

# Team 19 ChairVisE 4.0

Code Repository URL:

<https://github.com/CS3219-SE-Principles-and-Patterns/cs3219-ay2021-s1-project-2020-s1-g19>

<b>Student Name</b>	Chua Huixian	Keith Tan Jia Wen	Loh Sze Ying	Ng Jun Rong Terence	Toh Sheng Rong
<b>Matriculation Number</b>	A0190360M	A0168466N	A0185673R	A0184003U	A0226511R

**Individual contributions to the project:**

Chua Huixian	Frontend: Added navigation bar, adjusted homepage, creation of conference and corresponding version when importing data, fixed add section bug with Sze Ying Report: Shared
Keith Tan Jia Wen	Backend: Added schema file, new api endpoints, functionality to store multiple conference data and authentication aspect to verify access. Also converted logic classes to services, added Lombok library Report: Shared
Loh Sze Ying	Frontend: Linked conference to backend API, linked conference and version to presentation page, filter presentation according to records uploaded, display indication of types of record uploaded, fixed section bug together with Huixian, sort conference list, sort version within a conference, limit upload of files to .csv file (including warning of non-csv file uploaded), warning prompt for deletion of conference, upload box does not appear if input field (title/date) is empty, notifications Report: Shared
Ng Jun Rong Terence	Frontend: Import data mechanism Backend: Exceptions and Advices Report: Shared
Toh Sheng Rong	Frontend: Helped with API call from backend Report: Shared

<b>1. Introduction</b>	<b>4</b>
<b>2. Functional Requirements</b>	<b>5</b>
<b>3. Non-Functional Requirements</b>	<b>8</b>
<b>4. Use Cases</b>	<b>9</b>
<b>5. Design Diagrams</b>	<b>12</b>
5.1 Sequence Diagram	12
5.2 Frontend Architecture Diagram	13
5.3 Backend Architecture Diagram	14
5.4 Entity-Relationship (ER) diagram	15
5.5 Database Tables	16
<b>6. Design Decisions</b>	<b>17</b>
6.1 Frontend	17
6.1.1 Making website easier to use	17
6.1.2 Filter available sections to add to a presentation	17
6.1.3 Display indication of types of record uploaded for each version	18
6.1.4 Editing of navigation bar	19
6.1.5 Removing Calendar	19
6.1.6 Not making use of access control API	19
6.1.7 Sort conference list	20
6.1.8 Displaying versions in descending order	20
6.1.9 Warning prompt when deleting a conference	21
6.1.10 No warning prompt when deleting a section	21
6.2 Backend	22
6.2.1 Keeping vs Removing Presentation entity	22
6.2.2 Prefix of API endpoints	22
<b>7. Use of Design Principles and Patterns</b>	<b>23</b>
7.1 Model, View, Controller	23
<b>8. Other relevant Development Artefacts</b>	<b>24</b>
8.1 Adding of Lombok library	24
8.2 Use of Spring Aspect-Oriented Programming (AOP)	24
8.3 Adding of schema sql file	24
8.4 Adding of @Service annotation	24
8.5 API endpoints	25
8.5.1 Conference	25
8.5.2 Version	26
8.5.3 Records	27
8.5.4 PresentationSections	28

8.5.5 PresentationPermission	29
<b>9. Future enhancements</b>	<b>30</b>
9.1 Spring Security	30
9.2 Spring Hateoas	30
9.3 Error Handling	30
9.4 Add in Calendar feature and making it useful	30
<b>10. Extra Information: Using Travis-CI</b>	<b>31</b>

# 1. Introduction

ChairVisE is a web application created to help conference chairpersons to visualize and share the conference submission statistics. The purpose of this project is to enhance the previous ChairVisE3.0 into a better web application for conference chairpersons to use. Given that some conferences are hosted yearly, we have decided that a conference should contain different versions (i.e. years) for the conference chair to compare and visualize.

In the ChairVisE3.0, we also found it hard to use intuitively with limited feedback on the records uploaded. For example, a Conference chairperson must first import their data, and is unable to see what type of record has been uploaded for each version. Then, a conference chairperson must create a separate presentation in order to make use of the data uploaded. Everytime a Conference chairperson clicks on each presentation created, he/she must always click on the correct version in order to visualize the data correctly. Hence, we decided to make enhancements to ChairVisE3.0 and make it more intuitive to use.

## 2. Functional Requirements

FR #	Description	Priority	Comments
FR-1.1	The application should have a “Conferences” page to show the list of conferences belonging to a user	High	Each conference can have multiple versions, indicated by the year, and each version has its own presentation.
FR-1.2	The application should be able to store multiple conferences for each user in a database	High	<p>Backend will store conferences (denoted by the title) and versions (denoted by the year the conference is held) in a separate table to ensure Persistence and Management of multiple conference data for multiple conferences.</p> <p>Each presentation is tied to a conference and version.</p> <p>For example, “AI” 2020 is a presentation. “AI” 2019 is another different presentation.</p>
FR-1.3	The application should be able to store multiple versions for each conference in a database.	High	<p>For example, in a conference titled “AI”, it can store a 2020 version and a 2019 version. This results in two separate presentations, “AI” 2020 and “AI” 2019.</p> <p>However, both “AI” 2020 and “AI” 2019 are stored under the “AI” conference page.</p>
FR-2.1	Each presentation can store a set of records (author, review and submission) in a database.	High	“AI” 2020 is created.

FR-2.2	The application should replace the existing record of the same type (author, review or submission) if a new record is uploaded for a particular version.	High	For example, “AI” 2020 has an author record already uploaded. If a user uploads another author record under “AI” 2020, the previous author record will be replaced.
FR-3.1	The application should be able to delete a conference.	High	This means deleting a conference and the year it is being held. For example, there are “AI” 2020, “AI” 2019 and “Machine Learning” 2020 presentations. The application should be able to delete “AI”, which includes deleting both 2020 and 2019 of “AI” conference.
FR-3.2	The deletion of a conference will remove the versions and the sets of records associated with it.	High	<p>In the conference list, there are “AI” and “Machine Learning”.</p> <p>“AI” conference consists of a 2020 and a 2019 version.</p> <p>After deleting “AI” from the conference list, the data that was uploaded to “AI” 2020 and “AI” 2019 will be removed.</p>
FR-4	The application should filter and display only the possible visualisations to be used in a presentation based on the records uploaded.	Medium	
FR-5	The application should display an indication on what type of records have been uploaded for each version.	Low	For example, under the “AI” conference, there are 2020 and 2019 versions.

