令和５年度卒業研究報告書

電子出席簿・欠席連絡・時間割表

統合させたシステムの開発

情報技術科　石川　孝悌

石川　太郎

岩城　雄政

多田　小太郎

指導教員　　石舘　勝好

内容

第 1 章　はじめに……3
第 2 章　研究概要……4
2.1 現行のシステムについて……4
2.1.1　電子出席簿……4
2.1.2　時間割システム……4
2.1.3　欠席連絡システム……4
2.2 現行のシステムによる問題……5
2.2.1　電子出席簿の課題……5
2.2.2　時間割システムの課題……6
2.2.3　データベースの課題……7
2.3 改善案とシステム設計方針……8
2.3.1　電子出席簿の改善案……8
2.3.2　時間割システムの改善案……8
2.3.3　データベースの改善案……8
2.3.4　システム設計方針……9
2.4 開発環境……10
第 3 章　システム設計（全体）……11
3.1　サーバ構成……11
3.2　データベース設計……12
3.2.1 各テーブルの構造……14
3.3　ページ構成……22

第 4 章　　時間割表.....23
4.1 時間割表の機能.....23
4.2 時間割作成のインタフェースについて.....26
4.3 学生側の時間割インタフェースについて.....39
第5章　　電子出席簿について.....43
5.1　基本動作.....43
5.2　セキュリティ.....45
5.3　出欠確認ページについて.....47
第6章　　バックエンドシステムの実装.....65
第7章　　まとめ.....78
7.1 データベース・Java クラス.....78
7.2　時間割システム.....78
7.3　電子出席簿.....79
第8章　　おわりに.....80

第1章　はじめに

私たちは普段，過去の卒業生が開発したシステムを日常生活で使用している．令和2年度には，「欠席連絡システム」令和3年度には，「時間割システム」，昨年度は「電子出席簿」が作成された．

今回，私たちは，上記のシステムをより使いやすいものに作り直そうと考えた．

私たちがこの研究を選んだ理由は3つある．

1つ目は，学んだことを活かして，上流工程からWebシステムを開発してみたいと考えたからである．授業の中では，なかなか経験できない要件定義や設計などの上流工程を経験し，システム開発への理解をより深められるのではないかと考えたためである．

2 つ目は，現在使用している時間割，電子出席簿，欠席連絡システムにおいて見づらい，使いづらい，エラーが起こる等のいくつかの問題があり，システムの改善が必要だと考えたためである．

3 つ目は，時間割，電子出席簿，欠席連絡システムは，それぞれ別々のデータベースを使用しており，各システムが複雑に繋がりあっているため，改修するのは困難であると言われていたためである．

これらの理由から，私たちは現行の3つシステムを統合し，新たなシステムを作り直そうと考えた．

第 2 章　　研究概要

2.1　現行のシステムについて

現在，情報技術科では電子出席簿，時間割，欠席連絡の 3 つのシステムを使用している．このシステムは，卒業生が卒業研究で作成したものである．

2.1.1　電子出席簿

このシステムは，スマートフォンで学生の出欠確認をできるシステムである．

システムの主な流れは，電子出席簿システムで入力したデータをデータベースに格納する．さらに，必要であれば出欠状況を CSV ファイルとして出力し，学生の出席状況を一見して分かるようにしている．

2.1.2　時間割システム

このシステムは，スマートフォンや PC で時間割の作成，閲覧ができるシステムである．

システムの主な流れは，教員側の画面で時間割を作成し，データベースに格納する．そして，学生側の画面でデータベースを参照し，閲覧することができる．

2.1.3　欠席連絡システム

このシステムは，スマートフォンや PC で欠席連絡を送ることができるシステムである．

システムの主な流れは，欠席連絡システムで届け出を提出し，データをデータベースに格納する．さらに，届け出データをもとに PDF・メールを作成する．

2.2　現行のシステムによる問題

現行のシステムにはいくつか課題点があり，それにより日常生活に困難が生じている．主に，電子出席簿，時間割システム，そして別々に構築された３つのデータベースに課題点が挙げられる．

2.2.1　電子出席簿の課題

現行の電子出席簿の課題点は２つある．

１つ目は，１日全体を通して出席状況の確認ができない点である．現行の電子出席簿は１コマ単位での表示のため，１日全体の出席状況を確認したい際に手間がかかってしまう．（図 2.2.1）

２つ目は，出欠確認の完了ボタンを押すとセッションエラーが起きる点である．

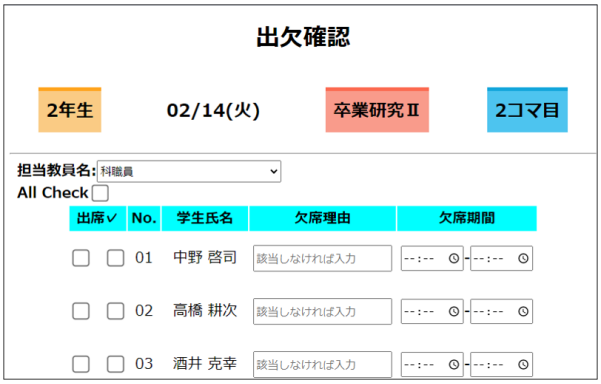

図 2.2.1　現行の電子出席簿

2.2.2　時間割システムの課題

現行の時間割システムの課題点は２つある．

１つ目は，学年の切り替えタブが分かりづらい点である．時間割表の罫線と切り替えタブが一致しているため，どの学年の時間割表なのか区別が難しい．（図 2.2.2）

２つ目は，１コマの情報の中に授業内容やイベント情報（文化祭など）を記入する欄がない点である．（図 2.2.3）

図 2.2.2　現行の時間割システム（生徒側）

図 2.2.3　現行の時間割システム（教員側）

2.2.3　データベースの課題

電子出席簿、時間割、欠席連絡の３つのシステムがそれぞれデータベースを持ち、各システムが複雑に繋がりあっているため，管理・修正が困難である．（図 2.2.4）

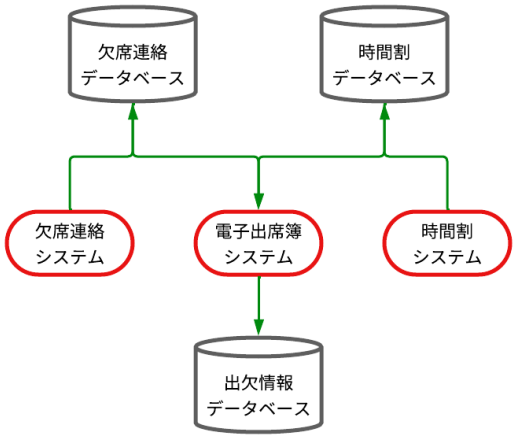

図 2.2.4　現行のシステムのデータベースモデル

2.3　改善案とシステム設計方針

2.2 で示した課題から出された改善案は次のとおりである.

2.3.1　電子出席簿の改善案

教員から，１コマ単位での出欠確認画面ではなく，１日単位の表示にして欲しいとの要望があり，一見して学生の出欠状況を見ることができるようにする．セッションエラーについては，現行の電子出席簿の機能をベースに１からシステムを作成することで，改善できるのではないかと考えた.

2.3.2　時間割システムの改善案

学年の切り替えをタブではなく，独立したボタンを設置することで，どの学年のページなのか分かりやすくする．コマを作成する際に，コマの中にイベント情報を記入する欄を作る．これによって，時間割を見るだけで授業の内容の変更なども把握できるようになる.

2.3.3　データベースの改善案

今あるシステムのデータベースを改修するのではなく，新たに１つに統合したデータベースを作成する．そして，必要なテーブルのみを選出し，データ項目の重複を無くし，より運用・管理のしやすいデータベースにする.

2.3.4　システム設計方針

2.3.4 で提案した改善案をもとにシステム設計をモデル化したものが図 2.3.5 である.

現行のシステムモデル（図 2.2.4）と比較すると，データベースが統一され、システム全体の流れがシンプルなものとなっている.

機能面の改善案をもとにした構想は、第 4 章、第 5 章で後述する.

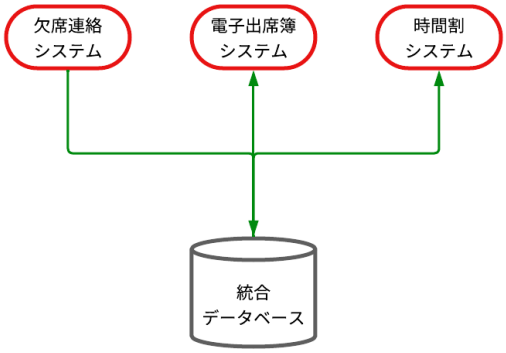

図 2.3.1　今回のシステムモデル

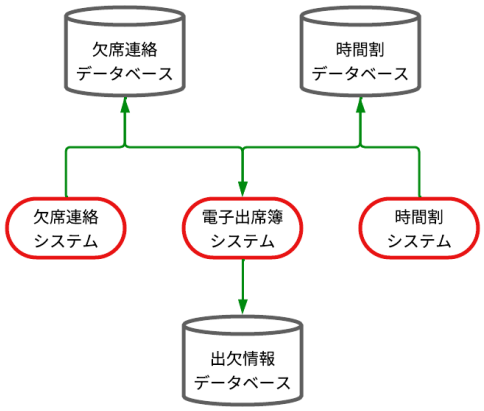

図 2.2.4　現行のシステムモデル（再掲）

2.4　開発環境

システムの開発環境及び動作環境を表 2.4.1 に示す．本研究では，本番用サーバで開発を進めた．

過去の研究との相違点は，

PHP ではなく，Java（Tomcat）を使っていること，

開発環境と動作環境で同様のサーバを使っていることの２点である．

上記の理由は，

Java（Tomcat）は授業で使用したことがあり，なじみがあることと，

同一のサーバを使用することで作業工程を少なくするためである．

表 2.4.1　開発環境及び動作環境

OS	Linux，　Windows10
使用言語	HTML，　CSS，　Java(Tomcat)，JavaScript，
データベース	MySQL
開発ソフトウェア	Eclipse
Web サーバ	情報技術科内部サーバ （http://reciente．iwate-it．ac．jp/iitadm）

第 3 章　　システム設計（全体）

3.1　サーバ構成

本研究で使用している内部サーバと外部サーバの構成について説明する．

内部サーバは，校内サーバである reciente.iwate-it.ac.jp を使用し，外部サーバは v7.iwate-it.ac.jp を使用する．これは，現行の 3 システムの方式を踏襲することにしている．

サーバを 2 つに分けている理由は，校内サーバは校内ネットワークに接続していないと使用することができないため，時間割システムや欠席連絡システムといった学外からの使用も想定されるシステムについては学外からでもアクセスすることができる外部サーバが必要となるからである．

しかし，外部サーバに個人情報を含むデータを保存しておくのは，個人情報の漏洩につながってしまうため，一時的にデータの保存はされるが，内部側から定期的に外部サーバへアクセスし，個人情報を含むデータがある場合，そのデータを内部側に移動し，外部にあるデータを削除するという仕組みとしている．

