The Hong Kong Polytechnic University

Department of Computing

COMP4913 Capstone Project

Report (Final)

An online system for examination timeslot allocation

Student name: Lee Ka Shue

Student ID No.: 19034824d

Program-stream code: 61431-FCS

Supervisor: Dr WANG Qixin

Co-Examiner: Dr SAXENA Divya

2nd Assessor: Prof. TAN KC

Submission date: 12 April 2024

Abstract

This report briefly states the importance of an examination timeslot allocation system for higher educational institutes, outlines previous efforts in implementing such a system and its shortcomings, describes the steps taken to improve the project, and covers the shortcomings within the project and future mitigation steps.

The final project has partially completed the outcomes and objectives stated during the project proposal phase, improving the project's development sustainability. However, due to the choice of framework has a high amount of complexity, caution is suggested for prospective contributors.

Contents

List of tables and figures

Background & problem statement

Introduction

COVID has caused major disruptions in our daily lives, including the education processes. As face-to-face interactions have been discouraged during the period, examinations were also to be conducted online, which brought forth the problem, invigilating a couple dozen of students through webcam feeds concurrently being quite difficult. While a severe resource demand, 1:1 monitoring of students is a much-needed process to eliminate cheating and ensure fairness. This then led to the need of allocating timeslots for individual students' examinations.

Although COVID-19 is now a part of history, there are no guarantees that no future circumstance necessitates online examinations. Hence, a suitable platform is needed to coordinate the timeslot allocation process.

Background

An existing teacher-student meeting reservation system has previously been extended and improved by senior students into a teacher-student examination timeslot preference/allocation system, and student Yu's project has been selected to be iterated upon by this year's students.

Problem

For this year, while not having specified a problem, us students have been tasked to "perfect" the system.

Having seen the project demonstration, it can be seen that the finished work is more-or-less in a prototype state, which may require some further development and polishing to reach a production ready state. But upon further inspection, the maintenance of the project may prove difficult.

Objectives & outcomes

Objectives

As aforementioned, the given objective is to:

0. “Perfect” the system.

Additionally, the major objectives of the Capstone Project subject should be considered as well:

1. Develop generic competencies
2. Prepare for workplace professional practice
3. Further academic pursuits
4. Lifelong learning

Outcomes

During the project proposal and project initialization phase, I concluded that the following facets of the system will need to be faced with scrutiny to identify suboptimal implementations and be iterated upon:

A. Technical design
    I. DB-level design
    II. Functional business logic
B. UI/UX design
    I. Language
    II. Localization / Internationalization
    III. Interaction flow
C. General code health
    I. Code tests
    II. Documentation
    III. Project structure
    IV. Convention adherence

Implementation

General

As an unfortunate compromise out of my full-time SWE workload, outside of the aforementioned areas that needs improvement, the database administration and user interaction platform has also been changed to utilized JMix instead of MyPHPAdmin.

Technical design

Framework

Existing implementation

Yu's work utilizes PhpMyAdmin, MySQL, PHP, and Apache to power the database and web interface (Appx. A).

Current implementation

As the author, I, have had minimal experience with web development, I have to learn all basics from the start. As such, whether continuing development with PHP and learning it, or implementing it from scratch using another framework is a decision I had to make.

Personally, I was intrigued by the Phoenix framework with Elixir as the programming language, as I have seen testimonials about its performance capabilities and am interested in learning functional programming.

Professionally, I consulted my internship workplace supervisor, and was recommended to use the Jmix framework with Java as the programming language, as the company uses that framework for multiple of their solutions as well, and I would be tasked on the job to use this framework.

Resultantly, I have decided to listen to my workplace supervisor due to multiple reasons:

1. Full-time software engineering work severely limits available time to learn new technologies and develop with it. If Jmix was used, the experience during work will help with my capstone project and vice versa.
2. Jmix is an open-source, bundled implementation of Vaadin, a frontend framework, and Spring, a backend framework. This allows quicker development of features and can empower me to do better documentation with the added time.

3. Utilizing Spring, Jmix is a solidly enterprise solution, as Spring is an overwhelmingly popular framework in the business world. This will enable future project contributors' and my career development.
4. I was recommended by the one who feeds me.

Additionally, there are benefits of built-in user system, resulting in better security.

Database design

Existing implementation

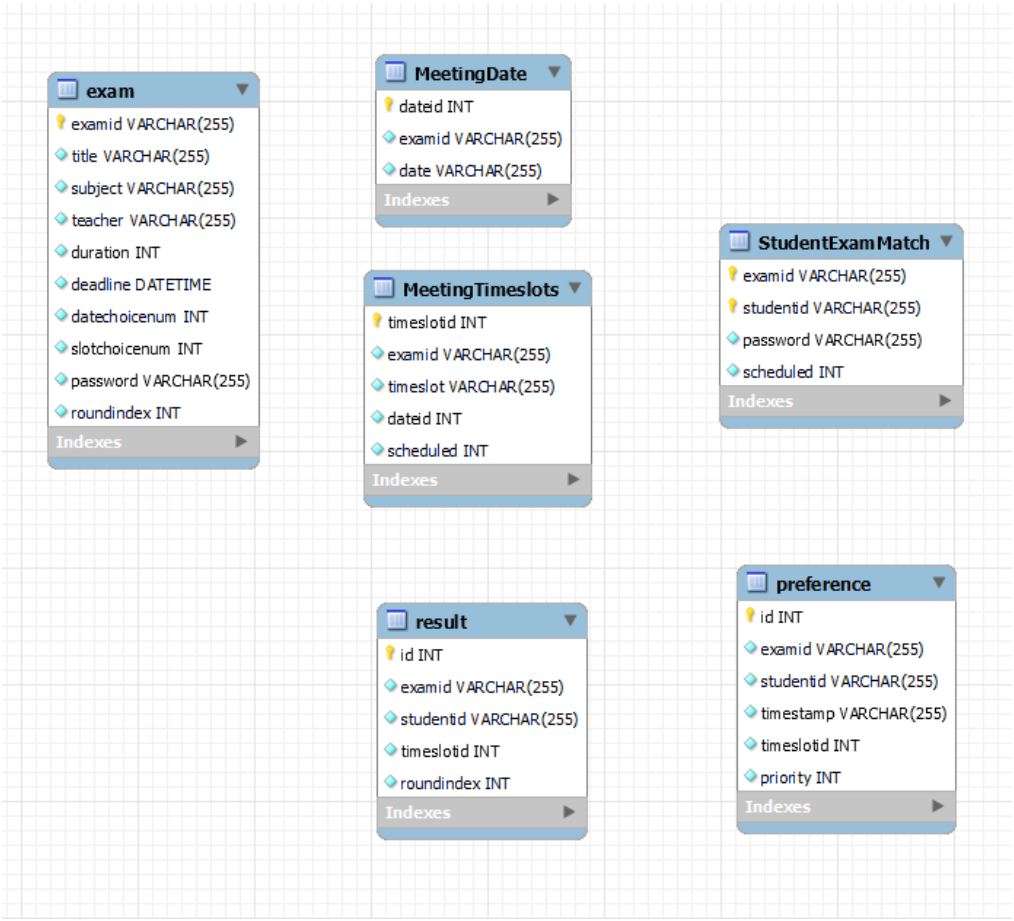


Figure 1 Database schema ER diagram of previous work

Above is the ER diagram of YU's existing work, generated utilizing MySQL workbench's model generation function from the provided SQL script with minor amendments on table studentexammatch's name casing to avoid errors.

