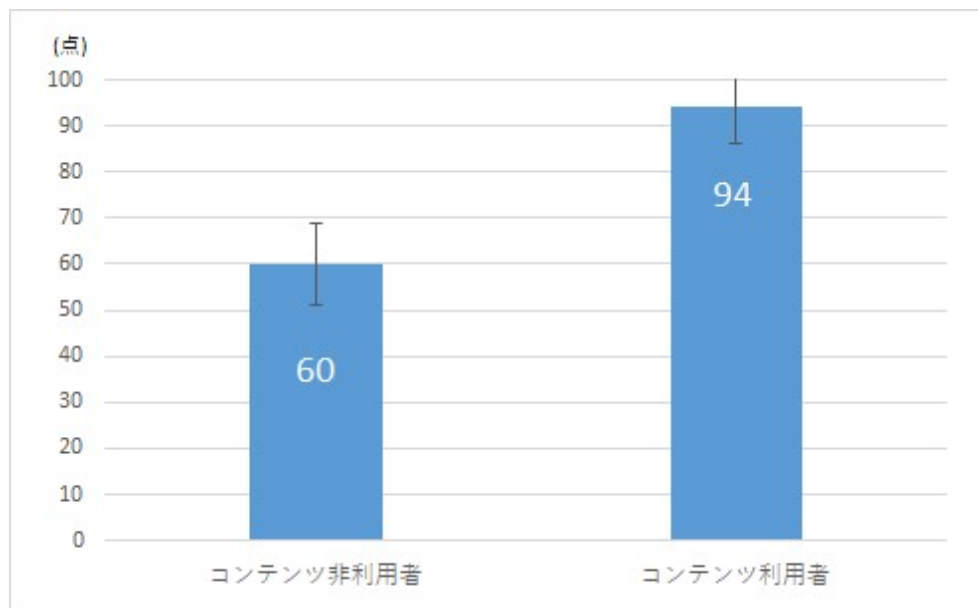


図 20 コンテンツ利用者群とコンテンツ非利用者群の獲得点数の平均と標準偏差

実験の結果、平均点においてコンテンツ利用者群が 94 点、コンテンツ非利用者群が 60 点と 34 点の差がついていることが分かった。また、標準偏差においてはコンテンツ利用者群が 8.00、コンテンツ非利用者群が 8.94 であり、0.94 の差がついている。以上から、本プラットフォームを用いて情報倫理について学習すると、点数のばらつきはコンテンツ利用者群の方がなく、点数も平均点の差からコンテンツ非利用者群と比べ高くなっていることがわかる。したがって、本プラットフォームは学習者に対して、情報倫理に関するトラブルの減少とリテラシー向上を期待できる。

5 結論・今後の課題

本研究では、持続的な学習環境を提供することを目的に、情報倫理教育における e ラーニングのためのプラットフォームを開発する。まず、本稿ではコンテンツの作成、統計情報の確認、外部アプリケーションの導入を補佐する機能を開発した。本プラットフォームを用いることで、情報倫理に関するコンテンツを web 上で管理、提供でき、持続的にコンテンツの提供ができる。これらのコンテンツを用いて学習することにより、トラブルの減少やリテラシーの理解を期待できる。今後、教材提供者に対し、本プラットフォームを用いて持続的なコンテンツの提供が可能か否かを確認する予定である。

謝辞

本研究を遂行するに当たり，熱心な御指導および御鞭撻をいただきました井口信和教授に深く感謝いたします。また，実験にご協力いただきました被験者の方々に深く感謝いたします。さらに，ネットワーク研究室の皆様をはじめ，研究活動を支えてくださりました皆様に深く感謝いたします。

参考文献

- [1] 総務省. 令和元年通信利用動向調査の結果. 入手先<https://www.soumu.go.jp/johotsusintokei/statistics/data/200529_1.pdf>. 参照 (2020-01-15).
- [2] みずほ情報総研株式会社. 社会課題解決のための新たな ict サービス・技術への人々の意識に関する調査研究-報告書-. 入手先<https://www.soumu.go.jp/johotsusintokei/linkdata/h27_06_houkoku.pdf>. 参照 (2020-01-15).
- [3] 文部科学省. 第 5 章 情報モラル教育: 文部科学省. 入手先<https://www.mext.go.jp/b_menu/shingi/chousa/shotou/056/shiryo/attach/1249674.htm>. 参照 (2020-01-15).
- [4] 辰巳丈夫. 情報フルーエンシーと情報倫理教育. 入手先<<https://gakkai.univcoop.or.jp/pcc/paper/2010/pdf/166.pdf>>. 参照 (2020-01-15).
- [5] チエル株式会社. 第 5 章 インターネットと教育 5.3 生涯学習と e ラーニング. 入手先<<https://www.chieru.net/HTMLMaterial/superj2014/0503.html>>. 参照 (2020-01-15).
- [6] Empowering app development for developers — docker. 入手先<<https://www.docker.com/>>. 参照 (2020-01-15).
- [7] Namespaces in operation, part 1: namespaces overview [lwn.net]. 入手先<https://lwn.net/Articles/531114/#series_index>. 参照 (2020-01-15).
- [8] Unionfs: A stackable unification file system. 入手先<<https://unionfs.filesystems.org/>>. 参照 (2020-01-15).
- [9] Docker hub. 入手先<<https://hub.docker.com/>>. 参照 (2020-01-15).
- [10] The official yaml web site. 入手先<<https://yaml.org/>>. 参照 (2020-01-15).
- [11] Welcome to python.org. 入手先<<https://www.python.org/>>. 参照 (2020-01-15).
- [12] Clang c language family frontend for llvm. 入手先<<https://clang.llvm.org/>>. 参照 (2020-01-15).
- [13] cplusplus.com - the c++ resources network. 入手先<<https://www.cplusplus.com/>>. 参照 (2020-01-15).
- [14] The web framework for perfectionists with deadlines — django. 入手先<<https://www.djangoproject.com/>>. 参照 (2020-01-15).
- [15] memcached - a distributed memory object caching system. 入手先<<https://memcached.org/about>>. 参照 (2020-01-15).
- [16] Home - django rest framework. 入手先<<https://www.django-rest-framework.org/>>. 参照 (2020-01-15).
- [17] The go programming language. 入手先<<https://golang.org/>>. 参照 (2020-01-15).
- [18] PostgreSQL: The world's most advanced open source database. 入手先<<https://www.postgresql.org/>>. 参照 (2020-01-15).
- [19] Chart.js — open source html5 charts for your website. 入手先<<https://www.chartjs.org/>>. 参照 (2020-01-15).
- [20] Json. 入手先<<https://www.json.org/json-en.html>>. 参照 (2020-01-15).
- [21] Nginx — high performance load balancer, web server, & reverse proxy. 入手先<<https://www.nginx.com/>>. 参照 (2020-01-15).

- [22] Gunicorn - python wsgi http server for unix. 入手先<<https://gunicorn.org/>>. 参照 (2020-01-15).
- [23] Pep 333 – python web server gateway interface v1.0 — python.org. 入手先<<https://www.python.org/dev/peps/pep-0333/>>. 参照 (2020-01-15).
- [24] 上田浩, 中村素典, 古村隆明, 神智也. 倫倫姫プロジェクト-学認連携 moodle による多言語情報倫理 e ラーニング-. 情報処理学会デジタルプラクティス, Vol. 6, No. 2, pp. 97–104, 2015. 参照 (2020-01-15).