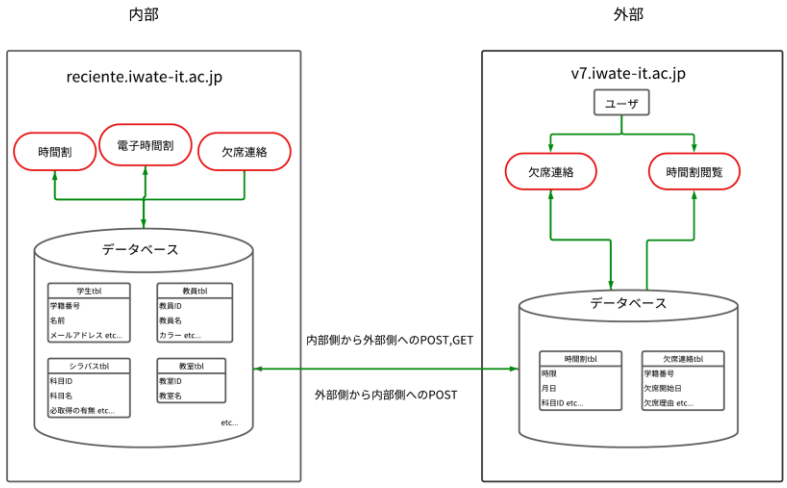

図 3.1.1　サーバ構成図

本研究では，内部サーバのみで外部サーバに関しては作成できていない．

ユーザ及びデータベースは表 3.1.1 のとおりである．

表 3.1.1　ユーザ及びデータベースの設定項目

設定項目	内容
Linux ユーザ名	iitacadm
Linux ユーザパスワード	takataroyoukota
MySQL ユーザ名	iitacadm
MySQL パスワード	takataroyoukota
MySQL データベース名	iitacadm

3.2　データベース設計

図 3.2.1 はデータベース内のテーブルの関係を表した ERD である．

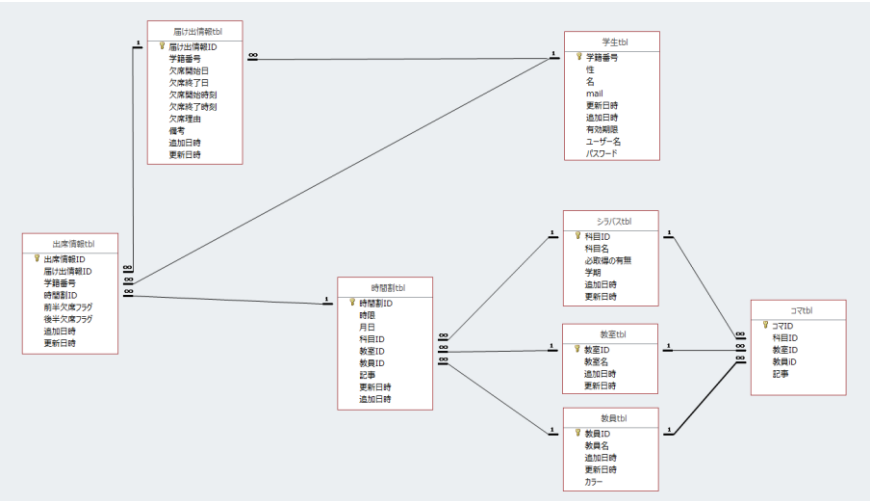

図 3.2.1　ERD

各テーブルの目的は次のとおりである．なお，追加日時，更新日時は，処理のタイミングが後から検証できるように，ほとんどのテーブルに設けている．

（1） 学生

欠席連絡システム，電子出席簿システムの機能で使用する学生のマスタデータである．

（2） シラバス

時間割システムの機能で使用するシラバスのマスタデータである．

（3） 教室

時間割システムの機能で使用する教室のマスタデータである．

（4） 教員

時間割システムの機能で使用する教室のマスタデータである．

（5） コマ

時間割システムの機能で使用するコマ情報の格納するテーブルである．シラバスの科目 ID，教室の教室 ID，教員の教員 ID の組で表される．

（6） 時間割

教員が作成した時間割が格納される．時間割システム，電子出席簿で使用される．コマと同じ，シラバスの科目 ID，教室の教室 ID，教員の教員 ID の組に加えて，実施月日と時限，記事を持つ．

（7） 届け出情報

学生からの欠席連絡の情報を格納する．電子出席簿で使用される．

（8） 出席情報

各時間割毎の学生の出席状況を表す．産技短は，ひとコマ 90 分で，前半 45 分，後半 45 分が出欠席の最小単位となる．前半欠席フラグ，後半欠席フラグそれぞれ，0 の場合は出席，1 の場合は欠席として表す．

3.2.1 各テーブルの構造

（1） 学生テーブル

リスト 3.2.1 学生テーブル作成 SQL

```
CREATE TABLE `student` (
  `student_number` varchar(6) NOT NULL COMMENT '学籍番号',
  `number` int NOT NULL COMMENT '出席番号',
  `first_name` text NOT NULL COMMENT '性',
  `last_name` text NOT NULL COMMENT '名',
  `mail_address` varchar(30) NOT NULL COMMENT 'メールアドレス',
  `expiration_date` date NOT NULL COMMENT '有効期限',
 `username` varchar(6) NOT NULL COMMENT 'ユーザーネーム',
 `password` varchar(64) NOT NULL COMMENT 'パスワード',
  `update_date_and_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
  '更新日時',
  PRIMARY KEY(student_number)
)
```

#	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	student_number	varchar(6)	utf8mb4_0900_ai_ci		いいえ	なし	学籍番号	
2	number	int			いいえ	なし	出席番号	
3	first_name	text	utf8mb4_0900_ai_ci		いいえ	なし	性	
4	last_name	text	utf8mb4_0900_ai_ci		いいえ	なし	名	
5	mail_address	varchar(30)	utf8mb4_0900_ai_ci		いいえ	なし	メールアドレス	
6	expiration_date	date			いいえ	なし	有効期限	
7	username	varchar(6)	utf8mb4_0900_ai_ci		いいえ	なし	ユーザーネーム	
8	password	varchar(64)	utf8mb4_0900_ai_ci		いいえ	なし	パスワード	
9	update_date_and_time	timestamp			いいえ	CURRENT_TIMESTAMP	更新日時	DEFAULT_GENERATED

図 3.2.2　学生テーブル

（2）　シラバステーブル

リスト 3.2.2　シラバステーブル作成 SQL

```
CREATE TABLE `syllabus` (
  `subject_id` varchar(5) NOT NULL COMMENT '科目 ID',
  `subject_name` varchar(20) NOT NULL COMMENT '科目名',
  `flag` tinyint(1) NOT NULL COMMENT '必取得の有無',
  `semester` text NOT NULL COMMENT '学期',
  `update_date_and_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
  '更新日時',
  PRIMARY KEY(subject_id)
)
```

#	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	subject_id	varchar(5)	utf8mb4_0900_ai_ci		いいえ	なし	科目ID	
2	subject_name	varchar(20)	utf8mb4_0900_ai_ci		いいえ	なし	科目名	
3	flag	tinyint(1)			いいえ	なし	必取得の有無	
4	semester	text	utf8mb4_0900_ai_ci		いいえ	なし	学期	
5	update_date_and_time	timestamp			いいえ	CURRENT_TIMESTAMP	更新日時	DEFAULT_GENERATED

図 3.2.3　シラバステーブル

（3） 教室テーブル

リスト 3.2.3 教室テーブル作成 SQL

```
CREATE TABLE `class` (
  `class_id` int NOT NULL AUTO_INCREMENT COMMENT '教室 ID',
  `class_name` text NOT NULL COMMENT '教室名',
  `update_date_and_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
  '更新日時',
  PRIMARY KEY(class_id)
)
```

#	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	class_id	int			いいえ	なし	教室ID	AUTO_INCREMENT
2	class_name	text	utf8mb4_0900_ai_ci		いいえ	なし	教室名	
3	update_date_and_time	timestamp			いいえ	CURRENT_TIMESTAMP	更新日時	DEFAULT_GENERATED

図 3.2.4 教室テーブル

（4） 教員テーブル

リスト 3.2.4 教員テーブル作成 SQL

```
CREATE TABLE `teacher` (
  `teacher_id` int NOT NULL AUTO_INCREMENT COMMENT '教員 ID',
   `teacher_name` text NOT NULL COMMENT '教員名',
  `color` text NOT NULL COMMENT 'カラー',
  `update_date_and_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
  COMMENT '更新日時',
  PRIMARY KEY(teacher_id)
)
```

#	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	teacher_id	int			いいえ	なし	教員ID	AUTO_INCREMENT
2	teacher_name	text	utf8mb4_0900_ai_ci		いいえ	なし	教員名	
3	color	text	utf8mb4_0900_ai_ci		いいえ	なし	カラー	
4	update_date_and_time	timestamp			いいえ	CURRENT_TIMESTAMP	更新日時	DEFAULT_GENERATED

図 3.2.5　教員テーブル

（5）コマテーブル

リスト 3.2.5　コマテーブル作成 SQL

```
CREATE TABLE `frame` (
  `frame_id` int NOT NULL AUTO_INCREMENT COMMENT 'コマ ID',
  `subject_id` varchar(20) NOT NULL COMMENT '科目 ID',
  `teacher_id` int NOT NULL COMMENT '教員 ID',
  `class_id` int NOT NULL COMMENT '教室 ID',
  `news` text NOT NULL COMMENT '記事',
  `update_date_and_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
  '更新日時',
  PRIMARY KEY(frame_id),
  ADD KEY `class_id` (`class_id`),
  ADD KEY `subject_id` (`subject_id`),
  ADD KEY `teacher_id` (`teacher_id`)
)
```

外部キー制約

リスト 3.2.6　コマテーブル外部キー設定 SQL

```
 ALTER TABLE `frame`
   ADD FOREIGN KEY (`class_id`) REFERENCES `class` (`class_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT,
   ADD FOREIGN KEY (`subject_id`) REFERENCES `syllabus` (`subject_id`) ON
DELETE RESTRICT ON UPDATE RESTRICT,
   ADD FOREIGN KEY (`teacher_id`) REFERENCES `teacher` (`teacher_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT;
```

#	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	frame_id	int			いいえ	なし	コマID	AUTO_INCREMENT
2	subject_id	varchar(20)	utf8mb4_0900_ai_ci		いいえ	なし	科目ID	
3	teacher_id	int			いいえ	なし	教員ID	
4	class_id	int			いいえ	なし	教室ID	
5	news	text	utf8mb4_0900_ai_ci		いいえ	なし	記事	
6	update_date_and_time	timestamp			いいえ	CURRENT_TIMESTAMP	更新日時	DEFAULT_GENERATED