			If a version consists of only “Author record”, there will be an indication “Author record uploaded”. As there are no review and submission records, the display will indicate “No review record found” and “No submission record found”.
FR-6.1	The application should verify the records being uploaded and only accept csv files	Low	
FR-6.2	The application should reject non-csv files with an error message	Low	If the application does not detect the .csv file being uploaded, an error message pop up will be displayed.
FR-7	The application should show a warning pop up when the user deletes a conference.	Low	
FR-8	The application should notify when a user sorts conference list according to ascending/descending order, selects a version, deletes a conference, adds a new section or deletes an existing section.	Low	
FR-9	The application should only allow the user to upload data if the input field of the conference title and the date is not empty.	Low	<p>The date of a conference will be automatically converted to year.</p> <p>If a user deletes the input field after the upload field appears, the upload field will disappear.</p>

### 3. Non-Functional Requirements

Non-functional Requirement	Description	Category
NFR-1	Users should be able to learn to navigate through the application after 10 minutes of interaction with the interface	Usability
NFR-2	The user clicks at the top of the conference list to change the sort sequence according to conference title (ascending or descending)	Usability
NFR-3	Each conference should automatically display the different years of a conference from the latest year to the oldest.	Usability



## 4. Use Cases

System: ChairVisE 4.0

Actor: Conference chairperson

### **Use Case 1: Login**

MSS

1. Conference chairperson clicks on the login button at the top right of the ChairVisE page.
2. Chairwise redirects conference chairperson to Google login page.
3. Conference chairperson logs in to his Google account.
4. Chairwise redirects users to the main page.

Use case ends.

### **Use Case 2: Create a new conference, new version for existing conference or replacing a record of already existing version**

Precondition

1. Conference chairperson is on the home page.

MSS

1. Conference chairperson clicks on the “Add new conference”(or “Import Data”) button.
2. Conference chairperson selects the conference type and record type to upload.
3. Chairwise renders the mapping information and version information sections.
4. Conference chairperson chooses mapping information (header and predefined mapping).
5. Conference chairperson typed the title of the conference and the date of the conference.
6. Conference chairperson clicks on the upload prompt and uploads the file.
7. Chairwise renders the field mappings.
8. Mapping of database field and imported data fields is done, and the user clicks on the “Upload” button.

Use case ends.

Extensions

- 1a. Conference chairperson has previous unmapped data.
  - 1a1. The “Import Data” page will be redirected to the mapping page.

Use case resumes at Step 6.

### **Use Case 3: Delete a conference**

Precondition

1. A conference already exists

2. Conference chairperson is on the home page.

MSS

1. ChairVisE displays a list of conferences.
2. Conference chairperson clicks on a conference.
3. ChairvisE loads the presentation page.
4. Conference chairperson clicks on the “Delete” button on the presentation page.
5. A warning prompt shows up warning the user to delete a conference.
6. Conference chairperson clicks “Yes” and deletes the conference.

Use case ends.

Extensions

- 6a. Conference chairperson clicks on “Cancel” when the warning prompt shows up.
  - 6a1. The conference remains there, and conference and its associating versions and data are not deleted.

Use case ends.

#### **Use Case 4: Select a version to visualize**

Preconditions

1. Conference chairperson is on a conference’s page

MSS

1. Conference chairperson clicks on a conference.
2. ChairVisE displays that conference’s page, with a list of available versions.
3. Conference chairperson chooses the version they want from the list.
4. ChairVisE updates the page to show presentation sections of that version.

Use case ends.

#### **Use Case 5: Add section to a version to visualize**

Preconditions: Conference chairperson is on a conference’s page

MSS

1. ChairVisE displays a list of possible visualizations based on which records have already been uploaded.
2. Conference chairperson selects one of the visualizations and adds it.

Use case ends.

#### **Use Case 6: Edit section of a version to visualize**

Preconditions: Conference chairperson is on a conference’s page, and a section already exists

MSS

1. Conference chairperson scrolls to the section they want to edit, and clicks the edit button.
2. ChairVisE displays the editing interface.
3. Conference chairperson edits what they want.
4. Conference chairperson clicks save to save the changes.
5. ChairVisE displays the updated section on page.

Use case ends.

Extensions

- 4a. Conference chairperson cancels changes.
- 4b. ChairVisE displays the unedited section on the page.

Use case ends.

### **Use Case 7: Delete section of a version**

Preconditions: Conference chairperson is on a conference's page, and a section already exists

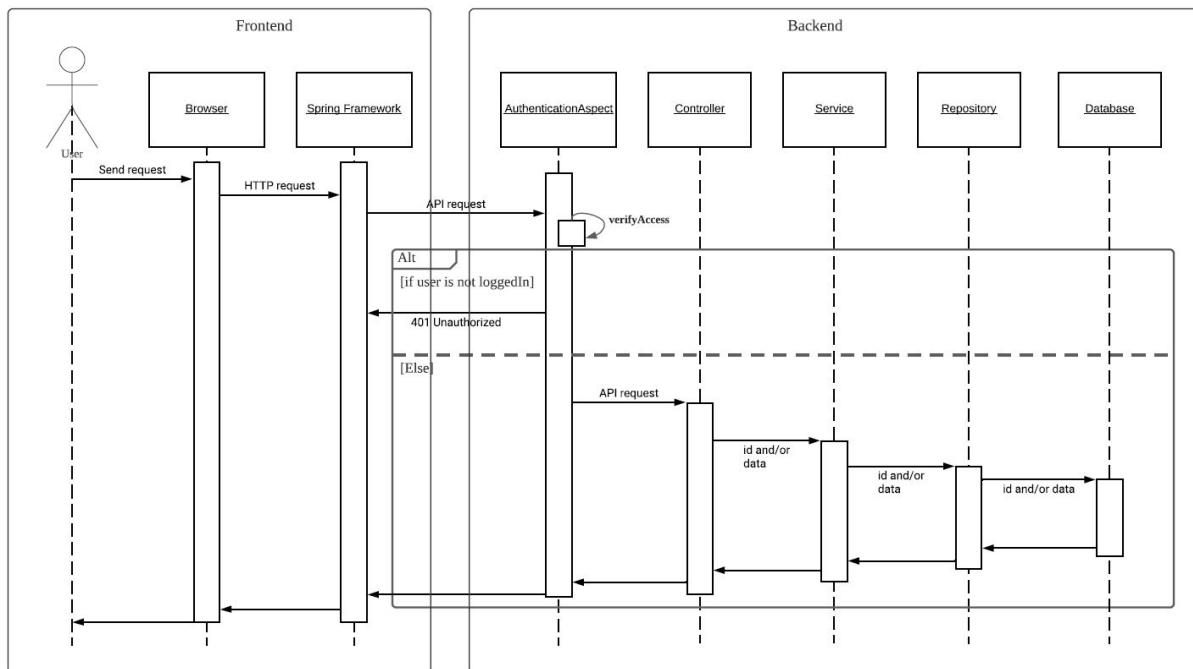
MSS

1. Conference chairperson scrolls to the section they want to delete, and the "Delete" button is clicked.
2. ChairVisE deletes the section.

Use case ends.

