学生と教員が共同運用する To-do 管理システムの開発

村上裕史[1]　宮田優作[1]　小島俊輔[1]

概要：高専・大学等において課題やレポート（以下 To-do）の提示は，SNS やグループウェア，メールといった情報ツール以外に，口頭，掲示版，研究室ホワイトボードなどアナログ手段が今でも存在する．そのため学生は To-do を個別に漏れなく管理することが困難な状況にある．そこで本研究では「チーム」という概念を導入することで，学生と教員が自由に参加し共同で管理する To-do 管理システムを設計・開発する．システムは，チームの誰かが To-do を入力することでチームに所属する全員が入力したのと同じ状況を作り出す．システムは入力漏れの予防だけでなく，各自の To-do の過密度をヒートマップで可視化する機能を実現する．

キーワード：インタフェースデザイン，Web サービス，協同学習支援

Development of a To-Do Management System to be Operated Jointly by Students and Faculty

HIROFUMI MURAKAMI[†1]　YUSAKU MIYATA[†1]
SHUNSUKE OSHIMA[†1]

Abstract: In technical colleges and universities, there are other ways to share assignments and reports (hereinafter referred to as "to-do") information tools such as SNS, groupware, and e-mail. In addition, there are other than information tools still analog methods such as oral communication, bulletin boards, and whiteboards in laboratories. This makes it difficult for students to individually manage their to-dos without omissions. Therefore, in this study, we introduced the concept of "teams" and designed and developed a system in which students and faculty members can freely participate and collaborate in managing their to-do. With this system, when someone in a team enters a to-do, it creates the same situation as if all team members had entered the to-do. In addition to preventing input omissions, the system can visualize the over density of each person's to-do on a heat map.

Keywords: interface design, Web service, Cooperative learning support

1. はじめに

高専・大学等の教員が課題やレポートの出題や提出先，締切日といった情報（以下 To-do）を提供する際は，インターネットの発展に伴い，SNS やグループチャット，メールといった多くのツールが用いられている．情報共有方法は情報ツール以外にも口頭，掲示版，研究室ホワイトボードなどのアナログ手段も根強く使用されている．情報が拡散しているため，学生は To-do を個別に漏れなく管理することが困難な状況にある．また，担任は担当するクラス学生の To-do の進捗状況を把握できず，個別指導がしにくい．

そこで本研究は，学生が To-do の進捗状況を一目で確認でき，学生と教員で共同し To-do を作成・編集・削除するシステムの構築を行う．共同 To-do によって入力情報をクラスや研究室で共有し，To-do の入力忘れを防ぐ．また，共同 To-do の個人の進捗状況を教員が確認できるため To-do が滞っている学生を把握し個別指導がしやすくなる．

2. 要件定義

2.1 想定する場面

システムの使用者は学生と教員を想定する．システムは使用者の役割に応じた機能をそれぞれ提供する．学生は学内での携帯性からスマートフォン（以下スマホ）で，教員は教員室での業務を想定してパソコン（以下 PC）でシステムへアクセスする状況を想定する．本稿で用いる概念とその用語を以下の通り定義する．

チーム

To-do を共同し互いに助け合って To-do リストを作り上げる団体．ここでは，日常で To-do を共有する割合から，クラスと研究室を「チーム」として想定する．

メンバー

チームに所属するユーザ（学生と教員）

共有 To-do

チームによって共有・管理される To-do．クラスであれば課題の提出，研究室であれば書類の作成など，そのチームに所属する全学生がこなすべき To-do が当てはまる．

個人 To-do

1 熊本高等専門学校
National Institute of Technology, Kumamoto College

チームに共有されない個人が管理する To-do.

2.2 要件

1 章で述べた問題点と 2.1 節で述べた想定場面から，システムを構築するための要件を定義した．ここでは，要件を学生と教員という 2 つのロールに分けて示す．

学生

- スマホの OS に依存することなくシステムを使用
- 個人 To-do の追加・編集・削除
- 共有 To-do の追加・編集・削除
- 完了した個人 To-do・共有 To-do
- 個人 To-do・共有 To-do の数を確認
- チームに参加

教員

- PC の OS に依存することなくシステムを使用
- 共有 To-do を追加・編集・削除
- 共有 To-do の数を確認
- チームに参加
- 共有 To-do の学生ごとの完了状況を確認