図 3.2.6　コマテーブル

（6）時間割テーブル

リスト 3.2.7　時間割テーブル作成 SQL

```
CREATE TABLE `timetable` (
  `timetable_id` int NOT NULL COMMENT '時間割 ID',
  `period` int NOT NULL COMMENT 'コマ目',
  `time` date NOT NULL COMMENT '月日',
  `subject_id` varchar(5) NOT NULL COMMENT '科目 ID',
  `class_id` int NOT NULL COMMENT '教室 ID',
  `teacher_id` int NOT NULL COMMENT '教員 ID',
  `frame_id` int NOT NULL COMMENT 'コマ ID',
  `news` text NOT NULL COMMENT '記事',
  `update_date_and_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
  '更新日時',
  PRIMARY KEY(timetable_id)
```

（7）外部キー制約

リスト 3.2.8　時間割テーブル外部キー設定 SQL

```
ADD FOREIGN KEY (`subject_id`) REFERENCES `syllabus` (`subject_id`),

ADD FOREIGN KEY (`class_id`) REFERENCES `class` (`class_id`),

ADD FOREIGN KEY (`teacher_id`) REFERENCES `teacher` (`teacher_id`);
```

#	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	timetable_id	int			いいえ	なし	時間割ID	AUTO_INCREMENT
2	period	int			いいえ	なし	コマ目	
3	time	date			いいえ	なし	月日	
4	subject_id	varchar(5)	utf8mb4_0900_ai_ci		いいえ	なし	科目ID	
5	class_id	int			いいえ	なし	教室ID	
6	teacher_id	int			いいえ	なし	教員ID	
7	frame_id	int			いいえ	なし	コマID	
8	news	text	utf8mb4_0900_ai_ci		いいえ	なし	記事	
9	update_date_and_time	timestamp			いいえ	CURRENT_TIMESTAMP	更新日時	DEFAULT_GENERATED

図 3.2.7　時間割テーブル

（8）届け出情報テーブル

リスト 3.2.9　届け出情報テーブル作成 SQL

```
CREATE TABLE `notification_information` (

  `notification_id` int NOT NULL COMMENT '届け出情報 ID',

  `student_number` varchar(6) NOT NULL COMMENT '学籍番号',

  `absence_start_date` date NOT NULL COMMENT '欠席開始日',

  `absence_end_date` date NOT NULL COMMENT '欠席終了日',

  `absence_start_time` time NOT NULL COMMENT '欠席開始時間',

  `absence_end_time` time NOT NULL COMMENT '欠席終了時間',

  `reason_for_absence` text NOT NULL COMMENT '欠席理由',

  `remarks` text NOT NULL COMMENT '備考',
```

```
  `update_date_and_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
  '更新日時',
  PRIMARY KEY(notification_id),
  ADD KEY `student_number` (`student_number`)
)
```

外部キー制約

リスト 3.2.10　届け出情報テーブル外部キー設定 SQL

```
ADD FOREIGN KEY (`student_number`) REFERENCES `student` (`student_number`);
```

#	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	notification_id	int			いいえ	なし	届け出情報ID	AUTO_INCREMENT
2	student_number	varchar(6)	utf8mb4_0900_ai_ci		いいえ	なし	学籍番号	
3	absence_start_date	date			いいえ	なし	欠席開始日	
4	absence_end_date	date			いいえ	なし	欠席終了日	
5	absence_start_time	time			いいえ	なし	欠席開始時間	
6	absence_end_time	time			いいえ	なし	欠席終了時間	
7	reason_for_absence	text	utf8mb4_0900_ai_ci		いいえ	なし	欠席理由	
8	remarks	text	utf8mb4_0900_ai_ci		いいえ	なし	備考	
9	update_date_and_time	timestamp			いいえ	CURRENT_TIMESTAMP	更新日時	DEFAULT_GENERATED

図 3.2.8　届け出情報テーブル

（9）出席情報テーブル

リスト 3.2.11　出席情報テーブル作成 SQL

```
CREATE TABLE `attendance_information` (
  `attendance_information_id` int NOT NULL COMMENT '出席情報ID',
  `notification_id` int NOT NULL COMMENT '届け出情報ID',
  `student_number` varchar(6) NOT NULL COMMENT '学籍番号',
  `timetable_id` int NOT NULL COMMENT '時間割ID',
  `First_half_absence_flag` tinyint NOT NULL COMMENT '前半欠席フラグ',
  `Second_half_absence_flag` tinyint NOT NULL COMMENT '後半欠席フラグ',
```

書式を変更: フォント：Consolas
書式を変更: フォント：Consolas
書式を変更: フォント：Consolas
書式を変更: フォント：Consolas
書式を変更: フォント：Consolas
書式を変更: フォント：Consolas
書式を変更: フォント：Consolas

```
  `update_date_and_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '
  更新日時',
  PRIMARY KEY(attendance_information_id),
  ADD KEY `student_number` (`student_number`),
  ADD KEY `timetable_id` (`timetable_id`)
)
```

外部キー制約

リスト 3.2.12　出席情報テーブル外部キー設定 SQL

```
ADD FOREIGN KEY (`student_number`) REFERENCES `student` (`student_number`),
ADD FOREIGN KEY (`timetable_id`) REFERENCES `timetable` (`timetable_id`);
```

書式を変更: フォント：Consolas

書式を変更: フォント：Consolas

#	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	attendance_information_id	int			いいえ	なし	出席情報ID	AUTO_INCREMENT
2	notification_id	int			いいえ	なし	届け出情報ID	
3	student_number	varchar(6)	utf8mb4_0900_ai_ci		いいえ	なし	学籍番号	
4	timetable_id	int			いいえ	なし	時間割ID	
5	First_half_absence_flag	tinyint			いいえ	なし	前半欠席フラグ	
6	Second_half_absence_flag	tinyint			いいえ	なし	後半欠席フラグ	
7	update_date_and_time	timestamp			いいえ	CURRENT_TIMESTAMP	更新日時	DEFAULT_GENERATED

図 3.2.9　出席情報テーブル

3.3　ページ構成

ページ構成は表 3.3.1 の通りである.これらは，時間割システム及び電子出席簿の画面について，HTML でデザインしたファイルである．システムに必要なサーバサイドのクラス，JSP の構成は未検討のまま終わっている.

表 3.3.1　プログラム一覧

ファイル名	役割
Maketable.html	時間割作成
Shussekibo.html	出席簿表示
Table_grade1.html	1 年生時間割表示
Table_grade2.html	2 年生時間割表示

第 4 章　時間割表

4.1 時間割表の機能

時間割全体の機能として，主に教員が使う時間割編成があり（図 4.1.1）ここではコマ目情報をフォームから入力し，コマ目を作成することができる図 4.1.3)．作成したコマ目はドラック＆ドロップして時間割のセルに当てはめることができる（図 4.1.4)．また学生側の時間割表（図 4.1.2）では，作成した時間割が反映され学生は時間割を確認することができる（図 4.1.5)．

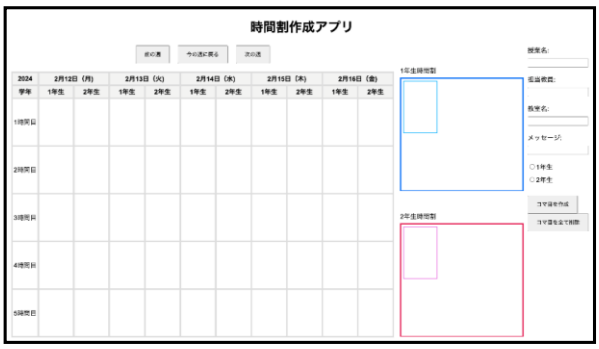

図 4.1.1　教師側の時間割作成アプリ画面

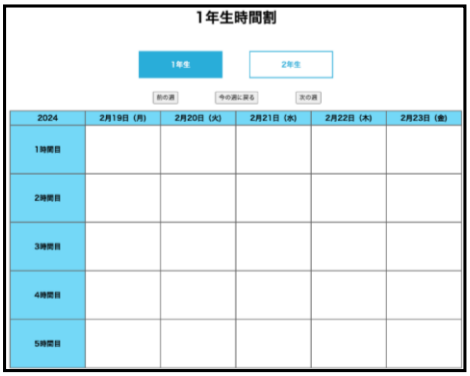

図 4.1.2　学生側の時間割表画面

（2）時間割編成の機能

図 4.1.3 の右部コマ目作成フォームとなっており，コマ目の内容を入力し作成できる．

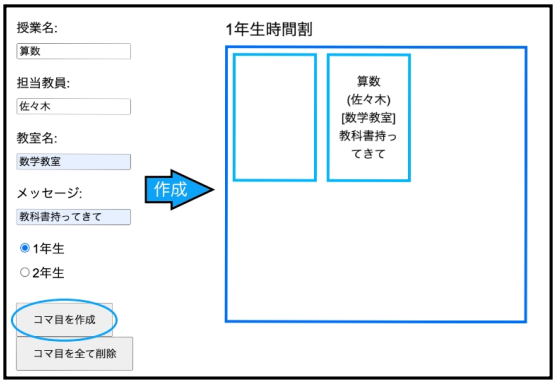

図 4.1.3　コマ目作成フォームイメージ図

図 4.1.3 の左部では作成したコマ目をドラック＆ドロップしてセルに割り当て，時間割を作成できる．

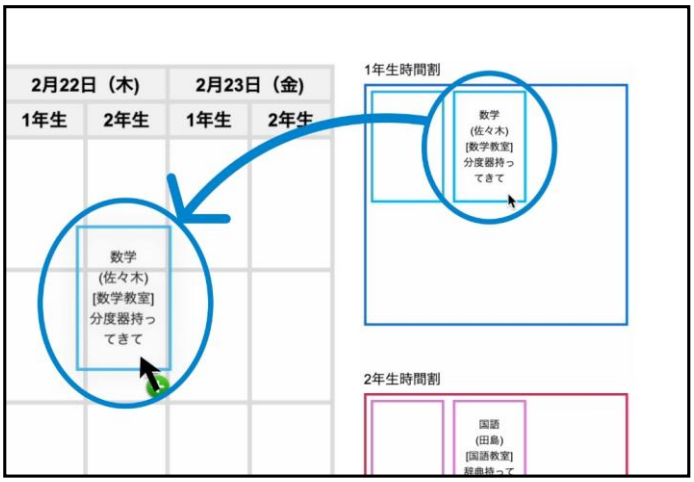

図 4.1.4　コマ目をドラック＆ドロップのイメージ図

（3）時間割表示

図 4.1.5 のとおり，作成した時間割表は学生側のページに反映される．

図4.1.5　作製した時間割表の反映イメージ

4.2 時間割作成のインタフェースについて

教師側の時間割編成能と作成した主な関数を以下に示す.

（1）時間割日付表示機能

・getWeekStartEndDates(weekNumber)：週番号から週の月曜日と金曜日の日付を取得する.

・formatDate(date)：日付表示する際のように文字にフォーマットする.

・updadeWeek()：月曜から金曜までの日付をセルに表示する.

・getDayOfWeek(dayIndex)：与えられた日付の曜日のインデックス番号で曜日文字列を返す.

0~6 がインデックス番号で日～土曜日となる.

（2）コマ目作成機能

・makeClassbox()：コマ目情報をフォームからローカルストレージに保存する.