Interestingly, the SQL script seemingly does not enforce entity relationships. The naming scheme for tables and columns are less than ideal as well, with inconsistent table casing patterns, non-existent column casing patterns, inconsistent terminology,

and vague naming. Following text references to such information will follow the casing convention described in Convention adherence

To give the benefit of doubt, as I am not familiar with PHP, the relationship business logic could be stored in that layer. As such, please refer to YU's final report's diagram (Appx. B) for the intended relationships.

Current implementation

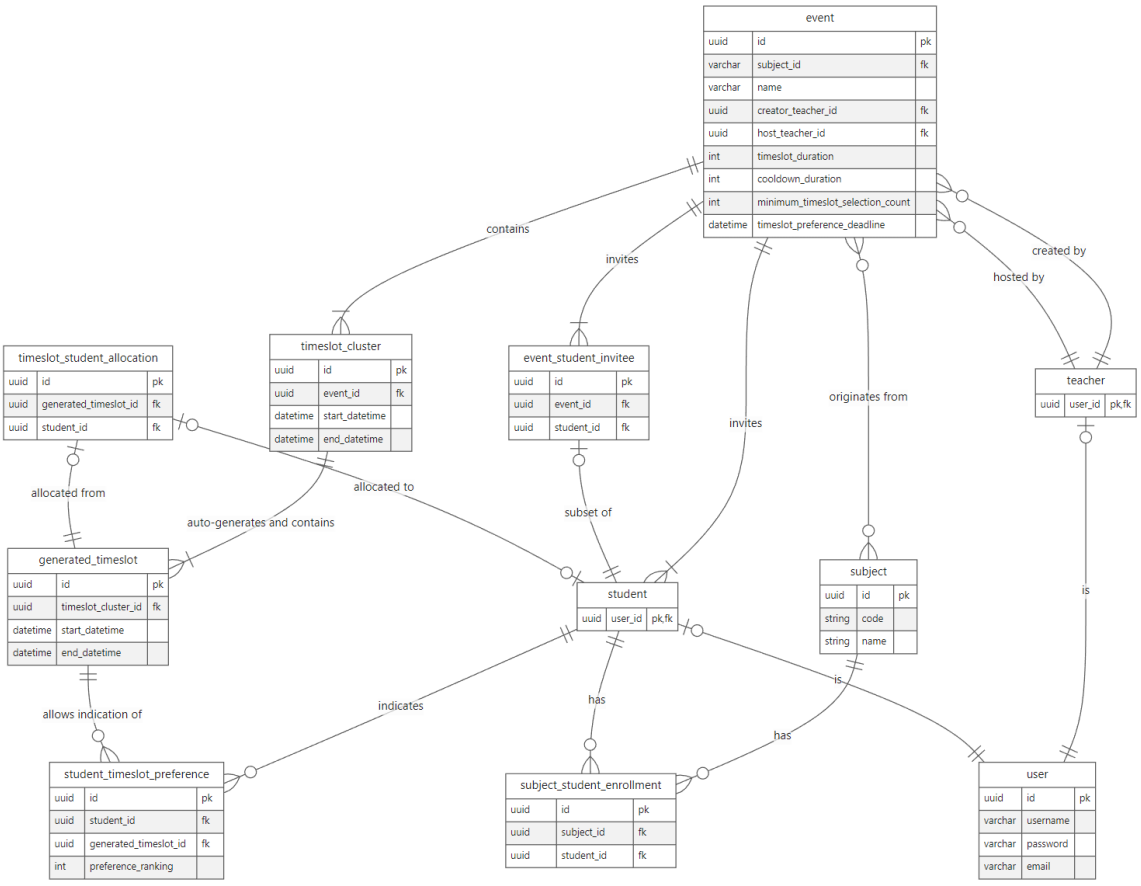


Figure 2 Database schema of current work

Above is the ER diagram of my work, generated utilizing Obsidian and Mermaid.JS from my living design / implementation markdown document. For parity, a diagram is also generated using MySQL workbench, which is relegated to the (Appx. C).

There is a separation of graphics as there are some framework-induced additional tables and columns that are unnecessary for understanding the design, and I have either abstracted away or removed them.

Normalization

During the design of the new DB schema, I had considered the spectrum across de-normalization and normalization.

Approach	*Normalization*	…	*Denormalization*
Consistency	Better		Worse
Complexity	Worse		Better

After more research, I have discovered the distinction between Online analytical processing (OLAP) and Online transactional processing (OLTP) in the software world, where OLAP is geared towards performant analytics by de-normalization, whereas OLTP is geared towards consistent processes by normalization.

Therefore, the new implementation opted for a comparatively normalized approach, as the information are not used in a throughput-first manner.

Expansion

There are also new tables introduced, including subjects and subject enrollments, as these tables would bring forth a greater quality of life for users by simplifying certain processes, and this information should be readily available in the school's systems.

For the ease of understanding, the above diagram will be further simplified to the point of view (POV) of the teacher and student respectively