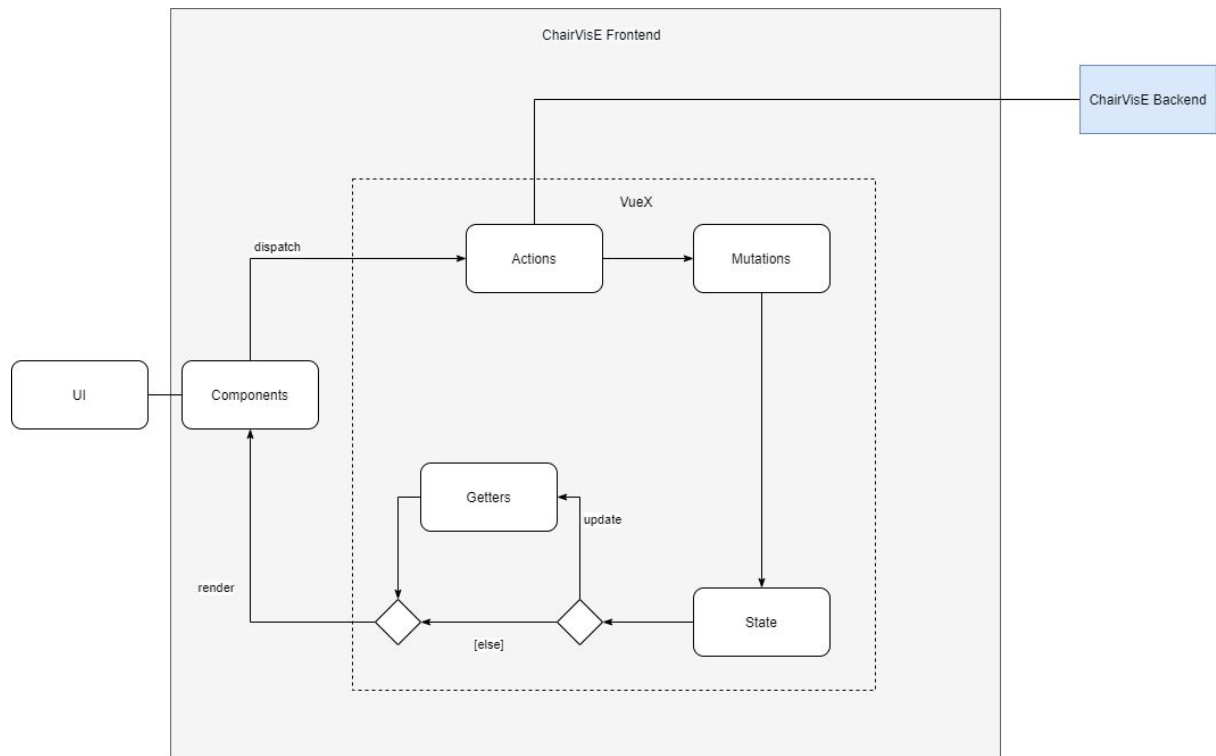
## 5. Design Diagrams

### 5.1 Sequence Diagram



This diagram shows the overall flow of ChairVisE4.0. As ChairVisE4.0 is inherited from ChairVisE3.0, the overall flow did not change much.

## 5.2 Frontend Architecture Diagram



This diagram shows the overall frontend architecture of ChairVisE4.0. As the frontend architecture did not change when we inherited from ChairVisE3.0, the architecture used for frontend in ChairVisE4.0 is similar to ChairVisE3.0.

### State

State is the data stored in the application. As we changed the way conferences are stored in ChairVisE 4.0, changes were also made to the state components.

- modules
  - accessControl.js
  - calendar.js
  - conference.js
  - dataMapping.js
  - dbMetaData.js
  - presentation.js
  - record.js
  - section.js
  - userInfo.js
  - version.js

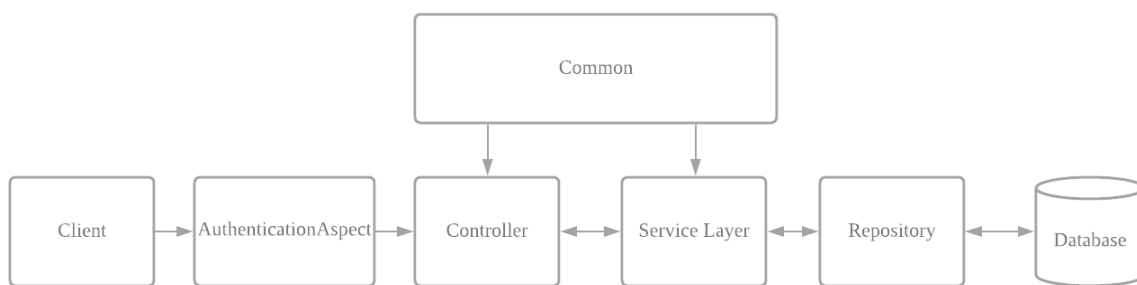
From ChairVisE 3.0, we changed conference.js into calendar.js, and added a new conference.js to represent different conferences (e.g. AI conference, Machine Learning conference etc.). We also added version.js to represent the different versions of a conference (e.g. AI 2019, AI 2020), and record.js to determine which records each presentation has.

## Components

In ChairVise4.0, components are Vue instances that are reusable, and can be used to act as a custom element. For example, the components of the list of conferences page consist of: ConferenceBrief, SelectListPanel, and Manage. Manage stores a list of conferences, while ConferenceBrief stores what details are contained in a Conference, such as title of a conference, and the creator of the conference. SelectListPanel contains the versions of a conference, where a Conference chairperson can select which version to visualize, and add sections to visualize.