（3）コマ目表示機能

・displaySubjectList()：保存されているコマ目情報を画面上の 1・2 年生のコマ目リストに表示する.

（4）時間割へのドラッグ&ドロップ機能

・updateIds()：コマ目を埋め込むセルにそのセルの年・日付・学年・何時間目なのかがわかる id を付与する.

（5）時間割の更新機能

・displayCellsContent()：時間割のセルにコマ目の内容を表示する.

各機能を詳しく紹介する.

(1)時間割日付表示機能

〇週の始まりと終わりの日付を計算し，図のとおり時間割表に表示させる．

図 4.2.1　時間割の日付セル

リスト 4.2.1　today 変数

```
// 日付を取得して変数 today に代入
let today = new Date();
```

・日付の取得をする．

リスト 4.2.2　currentWeek 変数

```
// 現在の週を追跡する変数
let currentWeek = 1;
```

・週を変更する際に使用する変数 currentWeek を初期化する．

リスト 4.2.3　getWeekStartEndDates(weekNumber)関数

```
// 開始日と終了日を取得する関数
function getWeekStartEndDates(weekNumber) {
  const today = new Date();
  const startOfWeek = new Date(today.getFullYear(), today.getMonth(), today.getDate()
                                        - (today.getDay()-1) + (weekNumber - 1) * 7);
  const endOfWeek = new Date(startOfWeek.getFullYear(), startOfWeek.getMonth(),
                                                        startOfWeek.getDate() + 4);
                                    // 週の終了は 5 日後（月曜から金曜まで）
  return { startOfWeek, endOfWeek };
}
```

・getWeekStartEndDates(weekNumber)関数で週の月曜日と金曜日の日付を取得する．

開始日は月曜日，終了日は開始日から 5 日後の金曜日である．

リスト 4.2.4　formatDate(date)・updateWeek()・getDayOfWeek(dayIndex)関数

```
// 日付を表示形式にフォーマットする関数
function formatDate(date) {
  const formattedDate = `${date.getMonth() + 1}月${date.getDate()}日（${getDayOfWeek(date.getDay())}）`;
  return formattedDate;  }

// 月曜から金曜までの日付を表示
function updateWeek() {
  const { startOfWeek, endOfWeek } = getWeekStartEndDates(currentWeek);
  // 年を表示
  const yearElement = document.getElementById('year');
  yearElement.textContent = startOfWeek.getFullYear();  // 年を表示
  // 各日付セルに日付を挿入
  for (let i = 0; i <= 4; i++) {
    const dayElement = document.getElementById(`day${i}`);
    const currentDate = new Date(startOfWeek);
    currentDate.setDate(startOfWeek.getDate() + i);
    // 月曜から金曜日までの日付を表示
    dayElement.textContent = formatDate(currentDate); }}
// 曜日を取得する関数
function getDayOfWeek(dayIndex) {
  const daysOfWeek = ["日", "月", "火", "水", "木", "金", "土"];
  return daysOfWeek[dayIndex]; }
```

・formatDate(date)・updateWeek()・getDayOfWeek(dayIndex)関数で日付をセルに表示する.

・「前の週」「今の週に戻る」「次の週」ボタンで表示する日付を週ごとで切り替える.

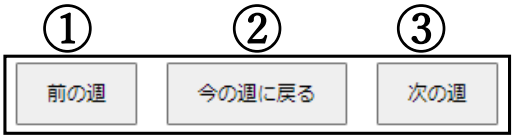

図 4.2.2 週変更ボタン

リスト 4.2.5 「前の週」「今の週に戻る」「次の週」ボタンのイベントリスナー

```
// "前の週" ボタンのためのイベントリスナー
document.getElementById('prevWeekButton').addEventListener('click', function() {
  currentWeek--;
  updateWeek();
});

// "今の週に戻る" ボタンのためのイベントリスナー
document.getElementById('currentWeekButton').addEventListener('click', function() {
  currentWeek = 1;
  updateWeek();
});

// "次の週" ボタンのためのイベントリスナー
document.getElementById('nextWeekButton').addEventListener('click', function() {
  currentWeek++;
  updateWeek();
});
```

①
②
③

前の週 id=prevWeekButton : currentWeek 変数をデクリメントして現在の週を 1 週前に変更し，その後に updateWeek()関数を呼び出して週の表示を更新する.

書式変更: インデント：最初の行： 1 字

今の週に戻る id=currentWeekButton： currentWeek 変数を 1 に設定して現在の週を最初の週に戻し，その後に updateWeek()関数を呼び出して週の表示を更新する.

次の週 id=nextWeekButton：currentWeek 変数をインクリメントして現在の週を 1 週後に変更し，その後に updateWeek()関数を呼び出して週の表示を更新する.

（2）コマ目作成機能

ユーザーが入力した授業名，担当教員，教室名，メッセージ，学年をもとに，新しいクラスボックスを作成することができる．

書式変更: インデント：左： 0 mm, 最初の行： 1 字

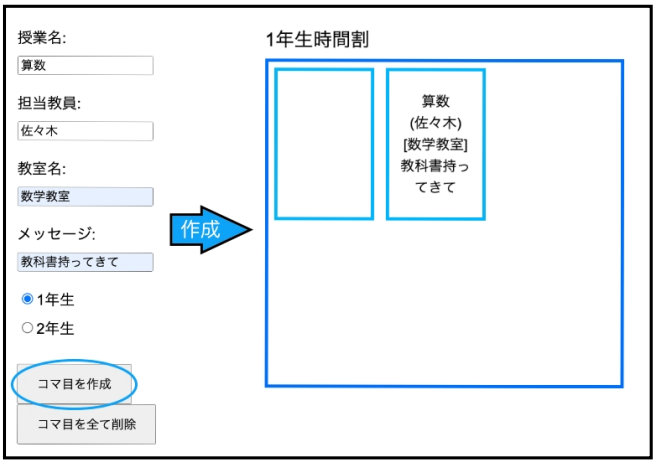

図 4.2.3　コマ目作成イメージ

機能の大まかな手順は次の通りである．

・フォームから入力されたデータを取得する．

・コマ目をローカルストレージに保存するためのキーを設定する．

書式変更: インデント：左 0 字

キーには，現在のタイムスタンプを使用する．

・クラスボックスの情報を JavaScript オブジェクトとして保存する．

書式変更: インデント：左 0 字

・クラスボックスの情報を JSON 形式に変換して，キーと一緒に，ローカルストレージに保存する．

（3）コマ目表示機能

○コマ目を作成し，ローカルストレージに保存したコマ目を画面上に表示する機能である．

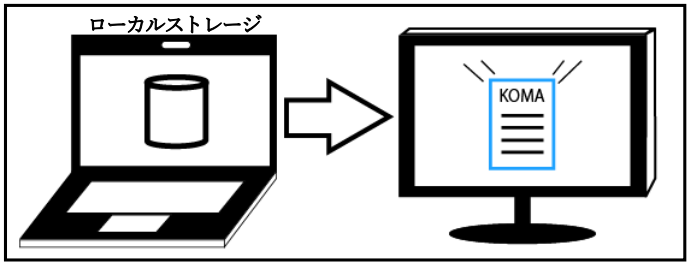

図 4.2.4 ローカルストレージからコマ目をディスプレイ表示イメージ．

機能の大まかな手順は次の通りである．

・Id=subjectList1, subjectList2 の要素を受け取る．

・ローカルストレージに保存したデータを for ですべて取得する．

書式を変更: 蛍光ペン

・取得したデータは JSON 形式で保存されているので JSON.parse()で戻す．

・コマ目を表示するための div 要素を作成し，そこに情報を埋め込みます．

・各コマ目に draggable 属性を追加します．

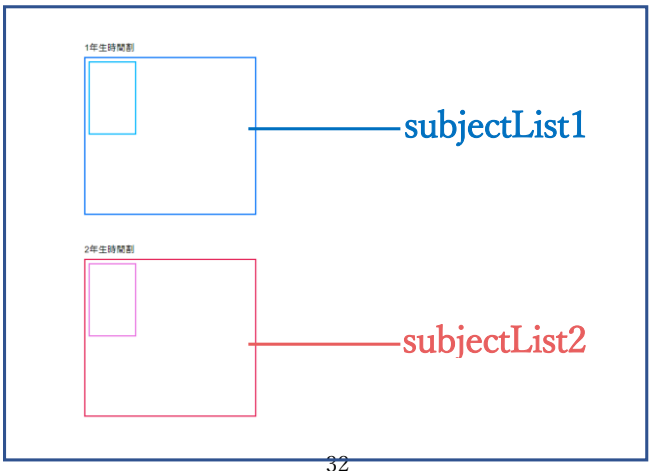

図 4.2.5　学年ごとのコマ目リスト

リスト 4.2.7　関数：displaySubjectList ()

```
// subjectList1&2 にローカルストレージ保存してあるコマ目を表示する関数
function displaySubjectList() {
  const subjectList1 = document.getElementById('subjectList1');
  const subjectList2 = document.getElementById('subjectList2');
  // ローカルストレージから保存されているデータを取得
  for (var i = 0; i < localStorage.length; i++) {
      let key = localStorage.key(i); //ローカルストレージのキーを取得
      let data = localStorage.getItem(key); // キーに対応するデータを取得
      let classBoxInfo = JSON.parse(data); // データを JavaScript オブジェクトに変換
      // コマ目の要素を作成し，subjectList に挿入する
      let subjectBox = document.createElement('div');
      subjectBox.classList.add('subjectbox' + classBoxInfo.grade);
      subjectBox.draggable = true;  //dorakku 属性を追加
      //コマ目の情報を設定する
      subjectBox.innerHTML = classBoxInfo.subjectName + '<br>' +
      '(' + classBoxInfo.teacher + ')<br>' +'[' + classBoxInfo.room + ']<br>'
+classBoxInfo.message;
      // radio ボタンで決めた学年の subjectList に挿入する
      if (classBoxInfo.grade === '1') {
          subjectList1.appendChild(subjectBox);
      } else if (classBoxInfo.grade === '2') {
```