Teacher POV

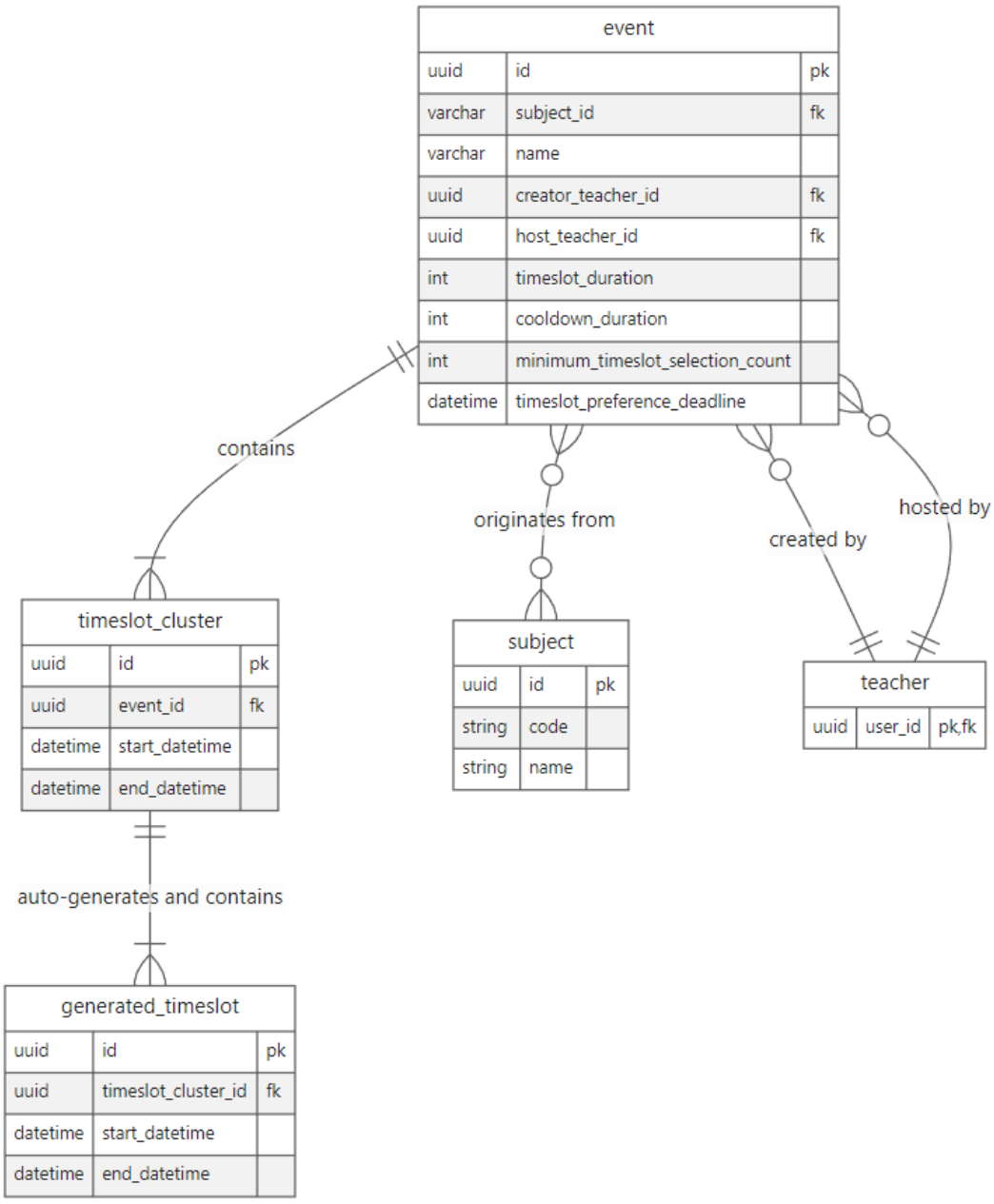


Figure 3 Simplified ER diagram from the teacher's POV

The new implementation majorly deviates from the previous' in the following places:

1. MEETING_DATE → TIMESLOT_CLUSTER
   Abstracted date separator into cluster for higher flexibility
2. Various normalizations by removing redundant columns
3. Explicit start and end times, and also a specified EVENT.cooldown_duration, which allows for better timeslot scheduling and explicit communication

4. Separated event creator and host (i.e. possibility of explicit declaration by event creating professor of invigilating assistants)

Student POV

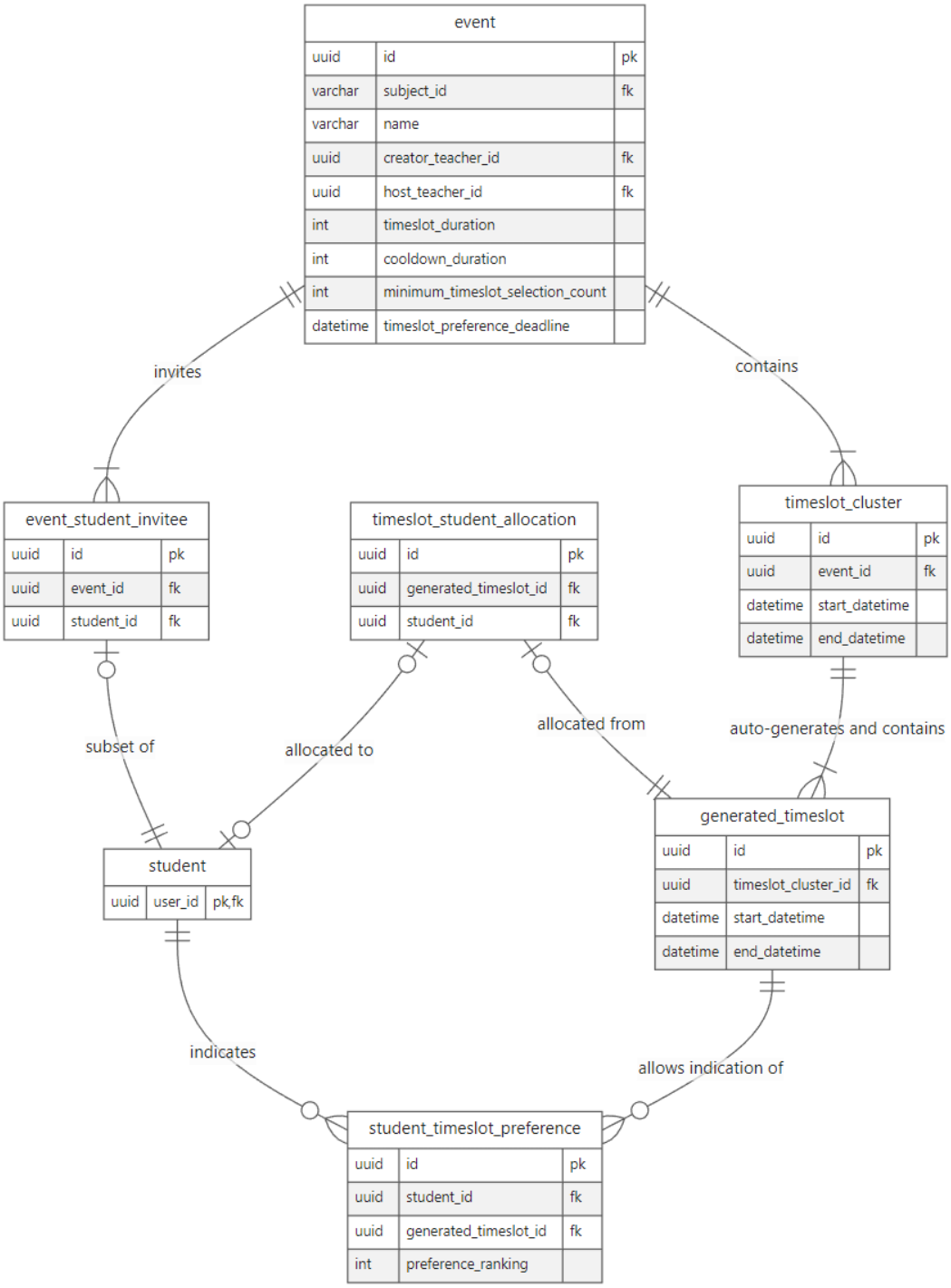


Figure 4 Simplified ER diagram from the student's POV

The new implementation majorly deviates from the previous' in the following places:

1. TIMESLOT_CLUSTER (formerly MEETING_DATE) no longer directly related to EVENT_STUDENT_INVITEE (formerly STUDENT_AND_EXAM_MATCHING)

Functional business logic

To accommodate the new schema, changed framework, and altered UI/UX design, the relevant code has been altered accordingly, such as the timeslot generation code, in order to accommodate for EVENT inter-timeslot cooldown_duration.

UI/UX design

Localization / Internationalization (l10n / i18n)

As a framework feature, the interface now has multi-language support, and has EN and ZH_HK locales implemented.