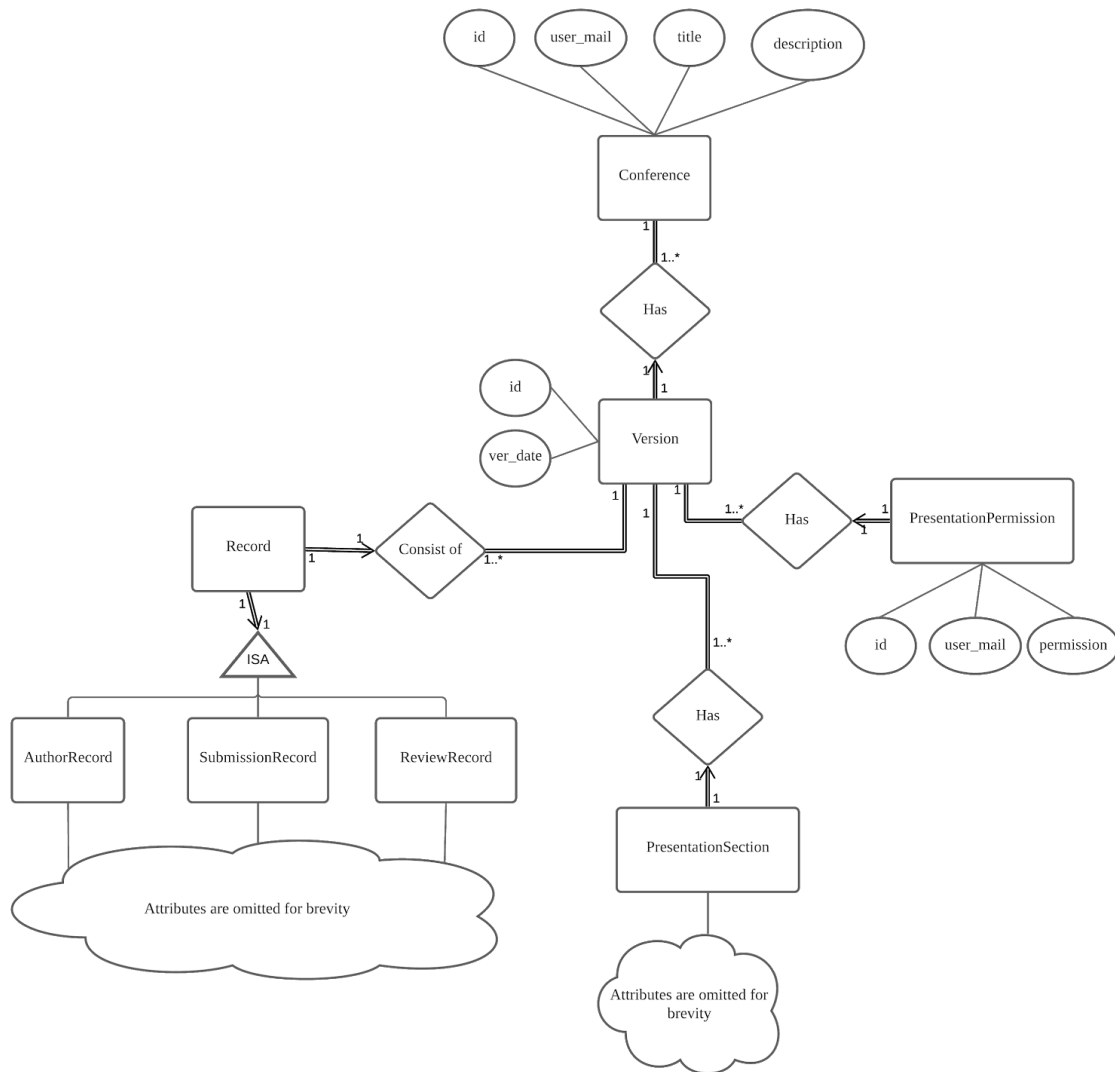
## 5.3 Backend Architecture Diagram

This diagram below shows the architecture diagram of the backend system. All client requests are first filtered by the AuthenticationAspect which checks if the user is authenticated. If the user is authenticated, the request then goes through the controller, service layer, repository and then the database. The common classes are shared in the Controller and Service Layer classes.



## 5.4 Entity-Relationship (ER) diagram

The ER diagram below illustrates the database design on a high-level view. The Version entity is now used to connect the PresentationPermission and PresentationSection entities.



## 5.5 Database Tables

The database tables shown below highlights the specific fields that are used to connect them together.

Conference		
PK	id	int
	title	varchar(255)
	description	varchar(255)
	user_email	varchar(255)

Version		
PK	id	int
	ver_date	varchar(255)
FK	conference_id	int

*Record		
PK	id	int
FK	version_id	int
	Other fields	Type

PresentationPermission		
PK	id	int
	user_email	varchar(255)
FK	version_id	int
	permission	verchar(255)

PresentationSection		
PK	id	int
FK	version_id	int
	Other fields	Type



## 6. Design Decisions

### 6.1 Frontend

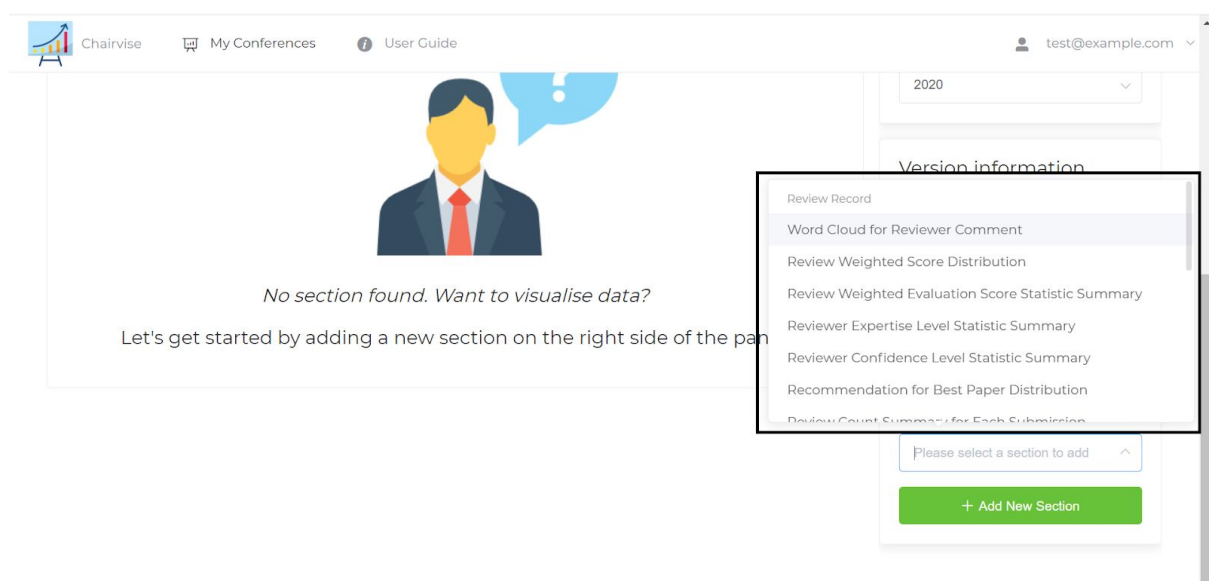
#### 6.1.1 Making website easier to use

As we tried out ChairVisE 3.0, we found it difficult to use intuitively. Hence, we decided to make it easier to use by doing the following.

When a conference chairperson imports data, ChairVisE 4.0 will automatically create a conference and corresponding version for them in the conference list. Hence, every set of records will be linked to a particular conference and version, which means there will not be any redundant data.

#### 6.1.2 Filter available sections to add to a presentation

In ChairVisE 3.0, the presentation page displays the list of all possible visualization sections, even though not all can be created with the data records present. Hence, we implemented a filter to display only the visualization sections that can be created based on which records have been uploaded.



For example, if only the author record has been uploaded, only the “Author Record” will be displayed. This is the same with review and submission records.

If both author records and submission records have been uploaded, “Co-authorship”, “Submission Record”, “Author Record”, and “Author Record + Submission Record” will be displayed in the display list.

If both author records and review records have been uploaded, “Review Record”, “Author Record”, and “Author Record + Review Record” will be displayed in the display list.