・保存されているコマ目情報をディスプレイに表示する．displaySubjectList()関数

（4）時間割へのドラッグ&ドロップ機能

コマ目をドラックして時間割のセルにドロップできる機能である．

書式変更: インデント：最初の行：１字

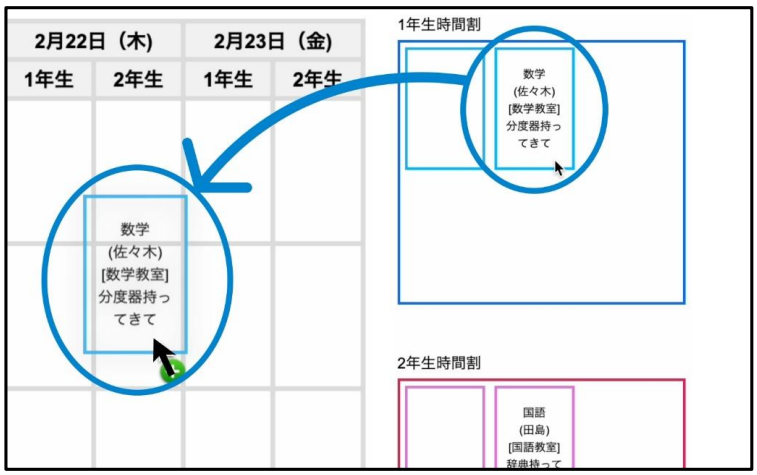

図 4.2.6 コマ目のドラック＆ドロップイメージ．

ドラック＆ドロップをした際の大まかな動きは次のとおりである．

・コマ目をドロップするセルにそのセル固有の id を付与する．

例)　2024 年 1 月 2 日 3 年生の 4 時間目のセルなら id=“2024_1_2_3_4”となる．

20XX年	○月△日（月）		○月△日（火）		○月△日（水）		○月△日（木）		○月△日（金）	
学年	1年生	2年生	1年生	2年生	1年生	2年生	1年生	2年生	1年生	2年生
1時間目	20xx_○_△_1_1	20xx_○_△_2_1	20xx_○_△_1_1	20xx_○_△_2_1	20xx_○_△_1_1	20xx_○_△_2_1	20xx_○_△_1_1	20xx_○_△_2_1	20xx_○_△_1_1	20xx_○_△_2_1
2時間目	20xx_○_△_1_2	20xx_○_△_2_2	20xx_○_△_1_2	20xx_○_△_2_2	20xx_○_△_1_2	20xx_○_△_2_2	20xx_○_△_1_2	20xx_○_△_2_2	20xx_○_△_1_2	20xx_○_△_2_2
3時間目	20xx_○_△_1_3	20xx_○_△_2_3	20xx_○_△_1_3	20xx_○_△_2_3	20xx_○_△_1_3	20xx_○_△_2_3	20xx_○_△_1_3	20xx_○_△_2_3	20xx_○_△_1_3	20xx_○_△_2_3
4時間目	20xx_○_△_1_4	20xx_○_△_2_4	20xx_○_△_1_4	20xx_○_△_2_4	20xx_○_△_1_4	20xx_○_△_2_4	20xx_○_△_1_4	20xx_○_△_2_4	20xx_○_△_1_4	20xx_○_△_2_4
5時間目	20xx_○_△_1_5	20xx_○_△_2_5	20xx_○_△_1_5	20xx_○_△_2_5	20xx_○_△_1_5	20xx_○_△_2_5	20xx_○_△_1_5	20xx_○_△_2_5	20xx_○_△_1_5	20xx_○_△_2_5

図 4.2.7　セルと対応する id のイメージ

・コマ目をドラックされたら dragstart イベントが発生する．

dragstart イベントでは，setDate()メソッドを使用して，ドラックされた要素のデータをデータ転送オブジェクトに設定しドロップされる準備をする．

・コマ目を特定のセルにドロップされたら drop イベントが発生する．

drop イベントでは，ドロップされたセルに id とコマ目の内部データを取得し，
セルの id をキーとしてローカルストレージに保存する．

リスト 4.2.8　updateIds 関数.

```
// ID を更新する関数
function updateIds() {
  const cells = document.querySelectorAll('td.komacell');
  for (let i = 0; i < cells.length; i++) {
    // 日付計算のためのロジックをここに追加
    let digit = i % 10;
    let result;
    switch (true) {
        case (digit === 0 || digit === 1):
            result = 0;
            break;
        case (digit === 2 || digit === 3):
            result = 1;
            break;
        case (digit === 4 || digit === 5):
            result = 2;
            break;
        case (digit === 6 || digit === 7):
            result = 3;
            break;
        case (digit === 8 || digit === 9):
            result = 4;
            break;
        default:
            result = -1; // デフォルトの場合、-1 を返すなど、エラー処理を行う場合
            break;
    }

    const { startOfWeek, endOfWeek } = getWeekStartEndDates(currentWeek);
    const currentDate = new Date(startOfWeek);

    let date = new Date(currentDate);
    date.setDate(currentDate.getDate()+result); //週の各日に対応
    const year = date.getFullYear();
```

```
      const month = date.getMonth() + 1; //月を1ベースで取得
      const day = date.getDate();

      //何年生のセルか？
      let grade = i % 2 === 1 ? 2 : 1;

      //何時間目のセルか？
      //i=0~49 なので i を 10 で割ったものに 1 を足すと 1~5 時間目になる。
      let row = Math.floor(i/10)+1;



      const cellId = `${year}_${month}_${day}_${grade}_${row}`;
      cells[i].setAttribute('id', cellId);
    }
  }
```

リスト 4.2.9　dragstart イベント

```
//コマ目をドラック
document.addEventListener('DOMContentLoaded', () => {
  // subjectList1 と subjectList2 の親要素にイベントリスナーを設定
  document.querySelectorAll('.KomaList-container').forEach(container => {
    container.addEventListener('dragstart', event => {
        const target = event.target;
        console.log('ドラッグされたコマ目の情報:', target);
        // ドラッグ開始時にデータを設定
        event.dataTransfer.setData('text',
            JSON.stringify({ content: target.innerHTML }));
      }
    );
  });
});
```

・dragstart イベントとは，ドラックされたときに実行されるイベントである．

リスト 4.2.10　drop イベント．

```
//コマ目を cell にドロップ
document.querySelectorAll('.komacell').forEach(cell => {
  cell.addEventListener('drop', event => {
      //デフォルトのイベントの動作をキャンセルする
      //（そうすることによって，独自との処理を施せる)
       event.preventDefault();
      const cellId = cell.id;
      // ドロップされたコマ目の情報を取得
      const boxInfo = event.dataTransfer.getData('text');

      //cell にコマ目を表示　※(5)で解説する．
      displayCellsContent()
  });

  cell.addEventListener('dragover', event => {
      event.preventDefault();
  });
});
```

・drop イベントは，コマ目を特定のセルにドロップしたら実行される．

（5）時間割の更新機能

時間割表の各セルにコマ目の情報を表示するための機能である.

書式変更: インデント：最初の行： 1 字

時間割を更新する流れは次のとおりである.

・querySelectorAll('.komacell')を使用して，時間割表のすべてのセルを取得

・各セルに対してループを実行し，それぞれのセルに対応するコマ目情報を取得

・localStorage.getItem(cellId)を使用して，セルの id に関連付けられたコマ目情報を取得

・取得したデータがある場合は，JSON.parse()を使用してセルに innerHTML でコマ目情報をセルに表示する.

リスト 4.2.11 displayCellsContent( )関数.

```
//cell にコマ目を表示
function displayCellsContent() {
  const cells = document.querySelectorAll('.komacell');
  cells.forEach(cell => {
    const cellId = cell.id;
    const storedData = localStorage.getItem(cellId);
    if (storedData) {
      const dataObject = JSON.parse(storedData); // データが JSON 形式の場合
      cell.innerHTML = dataObject.content; // content プロパティに保存された内容を設定
    } else{
      cell.innerHTML = ""; //その cell の id に何もセットされてなかったら空白をいれる
    }
```

・displayCellsContent()関数で，セルにコマ目情報を表示する.

4.3 学生側の時間割インタフェースについて

学生側では時間割表を確認することができる.

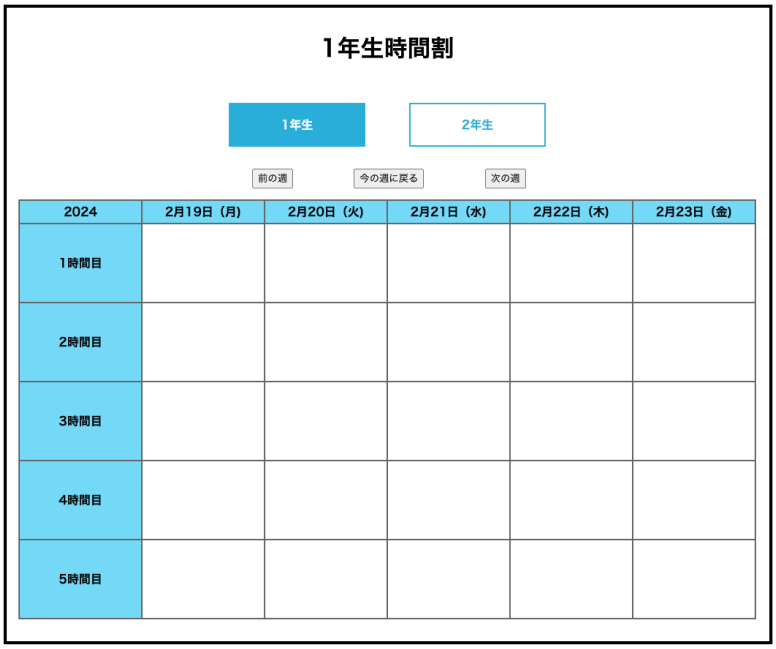

図 4.3.1　学生側の時間割

主な機能は次のとおりである.

（1）「前の週」「今の週に戻る」「次の週」ボタンで表示する日付を週ごとで切り替える

※1.1.2 の（1）と同様であるためコードは省略する

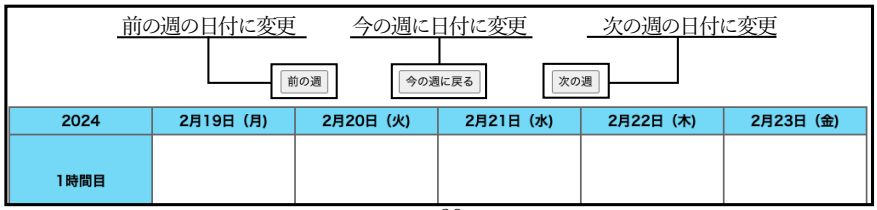

図 4.3.2　学生側時間割表での週変更ボタン

（2）教師側で作成した時間割を共有する.

時間割表のセルに固有の id を,

1.1.2 の（4）同様に付与する.

1 年生の時間割の場合

20XX年	○月△日（火）	○月△日（火）	○月△日（水）	○月△日（木）	○月△日（火）
学年	1 年生	1 年生	1 年生	1 年生	1 年生
1 時間目	20xx_○_△_1_1	20xx_○_△_1_1	20xx_○_△_1_1	20xx_○_△_1_1	20xx_○_△_1_1
2 時間目	20xx_○_△_1_2	20xx_○_△_1_2	20xx_○_△_1_2	20xx_○_△_1_2	20xx_○_△_1_2
3 時間目	20xx_○_△_1_3	20xx_○_△_1_3	20xx_○_△_1_3	20xx_○_△_1_3	20xx_○_△_1_3
4 時間目	20xx_○_△_1_4	20xx_○_△_1_4	20xx_○_△_1_4	20xx_○_△_1_4	20xx_○_△_1_4

2 年生の時間割の場合

20XX年	○月△日（火）	○月△日（火）	○月△日（水）	○月△日（木）	○月△日（火）
学年	2年生	2年生	2年生	2年生	2年生
1 時間目	20xx_○_△_2_1	20xx_○_△_2_1	20xx_○_△_2_1	20xx_○_△_2_1	20xx_○_△_2_1
2 時間目	20xx_○_△_2_2	20xx_○_△_2_2	20xx_○_△_2_2	20xx_○_△_2_2	20xx_○_△_2_2
3 時間目	20xx_○_△_2_3	20xx_○_△_2_3	20xx_○_△_2_3	20xx_○_△_2_3	20xx_○_△_2_3
4 時間目	20xx_○_△_2_4	20xx_○_△_2_4	20xx_○_△_2_4	20xx_○_△_2_4	20xx_○_△_2_4

図 4.3.3　セルと対応する id のイメージ

今回は 1 年生の時間割と 2 年生の時間割が分かれているのでそれに対応するように Id を付与する.