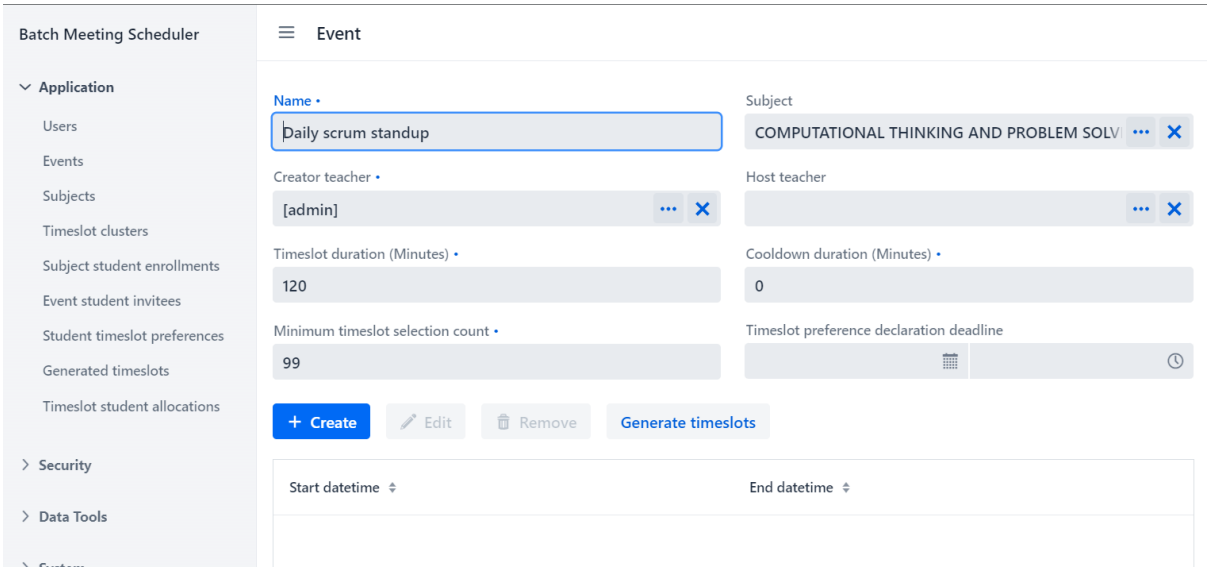


Figure 5 Web GUI in en locale

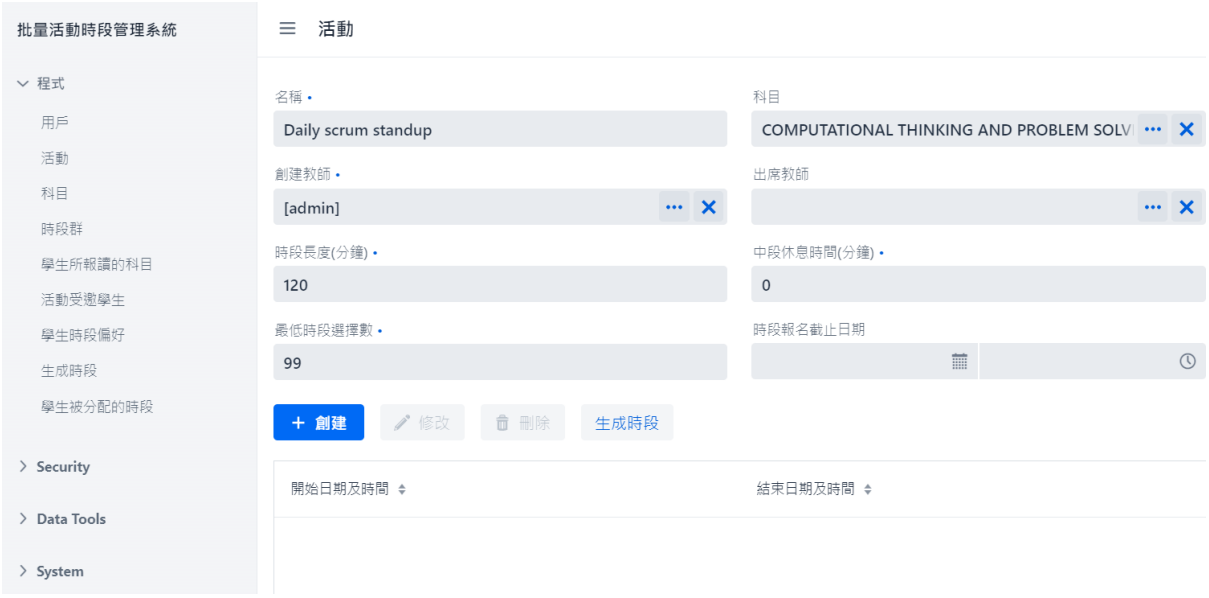


Figure 6 Web GUI in zh_HK locale

Interaction routes

The current revision has streamlined the UI, such as combining the view and edit pages of allocation results.

Previous implementation

Previously, the UI has been unnecessarily fragmented, and goes against the typical “modern” design, such as different navigation paths to the check results and edit results (Fig. 7).

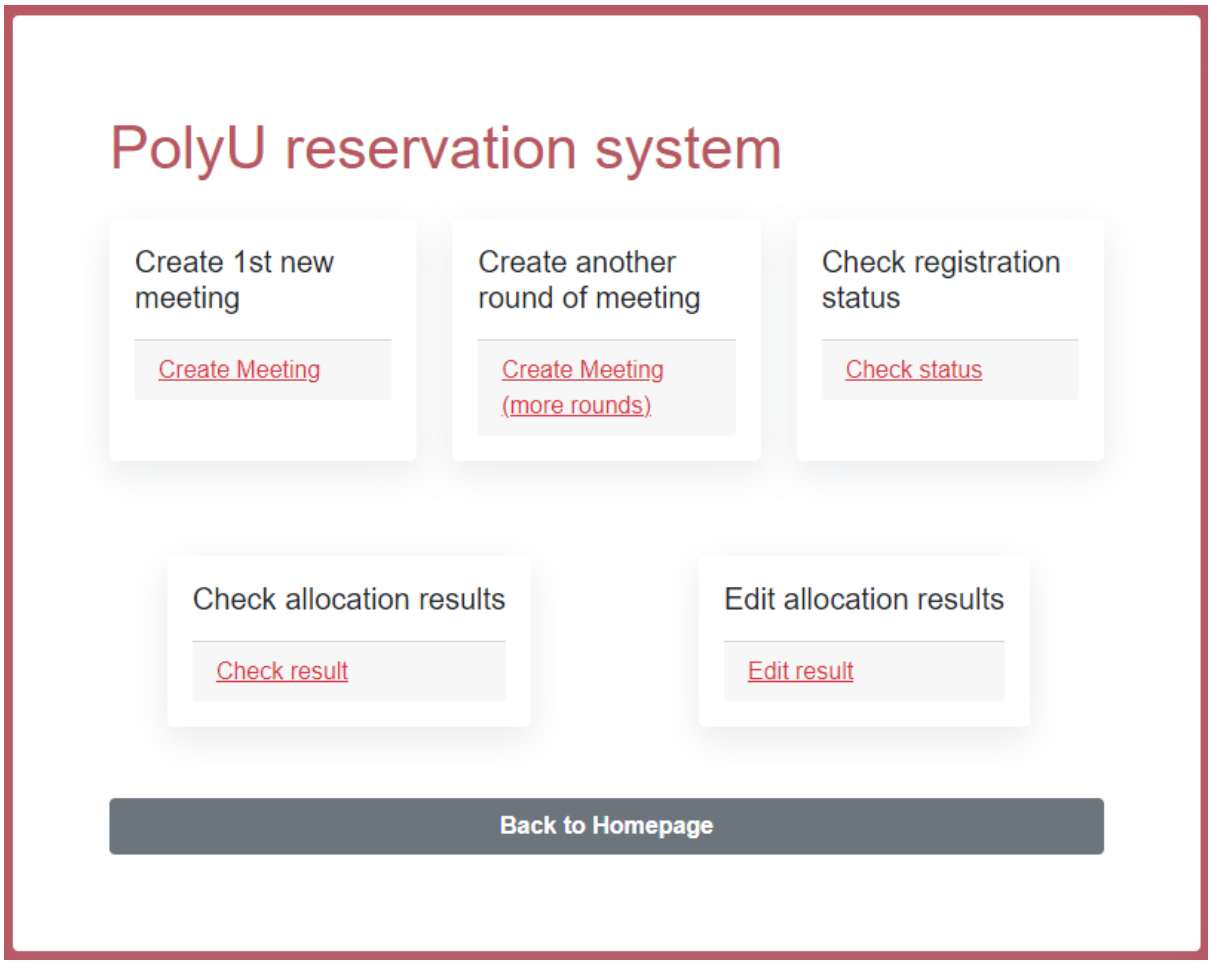


Figure 7 GUI of teacherview in previous work

Current implementation

Currently, the UI allows the user to view records, allocation results in this example, then navigate into a deeper route to create and edit (and remove) results.