既存の To-do リスト[1]を扱うアプリの中には，共有 To-do を作成・編集・削除する機能を持つものやグループで To-do を共有する機能 [2]が存在するが，入力した情報を一括してユーザ個々の To-do に割り当てる機能はない．

そこで本システムは，チームに所属するメンバーの誰かが To-do を追加すれば，メンバー全員に To-do が追加される機能を搭載する．これにより，To-do の追加忘れを予防することができ，また，チームメンバーの誰でも編集可能とすることで入力情報の修正が容易になる．

本稿では，システム全体の設計と学生機能の実装について報告する．学生機能を先に実装・公開した理由として，本システムの重要なコンセプトである学生の UX がある．UX を向上させるためには，実際にシステムを使用した学生からのフィードバックが必須であり，そのため学生機能の構築と公開を優先した．教員機能は設計までに留めており構築については別の報告としたい．

2.3 アプローチ

本研究ではシステム管理の効率と To-do の入力効率を目的としたシステムの実現を目指しており，次の 4 点に着目したアプローチをとる．

(1) 容易なシステム管理

本学では，すべての学生にグループウェア Google Workspace [3]を利用した Google アカウントが付与されている．Google Workspace はクラウドベースのグループウェアであり，学校は学生のアカウントをまとめて管理することができる．また，Google の提供するモバイルプラットフォーム Firebase [4]を用いれば，Google のアカウントを使ったユーザ認証が可能となり，セキュリティに配慮した実装がしやすい．Google アカウントでのログイン機能を備えたシステムとすることで，学校側が管理しやすく，学生にとって使いやすく，かつ実装しやすいシステムとなる．

(2) 誰でも使えるシステム

システムをブラウザ上で動作する Web アプリケーション（Web アプリ）として構築する．これにより，デバイスやオペレーティングシステム（OS）に依存しないシステムの開発・実装ができる．また，ユーザはどのデバイスからもシステムを利用できる．Web アプリは開発者側と利用者側，双方にとってメリットのあるシステムとなる．

(3) 使いやすいシステム

システムを SPA（Single Page Application）として構築する [5]．SPA とすることで，ページをフルリロードする必要がなく，動的コンテンツだけを更新できるため，アプリケーションの挙動が高速になりスムーズな動作が実現する．

また，SPA はシンプルで分かりやすいユーザインタフェース（以下 UI）を提供する．学生の多くはスマホでシステムにアクセスするが，少ないタップ数で目的の画面に到達でき，かつ片手でも操作しやすい画面構成となる．また，To-do の重要度や数といった情報が一目でわかるよう，ライブラリを用いてアイコンやアニメーションを工夫する．

(4) 一目でわかる忙しさ

課題の把握漏れの対策として，課題の量と締切日を容易に把握できるように，カレンダーをヒートマップとして表示し課題の進捗状況を可視化する機能を搭載する．

3. システムの機能と運用ポリシー

3.1 ユーザごとの機能

システムの持つ機能について，ユーザごとに分類して以下に説明する．

(1) ユーザが共通して使用する機能

- ログイン・ログアウト機能

ユーザを認証する機能．Google アカウントを用いて認証を行い，教員と学生を判別する．また，ユーザを認証した状態から未認証の状態へ戻す．

- チーム参加機能

共同したいチームへ参加する機能．チームのメンバーとなるには，ユーザ自身がチームに参加するだけである．チームに参加後は，そのチームの共有 To-do を共同管理することができる．

- 共有 To-do の閲覧・追加・編集・削除機能

共有 To-do を閲覧したり，操作したりする機能．チームに所属する全メンバーが使用できる．この機能によってチームが持つ共有 To-do リストの管理が容易になる．

(2) 学生が使用する機能

- 個人 To-do の閲覧・追加・編集・削除・完了機能

通常の To-do アプリにある個人 To-do を操作する機能．

共有 To-do と併せて個人へ提供する. 各個人 To-do は他のユーザやチームメンバーへは共有されない.

- 達成済み To-do の保管機能

達成した To-do は, ホーム画面から達成済み To-do 画面に移動する. この操作により To-do を完了したとみなす. 万一，操作ミスや再提出となった場合でも，達成済み To-do 画面で完了操作を取り消すことができる.

- ヒートマップ機能

To-do の過密度をカレンダーに可視化して表示する機能. 学生ごとに, 未達成の To-do の締切日と現在の日付から過密度を計算し，カレンダーの各日付のセルに色付けして表示する.