リスト 4.3.1　updateIds( )関数.

```
// ID を更新する関数
function updateIds() {
  const cells = document.querySelectorAll('td.komacell');
  for (let i = 0; i < cells.length; i++) {
    // 日付計算のためのロジックをここに追加
    let result = i % 5 ;
    const { startOfWeek, endOfWeek } = getWeekStartEndDates(currentWeek);
    const currentDate = new Date(startOfWeek);

    let date = new Date(currentDate);
    date.setDate(currentDate.getDate()+result); //週の各日に対応
    const year = date.getFullYear();
    const month = date.getMonth() + 1; //月を 1 ベースで取得
    const day = date.getDate();
```

```
    //何年生のセルか？
    let grade = 1;  //2 年生の時は grade = 2;

    //何時間目のセルか？
    //i=0~49 なので i を 10 で割ったものに 1 を足すと 1~5 時間目になる.
    let row = Math.floor(i/5)+1;


    const cellId = `${year}_${month}_${day}_${grade}_${row}`;
    cells[i].setAttribute('id', cellId);
  }
}
```

1.1.2 （5）と同様に各セルに保存されたコマ目の内容を表示する.

document.querySelectorAll('.komacell'）を使って，全てのセルを取得する.

各セルに対して，以下の処理を行う.

・セルの id を取得する.

・ローカルストレージからそのセルの id に対応するデータを取得する.

・取得したコマ目の内容をセルの innerHTML プロパティに設定する.

・もしそのセルに対応するデータ（コマ目）が存在しない時は，セルの内容を空に設定する.

このようにして時間割のセルにコマ目の内容を表示する.

リスト 4.3.2 displayCellsContent( )関数.

```
//cell にコマ目を表示
function displayCellsContent() {
  const cells = document.querySelectorAll('.komacell');
  cells.forEach(cell => {
    const cellId = cell.id;
    const storedData = localStorage.getItem(cellId);
    if (storedData) {
      const dataObject = JSON.parse(storedData); // データが JSON 形式の場合

       // content プロパティに保存された内容を設定
      cell.innerHTML = dataObject.content;
    } else{
       //その cell の id に何もセットされてなかったら空白をいれる
      cell.innerHTML = "";
    }
  });
}
```

第５章　　電子出席簿について

5.1　基本動作

まず，最初は，　ログインページでユーザ名とパスワードを入力する．

図 3.1.1　ログインページ入力例

出欠確認ページでは図 3.1.2 が最初に表示され，　欠席届が出ている学生がいる場合は欠席開始時間，欠席終了時間，　欠席理由が表示される．

図 3.1.2　出欠確認ページ

次に，出欠確認ページの出欠データ入力について説明する．図3.1.3では，学生１の場合は，一日休みなのでオールチェックをクリックしてすべてのコマを欠席に切り替えている．学生２の場合は，１コマ目と２コマ目の約半分を欠席しているので，図のようなデータになる．学生４の場合は，欠席届が出ていないので欠席開始時間，欠席終了時間，欠席理由が入力されていない．手動で欠席開始時間と欠席終了時間を入力し，１コマ目の半分を欠席としている．データの入力が完了したら画面下の「データの保存・送信」ボタンを押すことでデータが保存される．

図3.1.3　出欠確認ページ入力例

5.2　セキュリティ

図 3.2.1 はログインページである.

教員以外のユーザーが電子出席簿を操作できないようにするために,　ログインページを実装した.

ユーザー名とパスワードは用意したものを使用する.

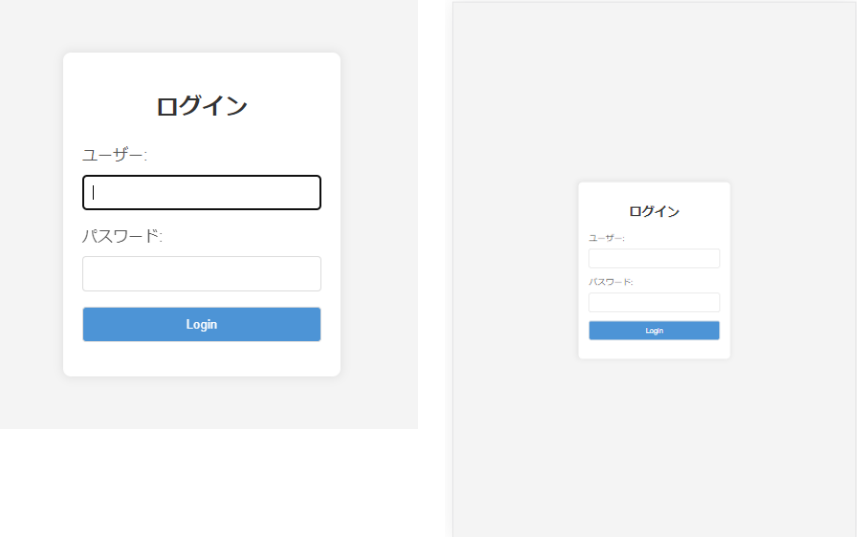

図 3.2.1　ログインページ(左)　レスポンシブ対応(右)

次にプログラムの説明をする.

リスト 3.2-1 Login.html

```
(省略)
<body>
    <form action="TopPage.html" method="post">
        <h2>ログイン</h2>
        <label for="username">ユーザー:</label>
        <input type="text" id="username" name="username" required>

        <label for="password">パスワード:</label>
        <input type="password" id="password" name="password" required>

        <input type="submit" value="Login">
    </form>
</body>
(省略)
```

Login.html では本来であれば，入力されたユーザー名とパスワードをデータベースにあるものと照合し， 合っていれば次のページに進むという仕様だった．だが， データベースとの結合が間に合わなかったため今回はデータを入力しただけで進むようになっている.

5.3　出欠確認ページについて

コマ名，欠席開始時間，欠席終了時間，欠席理由はデータベースを参照しページを開いたときに表示される．

図 3.3-1　出席確認ページ上部(左)　レスポンシブ対応(右)

欠席開始時間，欠席終了時間，欠席理由がデータベースを参照して表示されていて，コマ１，コマ２，となっている部分はその日付の授業名をデータベースから参照し表示している．

画面上部の日付はページを開いた日付となっていて，この日付をクリックするとカレンダーが表示されてそこから別の日付を選択することができる．コマ１，コマ２，となっている部分はその日付の授業名をデータベースから参照し表示する．

学生の出欠席は出欠席ボタンを押すことで出席状況を入力し，実際の出欠状況が欠席届と違う場合は欠席開始時間と欠席終了時間を変更し，メモ等に補足を書き込むなどをしてからデータを保存する．

本ページで使用する関数は次のとおりである.

① generateAttendanceTable() ：テーブルを生成する関数

② toggleAttendance() ：出欠ボタンをトグルする関数

③ saveAttendanceData() ：出欠データを保存する関数

④ fetchReasonDataFromDatabase() ：学生の欠席理由をデータベースから取得する関数

⑤ generateAttendanceTableForDate() ：特定の日付の出欠テーブルを生成する関数

⑥ updateCurrentDate() ：表示されている日付を更新する関数

⑦ goToToday() ：今日の日付に切り替える関数

⑧ toggleCheckmark() ：オールチェックボックスをトグルする関数

⑨ initFlatpickr() ：日付選択 Flatpickr を初期化する関数

⑩ formatDate() ：日付を指定された形式具体的な形式でフォーマットする関数

⑪ saveAndSubmitData() ：データを保存して送信する関数

次にプログラムの説明をする.

リスト 3.3-1 Shussekibo.html

```
                                    (省略)
  <script>
        const numClasses = 5;


        const database = [
            { id: 1,  name: "学生１", absenceStartTime: "08:50",  absenceEndTime:
"16:10"  },
            { id: 2,  name: "学生２"},
            { id: 3,  name: "学生３"},
            { id: 4,  name: "学生４"},
            { id: 5,  name: "学生５"},
            { id: 7,  name: "："},
            { id: 8,  name: "学生１８"},
            { id: 9,  name: "学生１９"},
            { id: 10,  name: "学生２０"},
        ];
  let currentDateElement = document.getElementById("currentDateText");
```

リスト 3.3-2 function generateAttendanceTable()

```
function generateAttendanceTable() {
            console.log("Generating attendance table");
            const table = document.getElementById("attendanceTable");
            table.innerHTML = "";

            const headerRow = document.createElement("tr");
            const nameHeaderCell = document.createElement("th");
            nameHeaderCell.textContent = "名前";
            headerRow.appendChild(nameHeaderCell);

            const checkmarkHeaderCell = document.createElement("th");
            checkmarkHeaderCell.textContent = "オールチェック";
            headerRow.appendChild(checkmarkHeaderCell);

            for (let i = 0; i < numClasses; i++) {
                const cell = document.createElement("th");
                cell.textContent = "コマ " + (i + 1);
                headerRow.appendChild(cell);
            }

            const startTimeHeaderCell = document.createElement("th");
```

①

②

③

④

```
startTimeHeaderCell.textContent = "欠席開始時間";
headerRow.appendChild(startTimeHeaderCell);

const endTimeHeaderCell = document.createElement("th");
endTimeHeaderCell.textContent = "欠席終了時間";
headerRow.appendChild(endTimeHeaderCell);

const reasonHeaderCell = document.createElement("th");
reasonHeaderCell.textContent = "欠席理由";
headerRow.appendChild(reasonHeaderCell);

const memoHeaderCell = document.createElement("th");
memoHeaderCell.textContent = "メモ";
headerRow.appendChild(memoHeaderCell);

table.appendChild(headerRow);

for (let k = 0; k < database.length; k++) {
    const row = document.createElement("tr");
    const nameCell = document.createElement("td");
    nameCell.classList.add("name-cell");
    const nameInput = document.createElement("div");
    nameInput.classList.add("name-input");
    nameInput.textContent = database[k].name;
```

④

⑤

⑥

```
nameCell.appendChild(nameInput);
row.appendChild(nameCell);


const checkmarkCell = document.createElement("td");
checkmarkCell.classList.add("checkmark-cell");
checkmarkCell.innerHTML = '<i class="fas fa-check"></i>';
table.appendChild(row);


checkmarkCell.onclick = function () {
    toggleCheckmark(this);
};


row.appendChild(checkmarkCell);
for (let i = 0; i < numClasses; i++) {
    const cellContainer = document.createElement("td");
    cellContainer.classList.add("button-container");
    const upperButton = document.createElement("button");
    upperButton.classList.add("attendance-button");
    upperButton.addEventListener("click",  function () {
        toggleAttendance(this);
    });
    upperButton.textContent = "出席";


    const lowerButton = document.createElement("button");
```