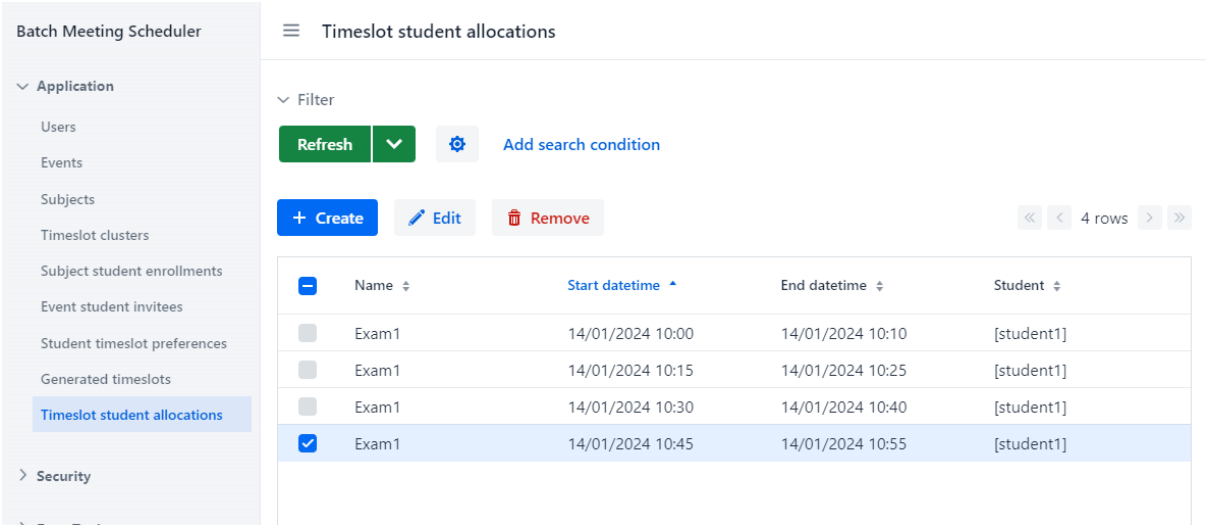


Figure 8 Web GUI of allocation results of current implementation

Code health

Reliability assurance

The project now has a minimal amount of tests covering user entities.

Documentation

Previous implementation

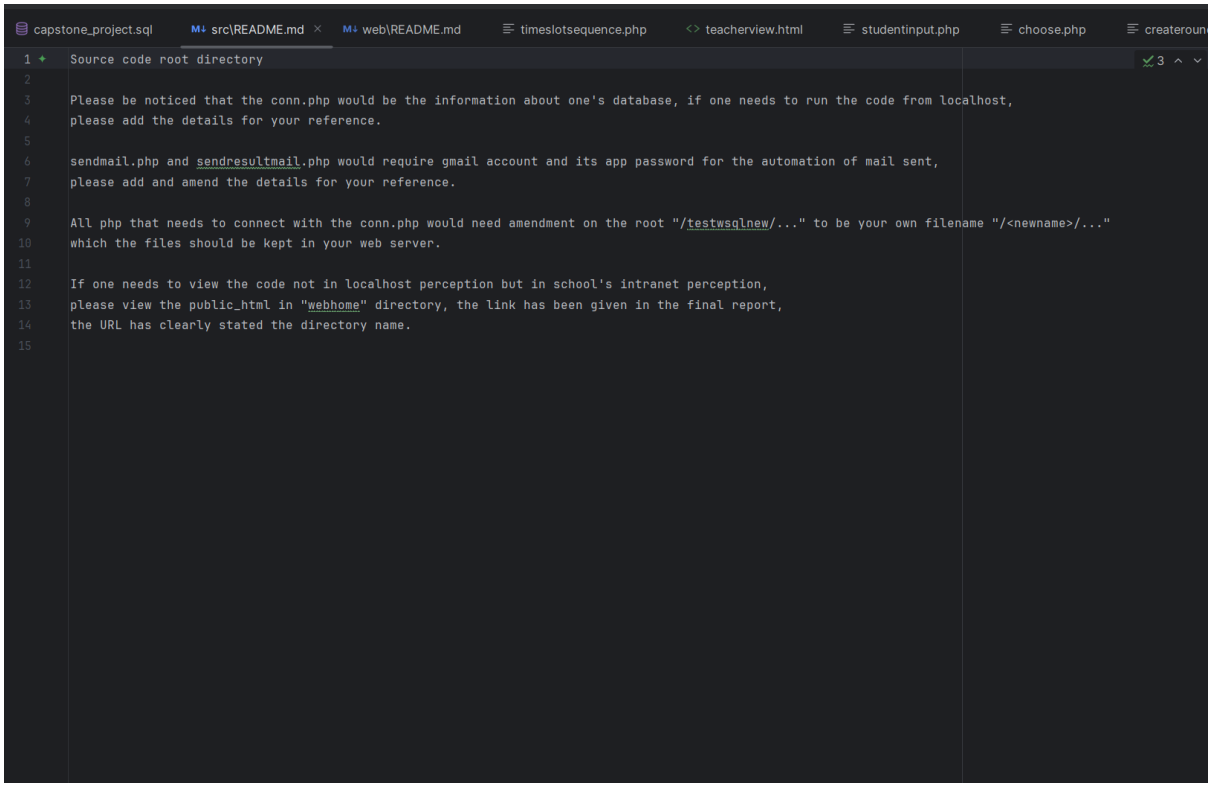
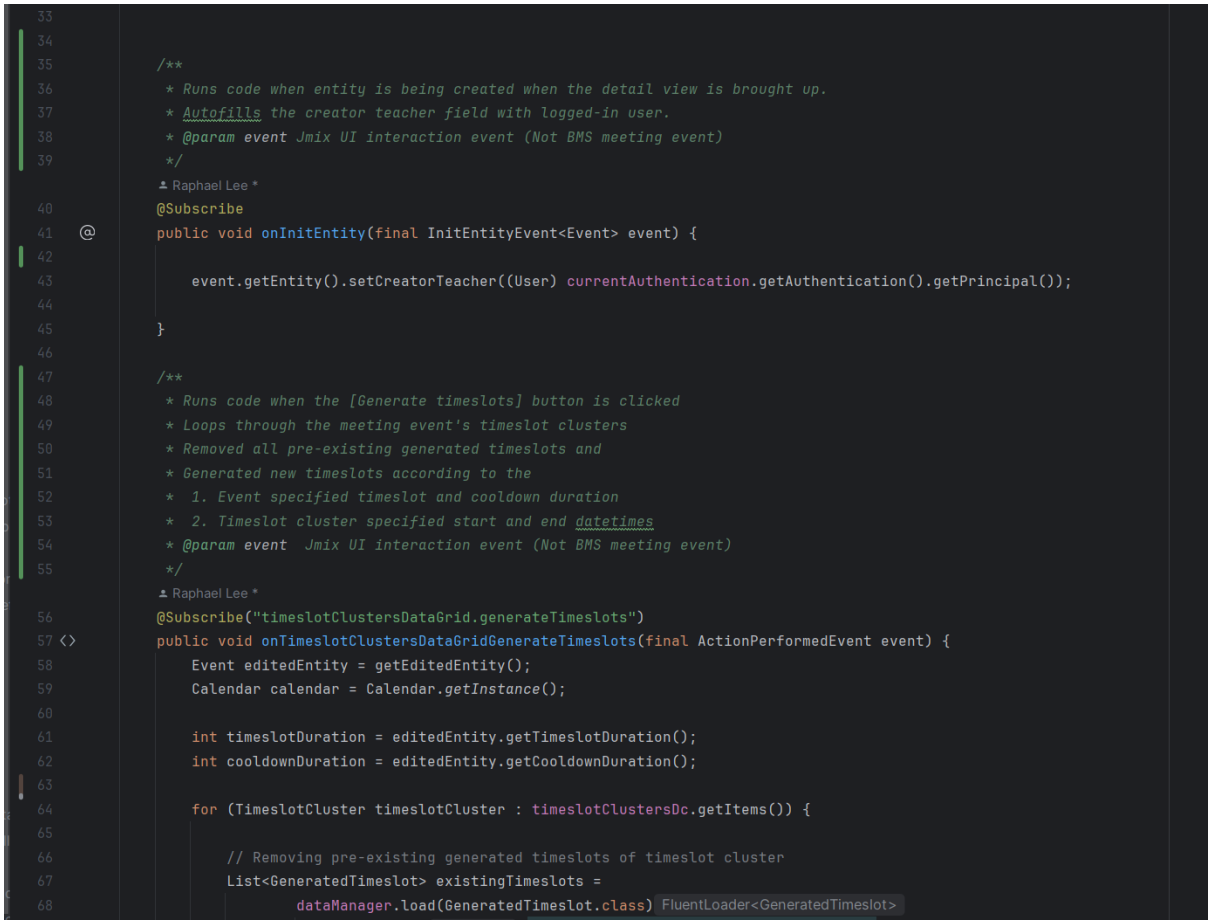