(3) 教員が使用する機能

- 共有 To-do 進捗閲覧機能

チーム内の共有 To-do について，各メンバーの進捗状況を確認できる機能．共有 To-do ごとに達成していないユーザをアカウント名で確認できる.

- チームのヒートマップ表示機能

To-do の過密度をカレンダーで閲覧する機能. チームごとに存在しており，そのチームの共有 To-do の締切日と現在の日付をもとに過密度を計算してカレンダーの各日付セルを色で重みづけする.

3.2 運用ポリシー

システム運用を厳密に行うためには細かな権限の設定が必要であるが，年度更新などにおいてシステム管理の手間が必要である. そこで, システム管理の手間を省くため，本システムの運用ポリシーを次のように定める.

個人 To-do

本研究は To-do の管理・把握が前提になるため，共有 To-do 機能に加えて従来型である個人 To-do 機能を搭載する．To-do をまとめて把握できるため，学生がより普段使いしやすいシステムとなる．管理の手間は本学で別に運用している Google アカウントでログインしており，アカウント追加などの管理は不要である.

共有 To-do

共有 To-do はチーム全員が作り上げるものである．つまり，To-do の作成者以外でも，同じチームのメンバーであれば編集・削除が可能であるが, 一方で誰かが悪意を持って編集・削除することも可能になる. そこで，チームすなわち同じクラスあるいは同じ研究室という閉鎖的な中では悪意を持ったメンバーは発生しないと仮定した．つまりチームの権限管理はせず，メンバー全員を対等として扱う. さらに, チームへの追加や退会もシステム管理者ではなくユーザとなる学生の自己申告制とした．チームとは本来関係ないユーザがメンバーになることが可能であるが, 無駄に To-do が多く表示されるだけであり，自ら好んで所属しないと判断した.

次に, To-do をチームメンバーの誰かが追加した際の動作について考える．まず共有 To-do が追加されるユーザは，その時点でチームに所属していたメンバーのみとする. つまり, 途中からチームに参加，あるいは途中で退会するユーザは, チームに参加している間に追加された To-do のみを反映することとした.

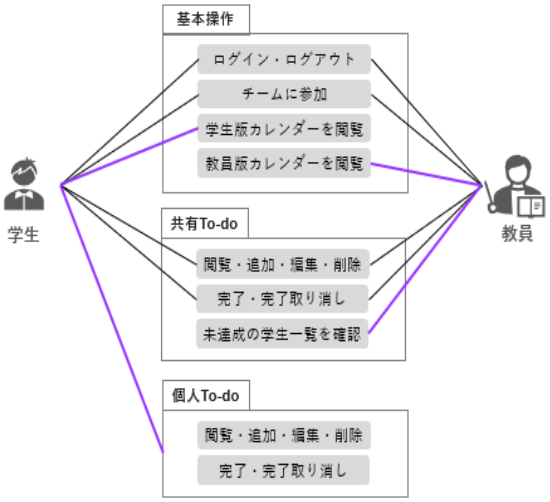

図 3.1 ユースケース図

以上，２つの運用ポリシーにより, システム管理者は最初にチームさえ作成しておけば，その後のチームメンバーや To-do リストの初期化処理は一切不要となる.

あるチームのメンバーとなった教員は，共有 To-do のユーザごとの完了情報を知ることができる．これによって, 教員が，そのチーム，例えば研究室や担任するクラスに所属する学生の To-do 状況を確認することができる.

3.3 ユースケース

3.2 節で述べた運用ポリシーに従い，各機能のユースケースを図 3.1 に示した．紫色の線で示す機能が学生と教員それぞれが使用する独自の機能となっている.

4. システム構成と設計・実装

３章で述べた運用ポリシーに従い，以下の様なシステム設計とした.

4.1 システム構成

開発段階におけるシステム構成を図 4.1 に示す．システムは，ローカルサーバ内にある Web サーバ，API サーバ，データベース（DB）で構成する．ユーザが各端末から Web サーバにアクセスすると，Web サーバが API サーバへリクエストを送信し，API サーバがデータベース内の情報を取得する．ユーザは学内ネットワークに接続し，ブラウザから既定のアドレスにアクセスしシステムを利用する．今回は，開発途中のセキュリティに配慮したため，サーバ・クライアント全てが学校内で完結しているが, システム完成後はサーバを学外に設置し, 運用しても問題はない.