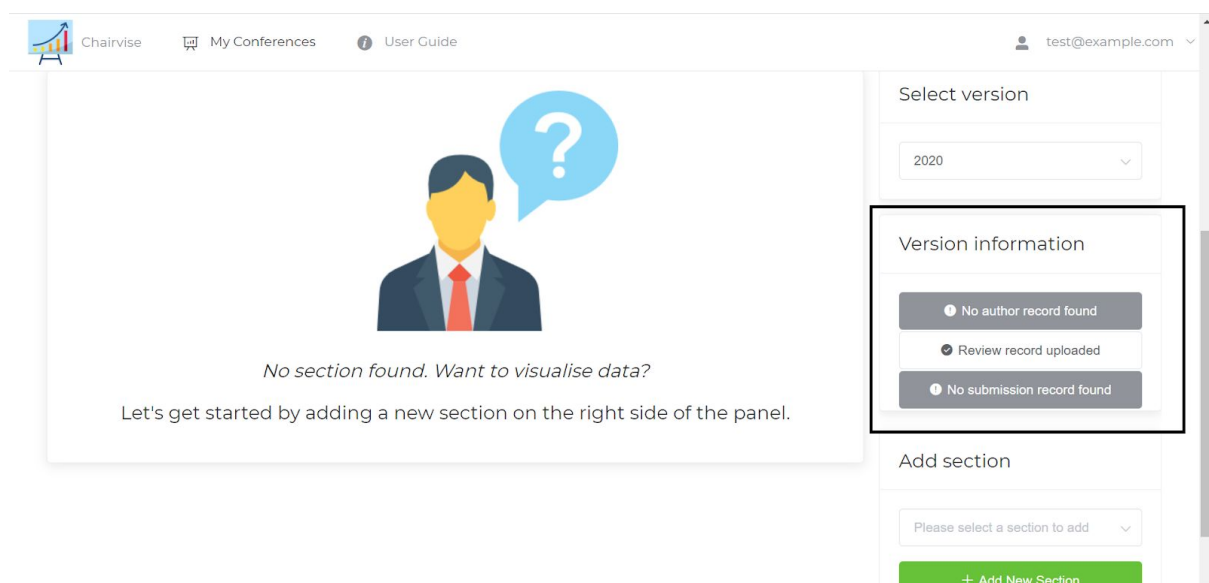
If both review records and submission records have been uploaded, “Submission Record”, “Review Record”, and “Review Record + Submission Record” will be displayed in the display list.

If all the records have been uploaded, the list will display all 7 groups, namely “Co-authorship”, “Submission Record”, “Review Record”, “Author Record”, “Author Record + Submission Record”, “Review Record + Submission Record”, and “Author Record + Review Record”.

### 6.1.3 Display indication of types of record uploaded for each version

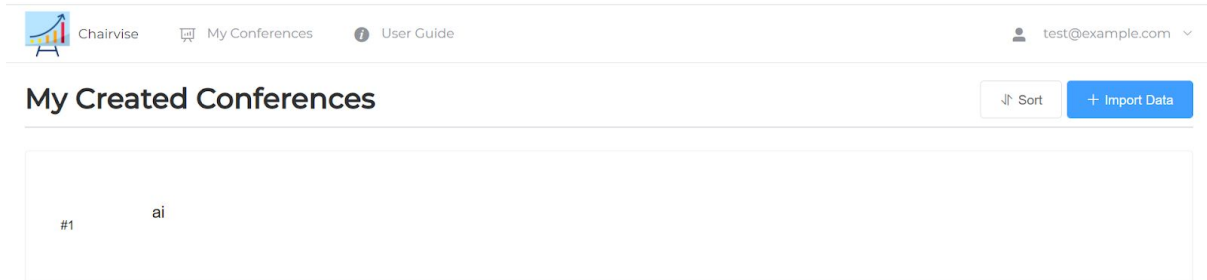
In ChairVisE 3.0, we cannot determine which type of records (author, submission, review) have already been uploaded. Hence in ChairVisE 4.0, we added this feature to give users an indication of what they have already uploaded and what might be missing.

We decided to use grey and white labels to indicate the uploaded status rather than green as green is commonly used in the “Import data” or “Add New Section” buttons where actions are executed. Since this is just meant to be a visual indication, using the colour green might convey a false affordance, and make users mistake it for a button.



#### 6.1.4 Editing of navigation bar

In ChairVisE 3.0, the navigation was done with a drop down menu. To make it more convenient for the users, we decided to add a navigation bar at the top. We also made the home page display the list of conferences so that the users are able to quickly view and access their conferences upon logging in.



#### 6.1.5 Removing Calendar

Originally, we intended to link the date of a conference with the calendar and we wanted to add new features to Calendar, such as notifying an upcoming conference. However, due to time constraints, we did not do so. There is also conflict of API where calendar is using “conference” API, and in 4.0, we are also using “conference” API for a different purpose. Hence, the calendar feature that was originally in ChairVisE3.0 was removed.

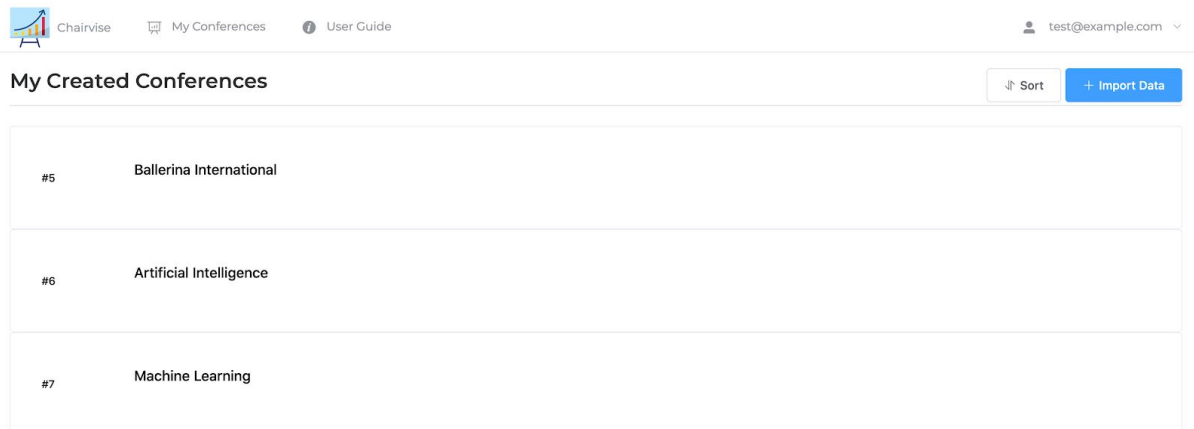
#### 6.1.6 Not making use of access control API

After the backend was refactored for the second time as presentation was not linked, we realized access control was making use of versions api rather than conference api. However, at that time, we were trying to get section visualization to work and we had no time to make further changes to access control API. Hence as a bandaid fix, we checked if the conference creator is the same as the logged in user by email, and do not allow unauthorized users to delete a conference, and add, edit or delete a section. Unfortunately this means a conference chair will not be able to share or download pptx/pdf as the stated features were using access control API.