Figure 9 The only documentation of previous work

Previously, the repository has little-to-no documentation.

Current implementation



```
33
34
35          /**
36           * Runs code when entity is being created when the detail view is brought up.
37           * Autofills the creator teacher field with logged-in user.
38           * @param event Jmix UI interaction event (Not BMS meeting event)
39           */
             Raphael Lee *
40          @Subscribe
41  @       public void onInitEntity(final InitEntityEvent<Event> event) {
42
43              event.getEntity().setCreatorTeacher((User) currentAuthentication.getAuthentication().getPrincipal());
44
45          }
46
47          /**
48           * Runs code when the [Generate timeslots] button is clicked
49           * Loops through the meeting event's timeslot clusters
50           * Removed all pre-existing generated timeslots and
51           * Generated new timeslots according to the
52           *  1. Event specified timeslot and cooldown duration
53           *  2. Timeslot cluster specified start and end datetimes
54           * @param event  Jmix UI interaction event (Not BMS meeting event)
55           */
             Raphael Lee *
56          @Subscribe("timeslotClustersDataGrid.generateTimeslots")
57  <>      public void onTimeslotClustersDataGridGenerateTimeslots(final ActionPerformedEvent event) {
58              Event editedEntity = getEditedEntity();
59              Calendar calendar = Calendar.getInstance();
60
61              int timeslotDuration = editedEntity.getTimeslotDuration();
62              int cooldownDuration = editedEntity.getCooldownDuration();
63
64              for (TimeslotCluster timeslotCluster : timeslotClustersDc.getItems()) {
65
66                  // Removing pre-existing generated timeslots of timeslot cluster
67                  List<GeneratedTimeslot> existingTimeslots =
68                          dataManager.load(GeneratedTimeslot.class) FluentLoader<GeneratedTimeslot>
```

Figure 10 Documentation of timeslot generation code

Currently, manually created code are documented. Deployment guides are also authored for better handover experiences (Appx D.). Graphical elements are also written in an easily extendible way with Obsidian, Mermaid.JS and markdown files.