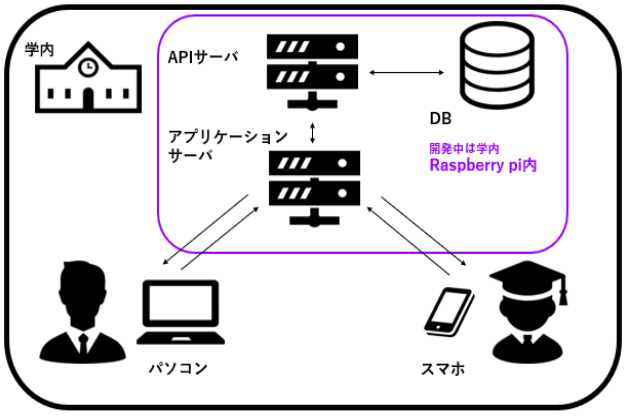

図 4.1　システム構成

4.2 開発・実行環境

本システムの開発における実行環境および開発環境を表 4.1 に示す．本研究で使用するサーバは，Raspberry Pi 4 を採用した．理由として，扱う情報はテキストデータが主であるため Raspberry Pi でも充分と判断しており，また，開発に関する情報が豊富であり，安価でかつ入手しやすいためである．Web アプリとして実装するため，Web サーバ・API サーバの両方で JavaScript を用いてコーディングすることとし，実行環境として Node.js [6]を採用した．開発するシステムは主要機能の多くをライブラリにより実現しており，これらライブラリのパッケージ管理システムとして Node.js と親和性の高い npm [7]を採用した．

4.3 ライブラリ

使用したライブラリとバージョンを表 4.2, 表 4.3 に示す．表 4.2 はフロントエンドで Web サーバ構築に，表 4.3 はバックエンドで API サーバ構築に使用した．フロントエンドには，UI ライブラリとして React [8]を使用する．React は Meta 社が開発・公開しているもので，UI の各要素を構成部品（コンポーネント）として小さく分けて考える「コンポーネント指向」が採用されている．モジュールやクラスがカプセル化されており，メンテナンス性が高く拡張性が高い．加えて，CSS フレームワークとして，React と相性がよい Material UI（MUI）[9]を使用する．MUI は Google が公開したマテリアルデザインと呼ばれるデザイン手法のガイドラインに則っており，シンプルで使いやすい UI を構築できる．また，バックエンドでは Web フレームワークとして Express [10]を使用する．軽量でカスタマイズ性が高く Node.js 上で動作するため，API の設計・実装用に採用した．

4.4 データベース

データベースは Raspbian OS が推奨する MariaDB を使用した．本システムのデータベース設計を，図 4.2 と図 4.3 に示す．図 4.2 は共有 To-do 機能実装に用いるテーブルで，図 4.3 は個人 To-do 機能実装に用いるテーブルである．主要テーブルにおけるカラムの説明を表 4.4 に示す．カラムやテーブルの命名では，シンプルな英単語を用い，後置修飾で統一性を持たせており可読性を重視した．

表 4.1　開発環境

項目	名前
開発マシン	Raspberry Pi 4
OS	Raspbian GNU/Linux 10
開発言語	JavaScript
データベース	MariaDB v10.6.11
JavaScript 実行環境	Node.js v16.18.1
パッケージ管理システム	npm v8.19.2
仕様想定ブラウザ	Google Chrome

共有 To-do 機能では，共有 To-do の情報を格納するテーブルと進捗状況を格納するテーブルを図 4.3 のように分け，外部キーを用いて進捗状況テーブルから情報テーブルを参照する．また，共有 To-do が追加された際に，進捗状況のデータを含むレコードをメンバーの人数分作成するよう設計することで，メンバーごとの進捗状況を保持できるようにした．各テーブルの用途を以下に示す．

todos_private テーブル

個人 To-do の情報と進捗状況を格納するテーブルである．個人 To-do の追加・編集・削除・追加の際に参照する．

teams テーブル

チームの名前と ID を紐づけるテーブル．

teachers テーブル・members テーブル

ユーザが所属するチームを管理するテーブルである．チーム ID とユーザ ID を紐づける．ロールを分けるため，教員は teachers，学生は members へそれぞれ登録する．

todos_shared テーブル

共有 To-do の情報を格納するテーブルである．共有 To-do の追加・編集・削除の際に参照する．

todos_shared_status テーブル

共有 To-do の各メンバーの進捗状況を格納するテーブルである．各個人の共有 To-do の進捗管理で参照する．

表 4.2　フロントエンド開発に用いたライブラリ

ライブラリ名	バージョン
React	17.0.2
React-router-dom	6.2.2
Material-UI	5.0.0
axios	0.27.2
Firebase	9.15.0
react-calendar	4.0.0
date-fns	2.29.3
styled-components	5.3.5