⑥

⑦

⑧

```
                lowerButton.classList.add("attendance-button");
                lowerButton.addEventListener("click",  function () {
                    toggleAttendance(this);
                });
                lowerButton.textContent = "出席";                    ⑧

                cellContainer.appendChild(upperButton);
                cellContainer.appendChild(lowerButton);
                row.appendChild(cellContainer);
            }

            // 欠席開始時間
        const startTimeCell = document.createElement("td");
        startTimeCell.classList.add("time-cell");
        const startTimeInput = document.createElement("input");
        startTimeInput.classList.add("time-input");
        startTimeInput.type = "time";                                ⑨
        startTimeInput.step = "300";
        startTimeInput.value = database[k].absenceStartTime; // デフォルト値を
セット
        startTimeCell.appendChild(startTimeInput);
        row.appendChild(startTimeCell);

        // 欠席終了時間                                              ⑩
```

```
            endTimeCell.classList.add("time-cell");
            const endTimeInput = document.createElement("input");
            endTimeInput.classList.add("time-input");
            endTimeInput.type = "time";
            endTimeInput.step = "300";
            endTimeInput.value = database[k].absenceEndTime; // デフォルト値をセッ
ト

            endTimeCell.appendChild(endTimeInput);
            row.appendChild(endTimeCell);


            const reasonCell = document.createElement("td");
            reasonCell.classList.add("reason-cell");
            const reasonInput = document.createElement("div");
            reasonInput.classList.add("reason-input");
            const sampleReasonData = fetchReasonDataFromDatabase(database[k].id);
            reasonInput.textContent = sampleReasonData;
            reasonCell.appendChild(reasonInput);
            row.appendChild(reasonCell);


            const memoCell = document.createElement("td");
            memoCell.classList.add("memo-cell");
            const memoInput = document.createElement("textarea");
            memoInput.classList.add("memo-input");
            memoCell.appendChild(memoInput);
```

⑩

⑪

⑫

```
            table.appendChild(row);                                    ⑫
        }
    }
```

① table は HTML で定義されたテーブル要素を取得している． その後， innerHTML をクリアすることで既存のテーブル内容を削除している．

② テーブルのヘッダ行を生成している． 各列のヘッダセル（名前とオールチェック）を作成し，ヘッダ行に追加している．

③ 各授業コマごとのヘッダセルを生成している， numClasses(クラスの人数) 回ループし，各コマのラベルを持つセルを作成し， ヘッダ行に追加している．

④ 欠席開始時間， 欠席終了時間， 欠席理由， メモの各列のヘッダセルを生成し， ヘッダ行に追加している． 最終的に， ヘッダ行全体をテーブルに追加している．

⑤ データベース内の各学生に対して行を生成するループ， 各学生ごとに新しい行（tr 要素）を作成する．

⑥ 各学生の名前を表示するセルを生成している． name-cell および name-input クラスを追加し， 名前を表示するための div 要素を作成しセルに追加している．

⑦ オールチェックボックスのセルを生成している． checkmark-cell クラスを追加し，<i> タグを用いて FontAwesome のアイコンをセルに追加している． また， クリックイベントリスナーを設定している． （※FontAwesome とは様々なアイコンをフォントファイル形式にして公開しているアイコン集のこと）

⑧ 各コマごとのボタンを生成している． 各ボタンは attendance-button クラスを持ち， クリック時に toggleAttendance 関数が呼び出される．

⑨ 欠席開始時間のセルを生成している． time-cell および time-input クラスを追加し，<input> 要素を用いて時刻を入力するための欠席開始時間フィールドを作成している．

⑩ 同様に欠席終了時間のセルを生成している．

⑪ 欠席理由のセルを生成している． `reason-cell` および `reason-input` クラスを追加し，`<div>` 要素を用いて欠席理由を表示するための欠席理由フィールドを作成している．

⑫ メモのセルを生成している． `memo-cell` クラスを追加し， `<textarea>` 要素を用いて複数行のメモを入力するためのフィールドを作成している． 各セルは行に追加され， 最終的にテーブルに追加される．

リスト 3.3-2 toggleAttendance

```
  function toggleAttendance(button) {
            const studentId = button.dataset.studentId;
            const classIndex = button.dataset.classIndex;
            const attendanceStatus = button.textContent === "出席" ? true :
false;

            saveAttendanceData(studentId, classIndex, attendanceStatus);

            if (attendanceStatus) {
                button.textContent = "欠席";
                button.classList.add("absent");
            } else {
                button.textContent = "出席";
                button.classList.remove("absent");
            }
        }
```

出欠ボタンをトグル(同一の操作を行うことで２つの状態を切り替えること)する関数.　出欠ボタンがクリックされたときに呼び出され,　出席と欠席を切り替える.

リスト 3.3-3saveAttendanceData

```
function saveAttendanceData(studentId, classIndex, attendanceStatus) {
            // ここにデータベースへの保存処理を追加

            console.log(`生徒 ID: ${studentId}, コマ: ${classIndex}, 出席:
${attendanceStatus ? 'true' : 'false'} のデータを保存しました.`);
        }
```

出欠データを保存する関数. 出欠が変更されたときに呼び出され, データベースに出欠情報を保存する（実際のデータベースへの保存は未実装).

リスト 3.3-4fetchReasonDataFromDatabase(studentId)

```
function fetchReasonDataFromDatabase(studentId) {
            const reasons = [
                "体調不良",
                "就活",
            ];

            if (studentId <= reasons.length) {
                return reasons[studentId - 1];
            }

            return "欠席理由なし";
        }
```

学生の欠席理由をデータベースから取得する関数. 各学生の欠席理由を表示する.

リスト 3.3-5generateAttendanceTableForDate(date)

```
function generateAttendanceTableForDate(date) {
            console.log("Generating table for date:",  date);
            generateAttendanceTable();
            updateCurrentDate(date);
        }
```

特定の日付の出欠テーブルを生成する関数（未実装）．日付が切り替わったときに呼び出され，指定された日付の出欠テーブルを生成する（実際の日付切り替えは未実装）．

リスト 3.3-6updateCurrentDate(date)

```
function updateCurrentDate(date) {
            const options = { year: "numeric", month: "numeric", day:
"numeric", weekday: "long" };
            const formattedDate = date.toLocaleDateString("ja-JP", options);
            currentDateElement.textContent = formattedDate;
        }
```

表示されている日付を更新する関数．日付が切り替わったときに呼び出され，画面上に表示されている日付を指定された日付に更新する．

リスト 3.3-7goToToday()

```
function goToToday() {
            const currentDate = new Date();
            console.log("Current Date:",  currentDate);

            // Flatpickr の dateFormat を変更する
            const flatpickrInstance = flatpickr(currentDateElement,  {
                dateFormat: "Y-m-d¥¥(D)",  // 例: "2024-01-23(Wed)"
                onClose: function (selectedDates,  dateStr,  instance) {
                    const selectedDate = selectedDates[0];
                    if (selectedDate) {
                        generateAttendanceTableForDate(selectedDate);
                        currentDateElement.textContent                    =
formatDate(selectedDate);
                    }
                },
                clickOpens: false,
                defaultDate: new Date(),
            });

            // Flatpickr の toggle メソッドを使用してカレンダーを開く
            flatpickrInstance.toggle();
        }
```

今日の日付に切り替える関数（未実装）．　今日の日付を表示する．

リスト 3.3-8toggleCheckmark(cell)

```
function toggleCheckmark(cell) {
            const icon = cell.querySelector('i');
            const                    attendanceButtons                    =
cell.parentElement.querySelectorAll(".attendance-button");
            if (icon.style.display === 'inline') {
                attendanceButtons.forEach(button => {
                    button.textContent = "出席";
                    button.classList.remove("absent");
                });
                icon.style.display = 'none';
            } else {
                attendanceButtons.forEach(button => {
                    button.textContent = "欠席";
                    button.classList.add("absent");
                });
                icon.style.display = 'inline';
            }
        }
```

オールチェックボックスをトグルする関数. オールチェックボックスがクリックされたときに呼び出され, 全ての出欠ボタンを一括で切り替える.

リスト 3.3-9initFlatpickr()

```
 function initFlatpickr() {
            const                    currentDateElement                    =
document.getElementById("currentDateText");
            // カレンダーを手動で開くためのインスタンスを作成
            const flatpickrInstance = flatpickr(currentDateElement,  {
                dateFormat: "Y-m-d¥¥(D)",  // `%D` ではなく `D` を使って曜日を表
示
                onClose: function (selectedDates,  dateStr,  instance) {
                    // カレンダーが閉じられた時に呼ばれるコールバック
                    const selectedDate = selectedDates[0];
                    if (selectedDate) {
                        generateAttendanceTableForDate(selectedDate);
                        // 以下の 2 行を追加してカレンダーが閉じたらカレンダーの日付に
戻らないようにする
                        instance.setDate(selectedDate);
                        currentDateElement.textContent = formatDate(selectedDate);
                    }
                },
                clickOpens: false,  // デフォルトのカレンダー表示を無効にする
                defaultDate: new Date(),  // 初期表示日を今日に設定
            });

            // 日付テキストをクリックした時に手動でカレンダーを表示
```

```
            flatpickrInstance.toggle();
        });

        // カレンダーの日付をクリックしても閉じないようにする
        flatpickrInstance.calendarContainer.addEventListener("click", function (e) {
            e.stopPropagation();
        });
    }
```

Flatpickr を初期化する関数。Flatpickr を使用して日付を選択するための初期設定を行う.

リスト 3.3-10formatDate(date)

```
function formatDate(date) {
    const options = { year: "numeric", month: "numeric", day: "numeric",
weekday: "long" };
    return date.toLocaleDateString("ja-JP", options);
}
```

日付を指定された形式でフォーマットする関数。日付の表示形式を整える.

リスト 3.3-11saveAndSubmitData()

```
function saveAndSubmitData() {
            // ここにデータを保存して送信する処理を追加

            // 例: データをコンソールに表示
            console.log("データを保存して送信しました. ");

            // 保存成功メッセージを表示
            alert("データを保存しました");

            // TopPage.html にリダイレクト
            window.location.href = 'TopPage.html';
        }
```

データを保存して送信する関数（未実装）. 出欠データを保存し, 必要に応じてサーバにデータを送信する.

第６章　　バックエンドシステムの実装

バックエンド（サーバサイド）のシステムについては，本格的な設計の前にデータベースとの連携を確認するために，３.２で述べた学生テーブルと時間割テーブルを参照し，図 6.1.1 の赤枠に，学生の名前と科目名を表示する機能を実装した.

Java を用いた Web ページで実装できた部分は以下のとおりである.