Project structure

Previous implementation

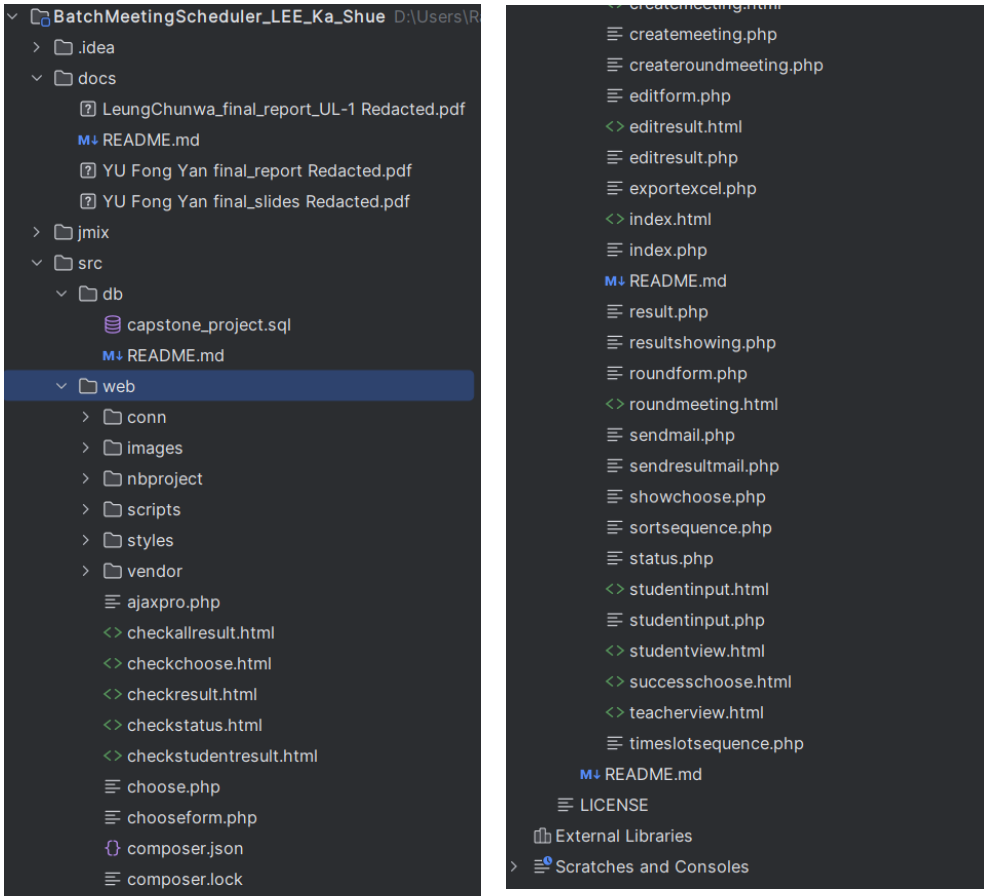


Figure 11 Project structure of previous work

Current implementation

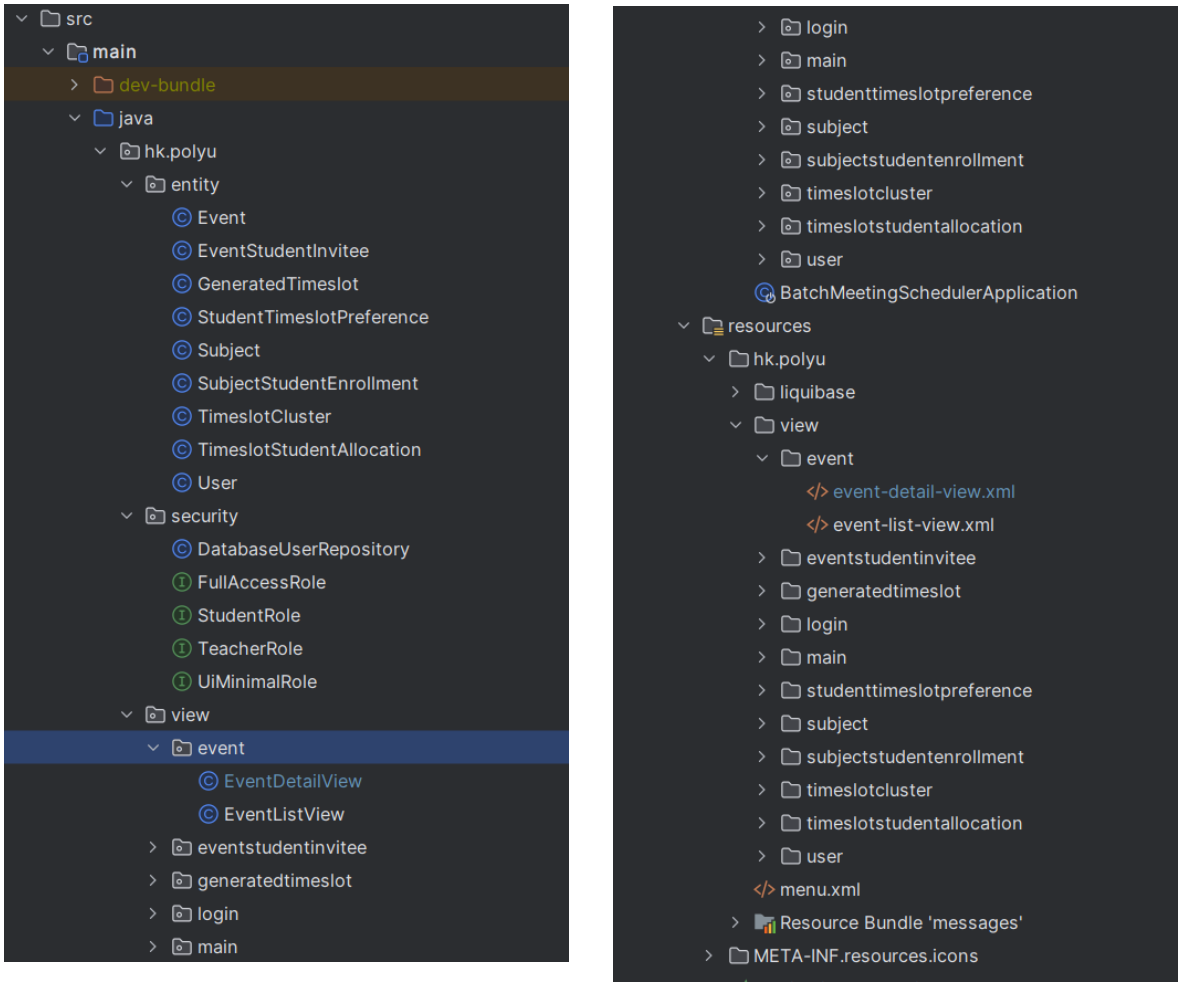


Figure 12 Project structure of current work

Convention adherence

The code repository contents now follow a naming and casing convention.

Figure 13 Project directory and file names of previous work

	Type	*Casing (stylized)*	*Plurality*
	Table name	MACRO_CASE	Singular
	Column name	snake_case	Singular

Domain	*Previous*	*Current*
Table	MEETING	EVENT

Table	EXAM	EVENT
Table	MEETING_TIMESLOTS	GENERATED_TIMESLOT
Table	STUDENT_AND _EXAM_MATCHING	EVENT_STUDENT_INVITEE
Table	PREFERENCES	STUDENT_TIMESLOT_PREFERENCE
Table	RESULT	TIMESLOT_STUDENT_ALLOCATION
EVENT	title	name
EVENT	teacher	creator_teacher
EVENT	duration	timeslot_duration
EVENT	deadline	timeslot_preference _declaration_deadline
EVENT	slot_choice_num	minimum_timeslot_declaration_count

Migration checklist

Functionality	*Migrated?*
Database	Yes
Record input GUI	Partial
Timeslot generation	Yes
Student e-mail invitation	No

Objective and outcome checklist

	Objective / Outcome	*Chk*	*Description*
1	Develop generic competencies	☑	Developed a database and front/backend system
2	Prep for workplace professional practice	?	Developed a system with processes, efficiency, conventions, maintainability, etc. in mind Neglected to communicate with professor

3	Further academic pursuits	?	Performed minor research and write up in DB OLTP / OLAP design Relates to CAP / PACELC theorem
4	Lifelong learning	?	Learned some things about development, workplace, and myself?
A.I	DB design	☑	Extended and normalized DB with defined details
A.II	Functional business logic	?	Partially migrated functions
B.I	Language	☑	Rewritten platform for cohesive experience
B.II	l10n / i18n	☑	Implemented and translated for multi-language support
B.III	Interaction flow	☑	Implemented nested interaction routes
C.I	Code tests	?	Auto-created user tests Failed to implement other tests Resulted in low test coverage
C.II	Documentation	☑	Written project documents, code documents, etc
C.III	Project structure	☑	Created project according to Java package conventions, & Relied on framework generated structures
C.IV	Convention adherence	☑	Rewritten platform for cohesive naming and casing experience
0	Perfection	?	Introduced both improvements and regressions Arguably for the better by allowing easier future development

Postmortem

As stated in the previous sections, not all stated objectives and outcomes are achieved, unfortunately. To prevent such incidents from happening again, reasons and preventive measures are to be investigated.

Overpromising & underachieving

Set out to migrate features to a new framework, I ended up short on implementing functionality of sending out e-mail invites to students and implementing distinct, well-defined GUI views for different roles (i.e. students and teachers).

The layer beneath this symptom is a lack of time, but the root-cause are multi-faceted:

1. Myself
    a. Arrogance: Thought too highly of myself in regards to the speed of learning a new framework, causing underestimation of fulfilment time needed
    b. Laziness: Was not able to dedicate more time to working instead of entertainment, causing overrun of estimated schedule
    c. Perfectionism: Scope-creeped during development, extending the DB as development went on, causing an increased fulfilment time needed
    d. Procrastination: Allowed my mind to wander when developing the system often, causing overrun of estimated schedule
    e. Naivety: Believed verbal future work plan promise on WFH policies and utilized framework, causing increased learning time needed and overrun of estimated schedule
2. Environment
    a. Unexpected toolset changes: Workplan / project changed to other domains instead of Jmix, causing increased learning time needed and overrun of estimated schedule
    b. Unexpected schedule changes: Workplace cancelled WFH policy, causing higher fatigue and decreased available time
    c. Unfamiliar environment: Started living on my own, causing a decreased available time

To avoid such a problem from arising again,

1. As suggestions to myself:
    a. Avoid over-employment / moonlighting, as my mental fortitude is weak and could not support large amount of work hours / week.
    Work regular hours or even consider taking less pay for less work in the future.

   b. Avoid planning according to verbal promises, WFH and future work are not set in stone. Get promises in contract, and also plan contingency plans even if the contracts fall through.
   c. Avoid the underperform → “I can salvage this” → depression → underperform negative feedback loop. Sometimes things just go south, get help.
2. As suggestions to potential project contributors:
   a. Consider available time, interest in enterprise Java CRUD software, and resultant commitment-availability. Spring can be overwhelming complex to begin learning. So, decide to use familiar frameworks or interested ones accordingly.

Conclusion

This project’s efforts are mostly for paying back the accumulated technical debt and temporarily alleviating the yearly knowledge-transferal (or the lack thereof) by establishing better conventions and patterns, which enables the project’s extendibility by allowing easier future changes.

While the framework migration was not fully realized and some regressions exist, it should be an overall improvement to the development experience and future possibilities.