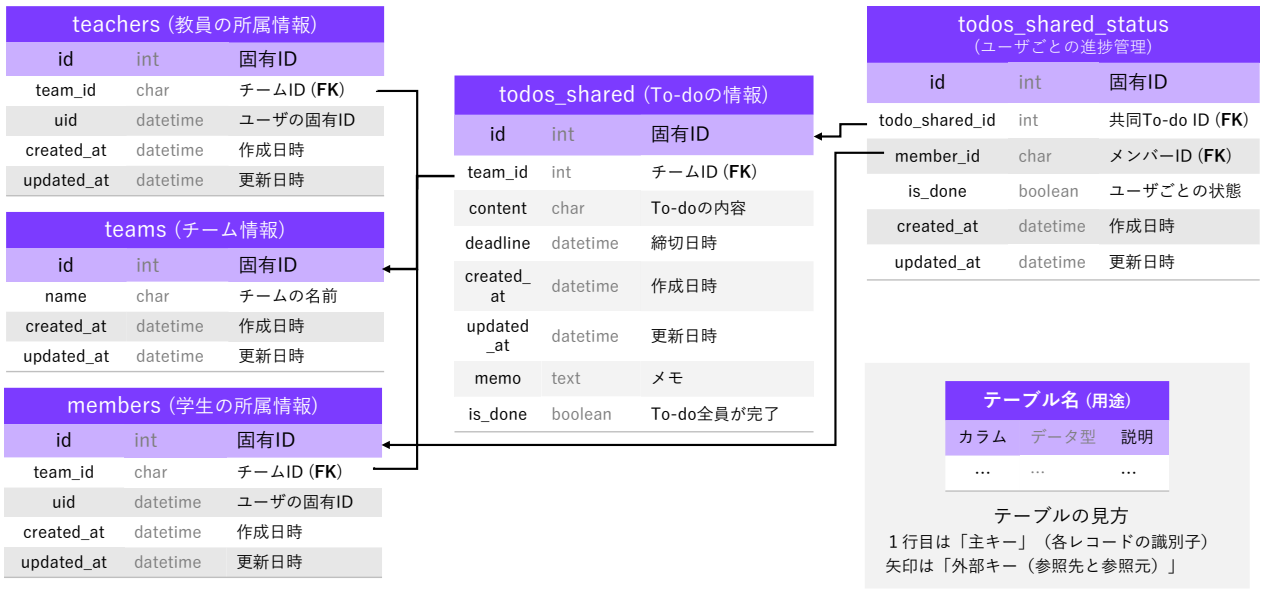

図 4.2　共有 To-do に関する DB テーブル設計

表 4.3　バックエンド開発に用いたライブラリ

ライブラリ名	バージョン
Express	4.18.1
CORS	2.8.5
Firebase-admin	11.2.1
MySQL	2.18.1
nodemon	2.0.20

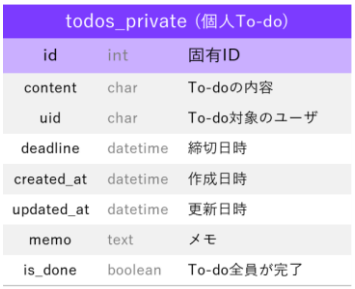

図 4.3　個人 To-do に関する DB テーブル設計

表 4.4　各カラムの説明

カラム名	説明
name	チームの名前
content	To-do の説明（タイトル）
uid	ユーザの固有 ID．初回ログイン時に Firebase から発行される
deadline	To-do の締切日
created_at	To-do レコードの作成日
updated_at	To-do レコードの最終更新日
memo	To-do のメモ（補足説明など）
is_done	To-do が達成されたかを判断する（1：達成済，0：未達成）

4.5　ヒートマップ

本節ではヒートマップの画面構成とヒートマップの色を決定する際の過密度の計算方法について説明する．

4.5.1 画面構成と機能

ヒートマップ画面はカレンダー表示と To-do リスト表示で構成する．

学生用カレンダーのヒートマップ表示

学生用カレンダーのヒートマップには，各ユーザの課題の過密度を表示する．過密度の計算対象には，ログイン中のユーザが未完了の個人 To-do と共有 To-do を使用する．