- 時間割，電子出席簿ページの表示
- JDBC によるデータベースの接続
- 電子出席簿での時間割テーブルを参照した科目情報の取得
- 電子出席簿での学生テーブルを参照した学生情報の取得

そして，実装したクラスは表 6.1.1 のとおりである.

表 6.1.1　各クラスの説明

ファイル名	内容
Data.java	各クラスで使用する JavaBeans
AttendanceLogic.java	電子出席簿のサーブレット
SearchData.java	データ検索
SearchDataDAO.java	データ検索の DAO
Attendance.jsp	電子出席簿のページ

図 6.1.1 は，各ファイル間のデータの遷移図である．
AttendanceLogic クラスから SearchData クラスの searchAll メソッドを参照し，SearchDataDAO クラスで学生テーブルから学生の氏名を，科目テーブルと時間割テーブルを結合させ，指定した日時の科目名を選択する．SearchData クラスに氏名と科目名を返し，セッションに保存する．
Attendance.jsp では，セッションから氏名と科目名を取得し，結果を表示する．

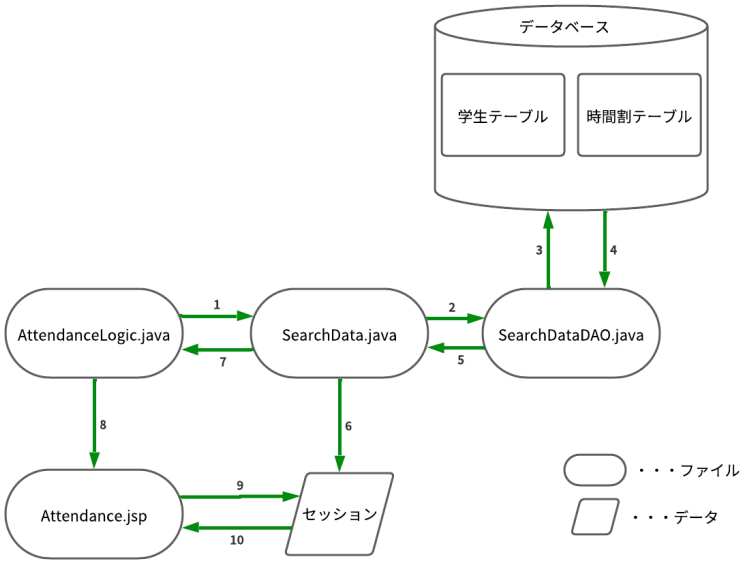

図 6.1.1　データの遷移図

（1） Data クラス

データベースから参照した，学籍番号，氏名・・・を表すモデルである．各フィールドは Setter メソッド，Getter メソッドを定義している．

リスト 6.1.1 Data.java

```
import java.io.Serializable;


public class Data implements Serializable {

	private String student_number;

	private String last;

	private String first;

	private int number;

	private String subject;

	private String subject_Name;

	public Data() {

		super();

	}

	public void setStudent_number(String student_number) {

		this.student_number = student_number;

	public void setLast(String last) {

		this.last = last;

	}

	public void setFirst(String first) {

		this.first = first;

	}
```

```
	public void setNumber(int number) {
		this.number = number;
	}
	public void setSubject(String subject) {
		this.subject = subject;
	}
	public void setSubject_Name(String subject_Name) {
		this.subject_Name = subject_Name;
	}
	public String getStudent_number() {
		return student_number;
	}
	public String getLast() {
		return last;
	}
	public String getFirst() {
		return first;
	}
	public int getNumber() {
		return number;
	}
	public String getSubject() {
		return subject;
	}
```

```
	public String getSubject_Name() {
		return subject_Name;
	}
```

（2） AttendanceLogic クラス

SearchData クラスでデータを検索し，{学生の氏名，時間割の科目名} を電子出席簿に表示するサーブレットである．

リスト 6.1.2 AttendanceLogic.java

```
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


@WebServlet("/ATTENDANCE")
public class MainLogic extends HttpServlet {
	@Override
	protected void doGet(HttpServletRequest req,  HttpServletResponse resp)
throws ServletException,  IOException {
		doPost(req,  resp);
	}


	@Override
	protected void doPost(HttpServletRequest req,  HttpServletResponse resp)
throws ServletException,  IOException {
		SearchData sd = new SearchData(req);
		sd.searchAll();
```

```
                RequestDispatcher   disp   =   req.getRequestDispatcher("WEB-
INF/jsp/Attendance.jsp");
                disp.forward(req,  resp);
        }
```

（3）SearchData クラス

学生テーブルのデータと時間割テーブルの科目データをそれぞれ Data 型のリストに格納し，セッションにより保持している．

セッションに保持した Data 型リストは jsp ファイルで扱うことができる．

リスト 6.1.3　SearchData

```
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

public class SearchData {
	private HttpServletRequest request;
	public SearchData(HttpServletRequest request) {
		this.request = request;
	}

	public void searchAll() {
		List<Data> studentinfo = new
SearchDataDAO().studentinfosearch();
		List<Data> subjectinfo = new
SearchDataDAO().subjectinfosearch();
		if (studentinfo != null) {
			HttpSession session = request.getSession(true);
			session.setAttribute("studentinfo", studentinfo);
			session.setAttribute("subjectinfo", subjectinfo);
}
```

（4）SearchDataDAO.クラス

Studentinfosearch メソッドと subjectinfosearch メソッドがあり，学生テーブルと時間割テーブルを参照し，データベースの接続，操作を行っている．どちらも JDBC という Java API でデータベースに接続している．データベースへの接続 URL はリスト 6.1.4 の通りである．

学生テーブルの SQL 文は，学生情報を全て参照している．

時間割テーブルの SQL 文は，科目 ID を外部キーとして，科目テーブルと時間割テーブルを内部結合している．リスト 6.1.4 では，例として 2024 年 1 月 2 日の時間割を参照している．

リスト 6.1.4　SearchDataDAO.java

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;


public class SearchDataDAO {

        public List<Data> studentinfosearch() {

                List<Data> lists = new ArrayList<Data>();

                try {

                        // JDBC Driver の登録

                        Class.forName("com.mysql.cj.jdbc.Driver");
```

```
                        // Connection の作成
                        Connection conn = DriverManager.getConnection(
                                        "jdbc:mysql://172.16.64.1/iitacadm" ,
"iitacadm",  "takataroyoukota");
                        // SQL の準備
                        PreparedStatement pstmt = conn.prepareStatement(
                                        "SELECT * FROM student ");
                        //セレクトの実行
                        ResultSet rset = pstmt.executeQuery();
                        //ResultSet の参照
                        while (rset.next()) {
                                Data list = new Data();
                list.setStudent_number(rset.getString("student_number"));
                                list.setNumber(rset.getInt("number"));
                                list.setFirst(rset.getString("first_name"));
                                list.setLast(rset.getString("last_name"));
                                lists.add(list);
                        }
                        //各リソースの開放
                        rset.close();
                        pstmt.close();
                        conn.close();
```

```
                } catch (Exception e) {
                        e.printStackTrace();
                }
                return lists;
        }
        public List<Data> subjectinfosearch() {
                List<Data> lists2 = new ArrayList<Data>();
                try {
                        // JDBC Driver の登録
                        Class.forName("com.mysql.cj.jdbc.Driver");
                        // Connection の作成
                        Connection conn = DriverManager.getConnection(
                                        "jdbc:mysql://172.16.64.1/iitacadm" ,
"iitacadm",  "takataroyoukota");
                        // SQL の準備
                        PreparedStatement pstmt = conn.prepareStatement(
                                        "SELECT * FROM timetable INNER JOIN
syllabus ON timetable.subject_id = syllabus.subject_id WHERE time = '2024-01-
02'");
                        //セレクトの実行
                        ResultSet rset = pstmt.executeQuery();
                        //ResultSet の参照
                        while (rset.next()) {
```

```
        Data list2 = new Data();
        list2.setSubject(rset.getString("subject_name"));
        lists2.add(list2);
        }
        //各リソースの開放
        rset.close();
        pstmt.close();
        conn.close();
        } catch (Exception e) {
                e.printStackTrace();
        }
        return lists2;
        }
}
```

（5）Attendance.jsp ファイル

電子出席簿を表示する JSP ファイルである．電子主席簿の UI については第 5 章で前述しているため，データベースとの連携部分のみ説明する．SearchData クラスでセッションとして保持した学生テーブルのデータと時間割テーブルのデータを HTML データの中に埋め込んでいる．

リスト 6.1.6　Attendance.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.util.List"%>
<%@ page import="sotuken2023.Data"%>
<%@ page import="javax.servlet.http.HttpSession" %>
<%
      List<Data>                    studentinfo                    =
(List<Data>)session.getAttribute("studentinfo");
      List<Data>                    subjectinfo                    =
(List<Data>)session.getAttribute("subjectinfo");
%>
                                    (省略)
const database = [<% for(Data list:studentinfo){ %>
{ id: <%= list.getNumber() %>,  name: "<%= list.getFirst() %><%=
list.getLast() %>"},
<% } %>];
const database2 = [<% for(Data list2:subjectinfo){ %>
"<%= list2.getSubject() %>", <% } %>]
                                    (省略)
```

第７章　　まとめ

7.1 データベース･Java クラス

本研究で取り組むことができた主な点は以下のとおりである.

- 内部サーバのデータベース作成
- 一部，Java クラスの定義と JSP 化

本研究で取り組めなかった主な点は以下のとおりである.

- 外部サーバのデータベース作成
- 電子出席簿での届け出情報テーブルを参照した欠席理由，欠席開始時間，終了時間の取得
- 電子出席簿での届け出情報テーブルを参照した欠席開始時間，終了時間から自動で出欠席ボタンのチェック
- 時間割システムでの時間割テーブルを参照した時間割情報の取得
- 時間割システムでの作成したコマ情報をコマテーブルにインポートする
- 時間割システムでの教員テーブルを参照した教員カラーをコマ情報に反映させる

7.2　時間割システム

本研究で取り組むことができた主な点は以下のとおりである.

- 画面レイアウト
- フォームでのコマ目作成
- コマのドラッグアンドドロップ機能

本研究で取り組めなかった主な点は以下のとおりである.

- 特定のコマ目削除

7.3　電子出席簿

本研究で取り組むことができた主な点は以下のとおりである.

- 画面レイアウト
- 一日単位での表示

本研究で取り組めなかった主な点は以下のとおりである.

- 出欠席データの CSV ファイル出力
- 無断欠席者への対応
- 同一人物の欠席データを一日に複数入力できるように

以上のことを来年度以降の卒業研究で取り組んでほしい.

第 8 章　　おわりに

本研究において，テーマ発表会で寄せられた多くの要望を受け，過去 3 年間の成果物を 1 年で向上させようとしたが，電子出席簿，時間割システムとデータベースとの連携が完全にできたわけではなく，想定していた動作の確認もできておらず，できるところまでやったというような形になった．

途中までしかできなかったことはとても悔しいことだが，研究の土台となる部分は，ある程度できていると思うから，第 7 章で述べたようなことを来年度以降の卒業研究で引き継いでシステムを完成させてほしいと思う．