The only thing to be concerned with, is the extensive level of involvement needed to learn the Jmix / Spring framework.

Future work

As a byproduct of the framework change, any future developers can easily enable a REST API plugin which provides an alternative, totally decoupled method of enhancing the system, so even if future documentations fall short of any newcomers expectations, a system-wide grokking and re-engineering should not typically be necessary.

Resource utilization

- Department managed URL

- Department allocated docker container running environment
- Department allocated MySQL DB connection
- Jmix academic license(s) (free)

Used software

- Obsidian
- Mermaid.JS
- IntelliJ Idea
- Jmix
- Java 17
- Docker
- MySQL

Appendices

Appendix A

Resources Utilization

Hardware

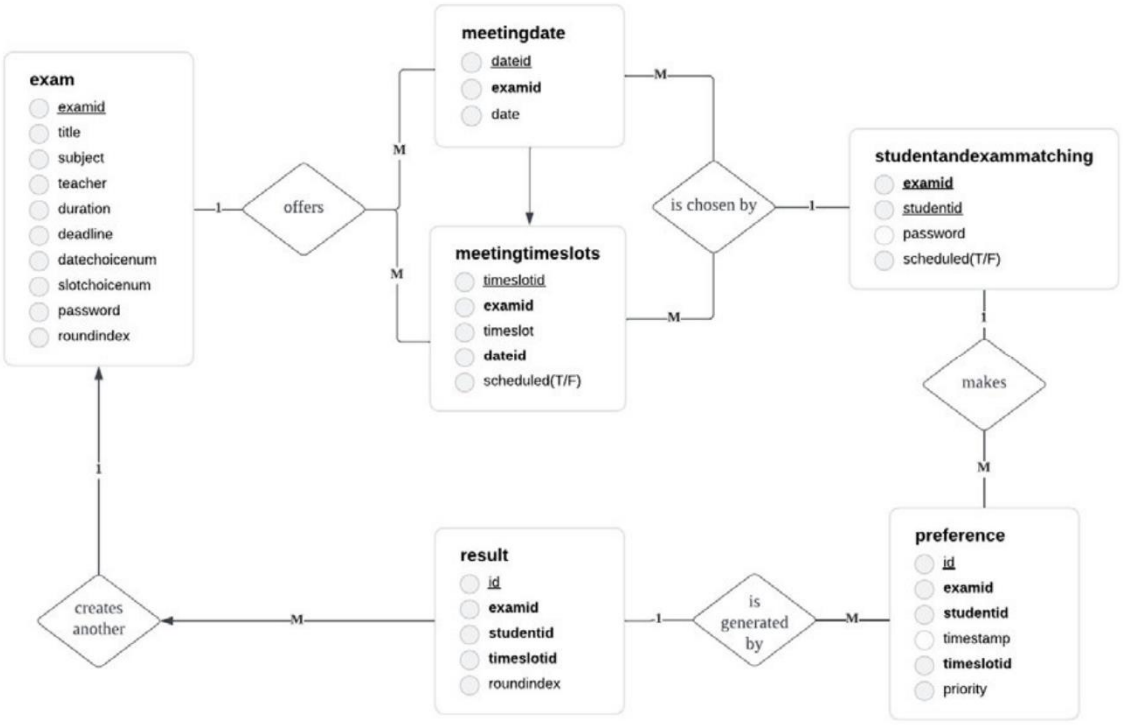
- Computer for web development
- Web server (provided by PolyU)

Software

- PhpMyAdmin
- MySQL
- Php
- Apache
- WinSCP (accessing personal page in PolyU server)

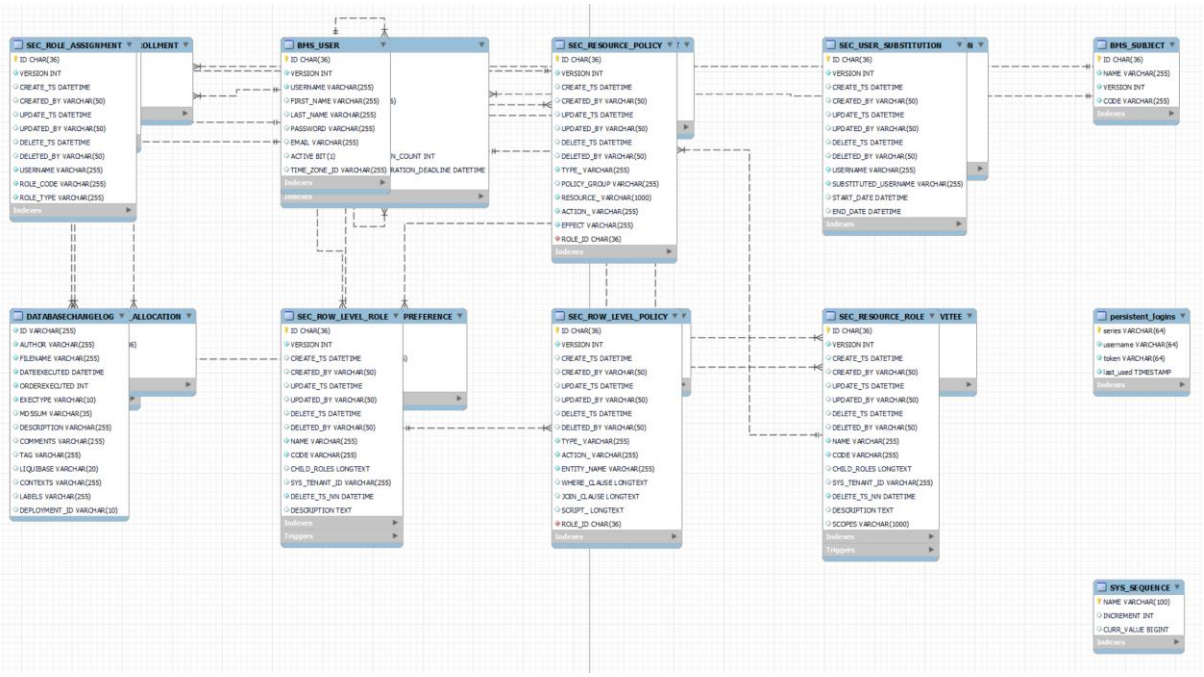
Resource utilization of senior YU Fong Yan's project (Final report excerpt)

Appendix B



ER diagram of Yu Fong Yan's project's DB schema (Final report excerpt)

Appendix C



ER diagram of current system’s DB schema from MySQL Workbench

Note that there are 21 tables and are visually stacked on generation

Appendix D

Deployment guide

1. Create docker network:

```
docker network create bms-network
```

2. Create MySQL database container volume:

```
docker volume create bms-db-volume
```

3. Create MySQL database container:

```
docker run --name bms-db-container -d ^
    -p 3306:3306 ^
    -e MYSQL_ROOT_PASSWORD=root ^
    -e MYSQL_ROOT_HOST=% ^
    --network=bms-network ^
    --mount source=bms-db-volume,target=/var/lib/mysql ^
    --restart unless-stopped ^
    mysql:8.0.36
```

4. Create database:

```
mysql -uroot -proot
```

```
CREATE DATABASE bms;
```

5. Configure Data Store to use docker network hostname

Create BMS Front/Back-End image

```
./gradlew "-Pvaadin.productionMode=true" bootBuildImage
```

6. Create BMS F/BE container

```
docker run --name bms-container -d ^
    -p 8080:8080 ^
    --network=bms-network ^
    --restart unless-stopped ^
    batch-meeting-scheduler:0.0.1-SNAPSHOT
```

Deployment guide with copy-able snippets, written in markdown, viewed in Obsidian