### 6.1.7 Sort conference list

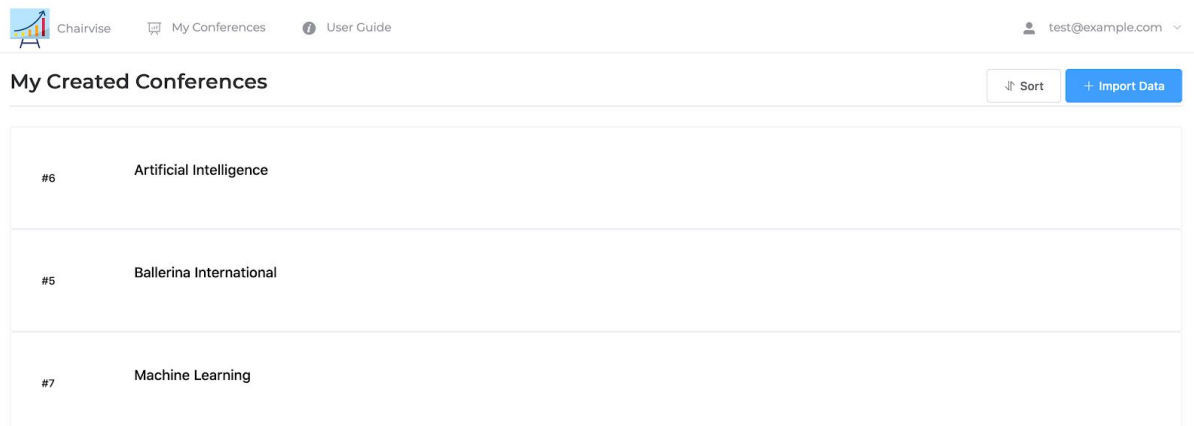
A Conference chairperson may chair multiple conferences, and we want to allow conference chairs to manage their conferences better with the ability to sort their conferences according to the titles of the conferences.

By default, a conference list is sorted according to conference ID, a unique ID that is created with a new conference created.



#5	Ballerina International
#6	Artificial Intelligence
#7	Machine Learning

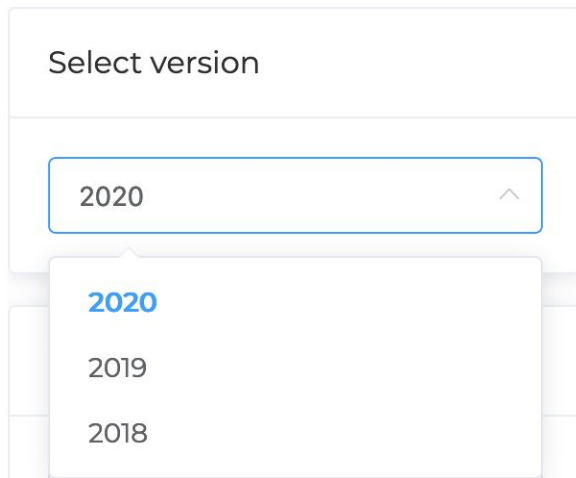
After clicking on the “Sort” button, the conference title can be sorted according to ascending or descending order. The below image shows the conferences title being sorted according to ascending order.



#6	Artificial Intelligence
#5	Ballerina International
#7	Machine Learning

### 6.1.8 Displaying versions in descending order

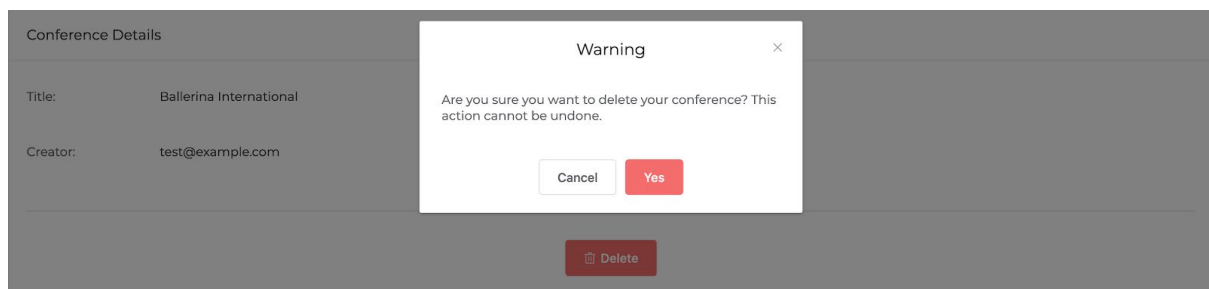
We believe that a conference chair would be more interested in the latest year in a conference, so the versions are automatically sorted in descending order, from latest to oldest.



#### 6.1.9 Warning prompt when deleting a conference

As deleting a conference means deleting all its uploaded versions and including the associating records, a prompt was added when a Conference chairperson wants to delete a conference, as deleting a conference cannot be undone and we do not want a Conference chairperson to accidentally delete a conference.

The below image shows the warning prompt when a Conference chairperson tries to delete a conference by clicking on the “Delete” button on the conference page.



#### 6.1.10 No warning prompt when deleting a section

Unlike 6.1.9 where deleting a conference will delete its associating records uploaded to ChairVisE, deleting a section has no effect on the database. Therefore, there is no warning prompt when a Conference chairperson wants to delete a section within a presentation as a section can easily be added back.

## 6.2 Backend

### 6.2.1 Keeping vs Removing Presentation entity

As we were building the capability to add multiple versions of conference data, we realized that the Version entity can be synonymous with the Presentation entity. Initially, the Presentation entity is connected to PresentationAccessControl and PresentationSection entities. This is because PresentationAccessControl and PresentationSection entities are tied to at most one Presentation entity. Similarly, these former entities can now be associated with the Version entity instead since they are tied to at most one Version entity. Hence, we decided to remove the Presentation entity as it would be a redundant entity.

### 6.2.2 Prefix of API endpoints

With the addition of versions to our conference data, the API endpoints that are closely related with versions are prefixed with /versions/{versionId} rather than conferences/{conferenceId}/versions/{versionId}. This shortens the endpoints and reduces code errors.

## 7. Use of Design Principles and Patterns

### 7.1 Model, View, Controller

In ChairVisE4.0, our team continued to make use of already existing MVC from ChairVisE3.0

Our user interface divides the related program logic into three interconnected elements - model, view and controller. This facilitates and optimizes the implementation of interactive intensive systems.

Models include Conference, Version, PresentationSection, PresentationPermission, AuthorRecord, ReviewRecord, and SubmissionRecord.

View includes ConferenceSection, Home, ImportData, Manage, NewConference, and UserGuide.

Controller includes AnalysisController, AuthInfoController, ConferenceController, DBMetaDataController, PresentationPermissionController, PresentationSectionController, RecordController and VersionController.

The benefits of adopting this design pattern is that it decouples the main UI from the business logic. It also allows multiple developers to collaborate and work together and future modification on individual components, for example, adding a new view and adding a new source of data.