教員用カレンダーのヒートマップ表示

教員用カレンダーのヒートマップとして，教員が指定したチームの課題の過密度を表示する．過密度の計算には，締切日が過ぎていないチームの共有 To-do を使用する．

学生用 To-do リスト表示

カレンダーのヒートマップ表示の下に To-do リストを表示する．表示する To-do は，ログイン中のユーザが未完了で，かつ To-do の締切日がタップし選択した日付を過ぎていない個人・共有 To-do である．これにより，To-do の締切日による絞り込み表示ができる．表示された To-do の 1 つをタップすると，その To-do の詳細情報（メモ・作成日）を確認できる．

4.5.2 過密度の算出アルゴリズム

学生のヒートマップ表示に使う過密度の計算アルゴリズムについて述べる．

(1)　To-do の重要度の計算

式(1)に示す通り，現在の日付 now と締切日 deadline から To-do の締切日までの残り日数 days_left を求め，この値から To-do の現時点での重要度 point を計算する．

$$\text{days\_left} = \text{deadline} - \text{now} \tag{1}$$

次に point の算出方法を式(2)に示す.

$$\text{point} = \begin{cases} \dfrac{1}{\text{days\_left}} & (\text{if days\_left} > 0) \\ 1 & (\text{if days\_left} = 0) \end{cases} \tag{2}$$

締切日が近いほど To-do の緊急性があると判断し，point は days_left に反比例する値とした. ゼロ除算防止のため，days_left = 0のときはpoint = 1と定める. すべての To-do は締切日までに完了すると仮定し，締切日が過ぎていない To-do を過密度の計算対象とした.

i 番目の To-do に対する重要度は式(3)により決定する.

$$\text{points}[i][j] = \text{point}\ (0 \leqq j \leqq \text{days\_left}) \tag{3}$$

カレンダーの各日付に対応した重要度を格納する配列 points を用意し，$0 \leq j \leq$ days_leftを満たすj日目に対して等しく point を配分する.

(2) To-do の重要度の合計

(1)の計算アルゴリズムに従い，n個の To-do すべてに対し，重要度を計算する. 計算式を式(4)に示す. 任意の日付（k日目）の重要度 points_all[k]は，points[i][k]の合計値として計算できる.

$$\text{points\_all}[k] = \sum_{i=1}^{n} \text{points}[i][k] \tag{4}$$

(3) 色による重みづけ

過密度を示す数値が格納された配列 points_all からヒートマップを生成する. 色は赤ベースとし，過密度が低いほど色は白く，高いほど赤くなるようにした. 開発したシステムで使用した過密度と RGB 値との対応を表 4.5 に示す. 日付セルを塗りつぶす際，完全な赤色（RGB(255, 0, 0)）だと鮮やかすぎる. そこで，色域を制限しかつ統一性を持たせるため，G 値・B 値には 255/6=42.5 を 5 等分した値を設定した.

5. 作成したシステムの画面表示

開発したシステムの画面表示について説明する. 主要な画面は，ホーム画面，To-do 追加画面，ヒートマップ画面の 3 つある.

5.1 共通ヘッダー・フッター

- 学生ホーム画面，完了 To-do 一覧画面，ヒートマップ画面では，共通のヘッダーとフッターが表示される
- ヘッダーは，ログイン中のアカウントのアイコンとID，チーム参加ボタン，ログアウトボタンからなる
- フッターは 3 つの画面遷移ボタンからなる
- チーム参加ボタンを押すとチーム参加画面に遷移する
- フッターの各ボタンを押すと対応した画面に遷移する

表 5.1 重みづけのレベル

過密度		RGB（255, x, x）
1.0 以上		43
0.7 以上	1.0 未満	85
0.5 以上	0.7 未満	128
0.2 以上	0.5 未満	170
0 より上	0.2 未満	212

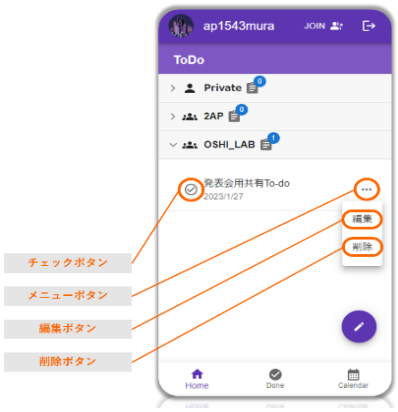

図 5.1 学生用ホーム画面

5.2 チーム参加画面

- チーム選択フィールド．参加ボタンから構成される
- チーム選択後，参加ボタンをクリックするとチームへ参加ができる．一度に複数のチームを選択でき，参加が可能.

5.3 学生用ホーム画面

- 作成した画面を図 5.1 に示す. ヘッダー，フッター，個人 To-do リスト，共有 To-do リスト，チェックボタン，メニューボタン，To-do 追加ボタンから構成される.
- 各 To-do の左にあるチェックボタンをクリックすると，To-do 完了確認アラートが表示される
- To-do の右にあるメニューボタンをクリックすると編集ボタンと削除ボタンが現れる．編集ボタンを押すと To-do 編集画面に遷移し，削除ボタンを押すと To-do 削除確認画面が表示される
- To-do 追加ボタンを押すと，To-do 追加画面に遷移する

5.4 To-do 追加画面

- 個人 To-do や共有 To-do を追加するページであり，チーム選択フィールド，To-do フォーム，メモフォーム，〆切日フィールド，To-do 登録ボタンから構成される．作成した画面を図 5.2 に示す.
- 参加しているチームのみがチーム選択フィールドに表示され，チーム選択を行う

5.5 学生用ヒートマップ画面

- ヘッダーとフッター，カレンダー，リストからなる

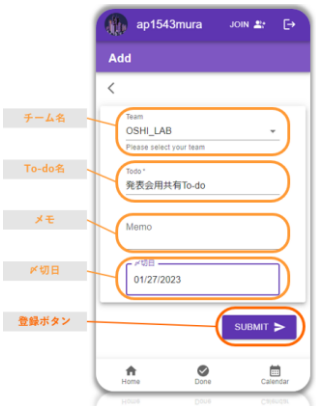

図 5.2　To-do 追加画面

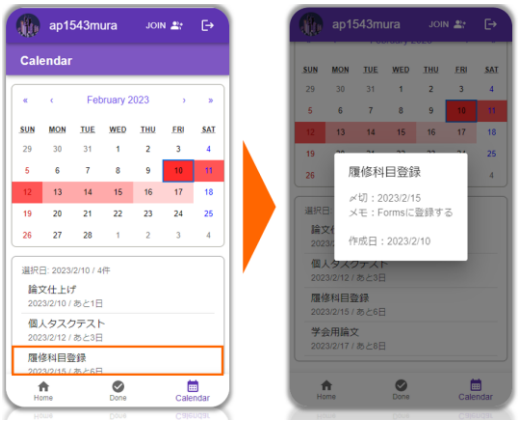

図 5.3　ヒートマップ画面

- 画面上部には，To-do の過密度に応じて色で重みづけされたカレンダーが表示される
- 画面下部には，カレンダー上の選択された日付に応じたリストが表示される．表示の例を図 5.3 に示す
- 選択された日付のセルは青い実線で囲まれている．画面にアクセスした直後は現在の日付が選択されている
- リストには各 To-do の名前・締切日・締切日までの日数に加え，ヘッダーに To-do の件数が表示される
- リスト上の To-do をクリックするとダイアログウィンドウが表示され，To-do のメモ・作成日等の詳細情報を確認できる．ウィンドウの外をタップすると閉じる

6. 評価

学生に本システムを利用してもらった後にアンケートを実施した．まず，本校の学生 22 名（本科 3 年生～専攻科 2 年生）に対して，システムの操作方法を教え，10 分程度使用してもらった．その後，5 点満点の点数方式，およびキーワードごとの記述式アンケートを実施した．機能や操作感，共有 To-do やヒートマップなど，本システムの有用性に関する質問は 5 点満点で入力してもらった．

表 6.1　点数方式アンケートの結果

	質問	平均点数
Q1	クラスのメンバーが個人でそれぞれタスクを登録するよりも，クラスの誰かが一括で登録する方が好ましい	4.2
Q2	このシステムにより自分の To-do を管理ができた	4.2
Q3	ヒートマップで To-do の量を把握できたのは良かった	4.2
Q4	クラスや研究室別にタスクが管理できたのは良かった	4.6
Q5	タスクの登録項目の内容や数は適切だった	4.6
Q6	このシステムの使い方は分かりやすかった	4.5