## 8. Other relevant Development Artefacts

### 8.1 Adding of Lombok library

The Lombok library is a Java library that reduces code verbosity, makes code cleaner and allows us to focus more on the business logic. Using a single annotation, it automatically generates setters and getters for us, removing them the need to add them explicitly.

### 8.2 Use of Spring Aspect-Oriented Programming (AOP)

In AOP, aspects are used to enable modularization of concerns such as transaction, logging or security that cut across several types and objects. It makes the code easier to maintain. In our case, we added AOP on the backend for authorization checks. This keeps the security related code in one class rather than over all the controller classes.

### 8.3 Adding of schema sql file

Although Hibernate automatically creates the tables for us, we wanted a more finer-grained control over how the tables and constraints were created. There is also the benefit of knowing the tables that are created in one file rather than searching through all entity classes.

### 8.4 Adding of @Service annotation

Classes containing business logic are added with the @Service annotation to classify them as business logic as part of Spring convention.



## 8.5 API endpoints

### 8.5.1 Conference

Request	Request Body	Example Response
GET /api/conferences		Status: 200 OK Body: [[{"id": 1, "title": "AI", "description": "An Artificial Intelligence conference", "user_email": "test@example.com"}, {"id": 2, "title": "Algorithms", "description": "An Algorithm conference", "user_email": "test@example.com"}]]
GET /api/conferences/1		Status: 200 OK Body: { "id": 1, "title": "AI", "description": "An Artificial Intelligence conference", "user_email": "test@example.com" }
POST /api/conferences	{ "title": "AI", "description": "An Artificial Intelligence conference" }	Status: 201 created Body: { "id": 1, "title": "AI", "description": "An Artificial Intelligence conference", "user_email": "test@example.com" }
PUT /api/conferences/1	{ "title": "AI", "description": "An Artificial Intelligence conference" }	Status: 200 OK Body: { "id": 1, "title": "AI", "description": "An Artificial Intelligence conference", "user_email": "test@example.com" }
DELETE /api/conferences/1		Status: 204 No Content

### 8.5.2 Version

Request	Request Body	Example Response
GET /api/conference/{confereceld}/versions		Status: 200 OK Body: [[ {"id": 1, "date": "2011-07-14"}, ], [ {"id": 2, "date": "2013-04-21"} ]]
GET /api/conference/{confereceld}/versions/1		Status: 200 OK Body: { "id": 1, "date": "2011-07-14" }
POST /api/conference/{confereceld}/versions	{ "date": "2011-07-14" }	Status: 201 created Body: { "id": 1, "date": "2011-07-14" }
PUT /api/conference/{confereceld}/versions/1	{ "date": "2011-07-14" }	Status: 200 OK Body: { "id": 1, "date": "2011-07-14" }
DELETE /api/conference/{confereceld}/versions/1		Status: 204 No Content

### 8.5.3 Records

Request	Request Body	Example Response
GET /api/versions/{versionId}  (Show whether each record exists or not) (Returns the id of the record if it exists, 0 if doesn't exists)		Status: 200 OK Body: <pre>{   "authorRecord": false,   "submissionRecord": true,   "reviewRecord": false }</pre>
POST /api/versions/{versionId}/authorRecord  (Add a author record)	<pre>{   submission_id,   first_name,   last_name,   email,   country,   organization,   web_page,   person_id,   is_corresponding }</pre>	Status: 201 created Body: <pre>{   same as request body + versionId }</pre>
POST /api/versions/{versionId}/submissionRecord  (Add a submission record)	<pre>{   s_id,   track_id,   track_name,   title,   submitted,   last_updated,   keywords,   decision,   notified,   reviews_sent,   abstract }</pre>	Status: 201 created Body: <pre>{   same as request body + versionId }</pre>
POST /api/versions/{versionId}/reviewRecord  (Add a review record)	<pre>{   review_id,   submission_id,   review_assignment_num,   reviewer_name, }</pre>	Status: 201 created Body: <pre>{   same as request body + versionId }</pre>
	<pre>{   field,   review_comment,   overall_eval_score,   review_submission_date,   review_submission_time,   best_paper_recommendation }</pre>	

### 8.5.4 PresentationSections

Request	Request Body	Example Response
GET /api/versions/{versionId}/sections		Status: 200 OK Body: [[ "Id": 1, versionId: 1, title, description, type, dataSet, selections, involvedRecords, filters, joiners, groupers, sorters, extraData ]]
POST /api/versions/{versionId}/sections	{ title, description, type, dataSet, selections, involvedRecords, filters, joiners, groupers, sorters, extraData }	Status: 201 created Body: { same as request body + versionId + id }
PUT /api/versions/{versionId}/sections/{sectionId}	{ title, description, type, dataSet, selections, involvedRecords, filters, joiners, groupers, sorters, extraData }	Status: 200 OK Body: { same as request body + versionId + id }
	groupers, sorters, extraData }	
DELETE /api/versions/{versionId}/sections/{sectionId}		Status: 204 No Content

### 8.5.5 PresentationPermission

Request	Request Body	Example Response
GET /api/versions/1/permissions		Status: 200 OK Body: [[ {"id": 1, "userEmail": "test@example.com", "versionId": 1, "permission": "CAN_WRITE" }]
POST /api/versions/1/permissions	{ "userEmail": "test@example.com", "permission": "CAN_WRITE" }	Status: 201 created Body: { {"id": 1, "userEmail": "test@example.com", "versionId": 1, "permission": "CAN_WRITE" }
PUT /api/versions/1/permissions/ 1	{ "userEmail": "test@example.com", "permission": "CAN_READ" }	Status: 200 OK Body: { {"id": 1, "userEmail": "test@example.com", "versionId": 1, "permission": "CAN_READ" }
DELETE /api/versions/1/permissions/ 1		Status: 204 No Content

## 9. Future enhancements

### 9.1 Spring Security

Spring Security is a Java/Java EE framework that provides authentication, authorization and other security features for enterprise applications. It is the de-facto standard for securing Spring-based applications. It can be used to replace the existing authentication and authorization system.

### 9.2 Spring HATEOAS

Spring HATEOAS provides some APIs to ease creating REST representations that follow the HATEOAS principle when working with Spring and especially Spring MVC. The core problem it tries to address is link creation and representation assembly.

### 9.3 Error Handling

Provide more helpful error responses with error messages for API endpoints.

### 9.4 Add in Calendar feature and making it useful

As the versions table stores date rather than year, it would be good to link the date of the conference and its version created into the calendar, and show notification of upcoming conference.












## 10. Extra information: Using Travis-CI

Up to PR #86 as shown from

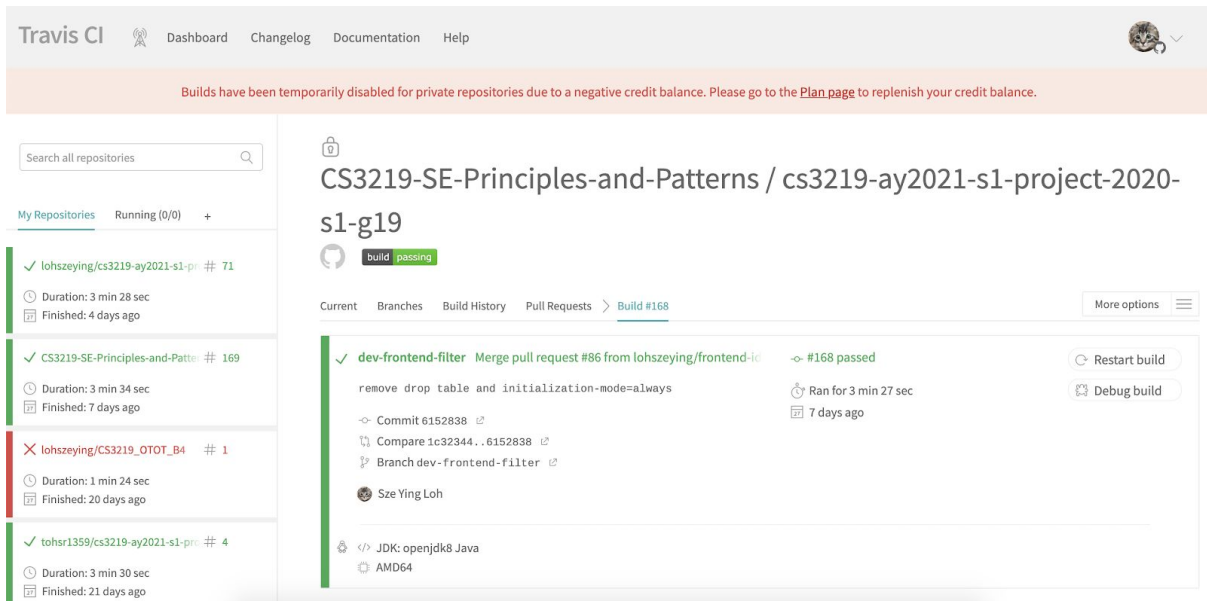
<https://github.com/CS3219-SE-Principles-and-Patterns/cs3219-ay2021-s1-project-2020-s1-g19/pulls?q=is%3Apr+is%3Aclosed> , Travis has always worked for our team. Our team uses creating a pull

request workflow via Github group repository before merging to the group repository branch, and

always had Travis-CI automatically built for us. Our group will only merge PR if Travis build passes.

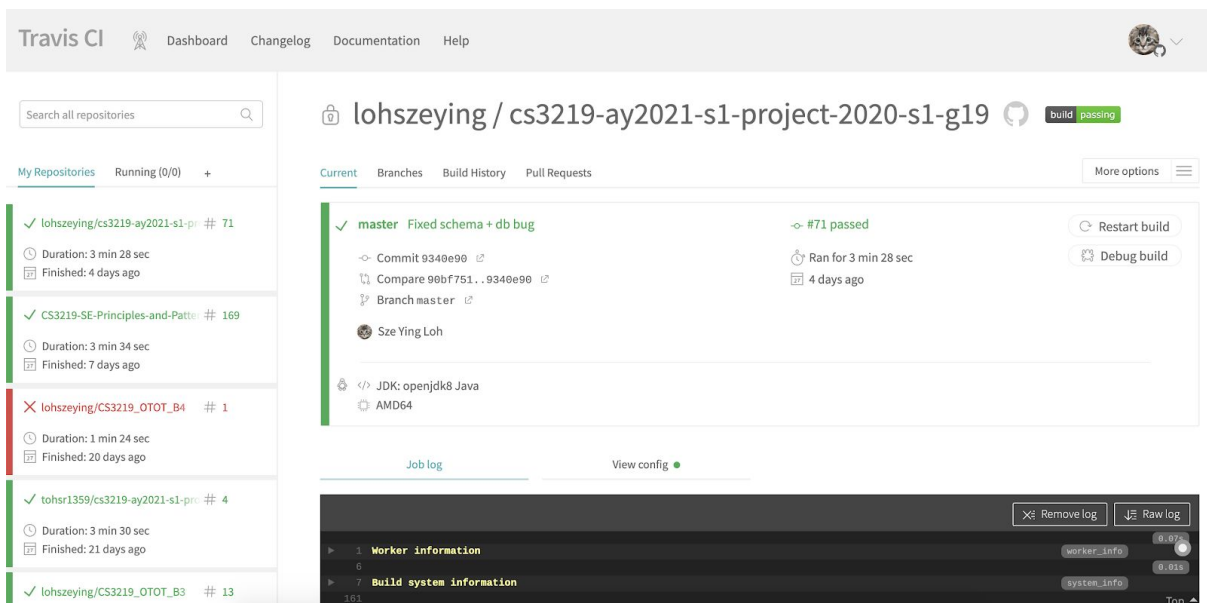
<input type="checkbox"/>	 2 Open	<input checked="" type="checkbox"/> 56 Closed	Author ▾	Label ▾	Projects ▾	Mi
<input type="checkbox"/>		<b>Fixed schema + db bug</b>				
		#88 by lohszeying was merged 3 days ago				
<input type="checkbox"/>		<b>Fixed schema + db bug</b>				
		#87 by lohszeying was closed 4 days ago				
<input type="checkbox"/>		<b>remove drop table and initialization-mode=always</b> ✓				
		#86 by lohszeying was merged 7 days ago				
<input type="checkbox"/>		<b>Integration into master branch</b> ✓				
		#85 by lohszeying was merged 4 days ago • Approved				
<input type="checkbox"/>		<b>Frontend: Updated website guide</b> ✓				
		#84 by lohszeying was merged 8 days ago				
<input type="checkbox"/>		<b>Merge branch from origin filter to dev filter</b> ✓				
		#83 by lohszeying was merged 8 days ago				
<input type="checkbox"/>		<b>Frontend: Filter sections according to type of record uploaded, visual indicator for records uploaded</b> ✓				
		#82 by lohszeying was merged 8 days ago				
<input type="checkbox"/>		<b>Frontend idea1 filter</b> ✓				
		#81 by lohszeying was merged 8 days ago				
<input type="checkbox"/>		<b>Frontend: Only authorized user can delete/edit/add section, added notifications</b> ✓				
		#80 by lohszeying was merged 8 days ago				
<input type="checkbox"/>		<b>Commented log in backend</b> ✓				
		#79 by lohszeying was merged 10 days ago				

In the screenshot above, this shows that travis had worked for our group, until #87 onwards.



The above screenshot shows Travis passing on PR #86, which the PR is titled “remove drop table and initialization-mode=always”. This PR is the last PR where Travis build worked for a private group repository.

After PR #86, Travis could not be automatically built anymore for our team’s private group repository due to negative credit balance. Following Jonathan’s advice, PR #88 “Fixed schema + db bug” was merged despite Travis not building for our team’s private group repository.



As a proof a screenshot, the above screenshot shows that the commit for PR #88 passed Travis build for a personal private repository.