点数方式のアンケートの結果を表 6.1 に，記述式アンケートの結果をキーワードごとに表 6.2 に示す．点数式アンケートでは，全体として 4.2 以上と高い評価が得られた．[Q1] [Q4] が 5 点満点中それぞれ 4. 2，4.6 となり，共有 To-do 機能を便利と感じる学生が多かったことが伺える．また，[Q3] の結果から，ヒートマップに対する評価が 4.2 となった．最後に，[Q2] で学生自身による To-do 管理の評価が 4.2 となった．

記述式アンケートについては，点数式アンケート[Q3] と同様，カレンダーで可視化するという手法を高く評価した学生がいた．また，重みづけをする色の変更や To-do カテゴリごとのヒートマップなど，機能を提案する回答が多くあった．

7. 結論

本研究では，共有 To-do の入力を簡便にするためのシステムを設計・開発した．本システムは学生と教員を対象者としており，現在，学生用の機能は構築できている．学生はスマートフォンでの使用を想定しており，React を用いた SPA アプリケーションとすることでストレスなく使用することができる．共有 To-do の実現にあたり，「チーム」と呼ばれるグループを提案し，チームに所属する学生・教員全員が To-do を管理するシステムとして開発した．これにより，チームに所属した学生の誰かが To-do を入力することで，全員が To-do を入力したのと同じ状況を作り出すことができ，チームによる効率的な To-do 入力が実現できた．

開発したシステムを一般学生に使用してもらい，点数形式でのアンケート調査を実施した．その結果，「システムにより自分の To-do を管理することができた」，「クラスや研究室別に To-do が管理できたのはよかった」，「ヒートマップで To-do の量を把握できたのは好ましかった」

のそれぞれの評価において，5 点満点中 4.2 点以上の評価を得た．このことから，研究の目的である学生の共有 To-do 管理は達成できたといえる．

今後の展望として，教員画面の実装を予定している．共有 To-do の各学生の進捗を確認できる機能や，各チームのヒートマップを確認できる機能を実装する予定である．学生用機能の開発と同様，実際に教員に使用してもらい，フィードバックを得ながら改良を重ねていく．

謝辞 本研究の遂行にあたり，熊本高等専門学校 技術・教育支援センターの岩本舞 氏には研究の遂行にあたり，日頃より有益なご助言を頂いた．また，評価にあたり熊本高等専門学生にはアンケートに回答して頂いた．ここに感謝の意を表する．

参考文献

[1] Microsoft To Do, https://todo.microsoft.com/tasks/. (参照 2023-02-13).
[2] Garoon, https://garoon.cybozu.co.jp/. (参照 2023-02-t3).
[3] Google Workspace, https://workspace.google.co.jp/intl/ja/. (参照 2023-02-13).
[4] Firebase, https://firebase.google.com/?hl=ja (参照 2023-02-13).
[5] SPA とは ：Devblog, https://blog.shiftasia.com/spa-single-page-application/. (参照 2023-02-13).
[6] Home | Node.js, https://nodejs.org/en/. (参照 2023-02-13).
[7] npm, https://www.npmjs.com/. (参照 2023-02-13).
[8] React, https://reactjs.org/. (参照 2023-02-14).
[9] MUI, https://mui.com/. (参照 2023-02-14).
[10]Express | Home, https://reactjs.org/. (参照 2023-02-14).

表 6.2 記述式アンケートの結果

キーワード	アンケート結果
To-do	− To-do 締め切りを入力する部分で日付のみではなく，時間まで決めたい． − 重要度を設定し色が変わる仕組みにしてほしい． − 詳細ボタンをクリックするのが面倒なのでホームページにて一目ですべての To-do がわかるようにして欲しい． − 誰が追加した共有 To-do なのかをわかるようにして欲しい． − 共有 To-do ではみんなの進捗が知りたい．
通知	− 期限が迫ると通知が来るようにすると使用してもらいやすくなる． − 共有 To-do が追加された際に通知が来ると使いやすいと思う． − ウィジェットへの追加や通知が来る設定をできるともっとよかった．
ヒートマップ	− カレンダーの色を自分で変更できると個性が出て良いなと思った． − カレンダーでタスク確認できるのがすごくよかった．
チーム	− チームを作れるようになりたい． − チームメンバーの確認や脱退などの，チームの管理画面が欲しい．
操作感	− 簡単な操作でわかりやすかった． − ローディング画面はバーかサークルどちらかでよいと思う． − サクサク動いて使いやすいなと感じた．