

CS3219

ChairVisE3.0

Group 17

Codee Ong Wen Ci	A0157816W
Jeffrey Goh Keng Wee	A0149850X
Seet Ying Hui Renee	A0157366W
Shradheya Thakre	A0161476B

ChairVisE3.0 Code Repository URL:

<https://github.com/CS3219-SE-Principles-and-Patterns/chairvise3-0-2019-team-17>

Application URL: <https://cs3219-1920-team17.appspot.com>

Introduction	5
Existing version : ChairVisE2.0	6
2.1 Non Functional Requirements	6
2.1.1 Performance Requirements	6
2.1.2 Security Requirements	6
2.1.3 Software Quality Attributes	6
2.1.4 Software Constraints	6
2.2 Functional Requirements	6
2.2.2 User Stories	7
2.2.3 Use Case	7
2.3 Design	9
New Version: ChairVise 3.0	10
3.1 Non Functional Requirements	10
3.1.1 Security Requirements	10
3.1.2 Software Quality Attributes	10
3.2 Functional Requirements	10
3.2.2 User Stories	11
3.2.2 Use Case	11
3.3 Design	13
3.3.1 Enhance existing visualizations for easier understanding	14
3.3.1.1 Download the visualizations in a presentation format PDF	14
3.3.1.2 Download time & date to the PDF	16
3.3.2 Include new visualisations to provide new insights	17
3.3.2.1 Section for Overview of Reviews, Submissions and Authors	17
3.3.3 Improve general user interface and experience	19
3.3.3.1 Shared Presentation list	19
3.3.3.2 Delete Pop-up Before Deletion of Presentation/Section	21
3.3.3.3 Ability to Reorder Sections	23
3.3.3.4 Prevent duplicate presentation names	25
3.3.3.5 Search Bar to Filter by name	27
3.3.3.6 Delete all sections that do not have data	28
3.3.4 Persistence and Management of multiple conferences data	29
3.3.4.1 Persist Data of Multiple Conferences	29
3.3.4.2 Visualization Generated from Different Conferences	32
3.3.5 Co-authorship Queries	34
3.3.5.1 Section Network Graph for Co-Authorship	34
3.3.5.2 Inter-Country Co-Author Submissions Bar Chart Visualization	36

3.3.6 Flexible data schemes	37
3.3.7 Technical Improvements	37
3.3.7.1 ESLint Rule Configurations	37
3.3.7.2 Modularization and Configurations	38
3.3.7.3 Security Improvements	38
3.3.7.3 Other Improvements	38
Suggestions and Improvements	39
4.1 Access Control With Different Type of Privileges	39
4.2 Improve Data Upload Interface	39
4.3 Generic CSV support	39
4.4 REST API Communication Pattern and Error Handling	40
4.5 Sharing presentations and Sharing Conference Data	40
Development process	40
5.1 Set-up and Prioritization of Tasks	40
5.2 Weekly Sprints	41
5.3 Development Workflow	42
Appendix A - Documentation	43
Developer Documentation	43
Vue, Vuex4 and Spring Boot5	43
Google App Engine	43
Google Cloud SQL	43
Continuous Development (CD)	43
Design Diagrams	44
Overall Architecture	44
Sequence Diagram	45
Backend Architecture	46
Frontend Architecture	52
Setting Up Development Environment	59
Step 1: Get a Copy	59
Step 2: Install necessary tools and languages	59
Step 3: Setup Project specific settings and configuration	60
Step 4: Test the project	61
Step 5: Deploy to GCP (Optional)	61
Appendix B - A Peek in the Data	63
Appendix C - Join Schematics	68
Appendix D FAQs	69

Team Info and Individual Contributions

Name	Matriculation number	Type	Contributions
Codee Ong Wen Ci	A0157816W	Technical	Feature implementation: Give users a warning pop-up before deletion of presentations and sections. Implementation includes front-end Vue test for new DeleteModal component.
			Feature implementation: Allow users to reorder the visualizations without having to add and remove them. Implementation includes new Java tests for new API end-point.
			Feature implementation: Bar chart showing the number of papers submitted for top country co-authorships (e.g., China- USA, India-China, etc.)
			Upgrade Tests: Implemented front and back-end tests for new features
		Non-technical	Regularly did code reviews and user testing for teammates' features.
			When there are bugs spotted, I would notify group members either via group chat or by creating an issue on Github.
			Documentation
Jeffrey Goh Keng Wee		Technical	Feature implementation: Prevent users from adding presentations with duplicate names, or editing a presentation to have a duplicate name
			Feature implementation: Added a search bar to the presentation list to allow the user to filter presentations by name
			Feature implementation: Added a button to allow users to remove sections without data
		Non-technical	Regularly did code reviews and user testing for teammates' features.

			When there are bugs spotted, I would notify group members either via group chat or by creating an issue on Github.
			Documentation
Seet Ying Hui Renee	A0157355W	Technical	Feature Implementation: Add list of shared presentation for the collaborators so that they can access the presentation without having to type the shared URL
			Feature Implementation: Show simple data such as the total number of reviews, authors and submissions (Summary)
			Feature Implementation: PDF should show the time and date of when the analysis was generated
			Feature Implementation: Add Graph network section of organizations and countries of co-authors
			Testing enhancement : Add basic backend test cases for new features
		Non-technical	When there are bugs spotted, I would notify group members either via group chat or by creating an issue on Github.
			Regularly did code reviews and user testing for teammates' features.
			Documentation
Shradheya Thakre	A0161476B	Technical	Feature Implementation: Download the analysis in a presentation format
			Feature Implementation: Add ability to store data for multiple conferences
			Feature Implementation: Increase backend test

			coverage from 85% to 88% covering critical paths and endpoint tests and write sanity tests
			Feature Implementation: Add ability to generate presentations/reports from data of different conferences
			Feature Implementation: Few bug fixes and front-end improvements
			Developer Maintenance: Includes security updates for dependencies, code quality metrics and designs, initial developmental setup and deploy app to cloud, continuous deployment(CI & CD) and reduce technical debt.
		Non-technical	When there are bugs spotted, I would notify group members either via group chat or by creating an issue on Github.
			Designing developmental process and tracking project progress and updates
			Developer Documentation

1. Introduction

This project is designed to enable conference program chairpersons to visualize and share conference submission statistics. By parsing information in different formats, we aim to assist users to obtain the most value out of the information uploaded. We also support sharing and exporting of such visualization.

This report is about the webapp built over the past 8 weeks. We have improved the functionality of the application by adding new features and improving existing ones while applying software engineering principles and design patterns along the way.

2. Existing version : ChairVisE2.0

2.1 Non Functional Requirements

2.1.1 Performance Requirements

1. Pages should not take more than 5 seconds to load
2. Visualisations should generate within 5 seconds.
3. Downloading of visualisations PDF should be within 5 seconds

2.1.2 Security Requirements

1. Users must login before they can see their presentations.
2. Users must be verified before editing any presentations and visualisation.

2.1.3 Software Quality Attributes

1. Reliability
 - a. Testing is done to ensure reliability. As many possible test conditions that can be tested should be done via test automation
2. Reusability
 - a. The components should have high cohesion and low coupling, so they can be reused easily.
3. Extensibility
 - a. Developers with an understanding of the frameworks and tools used in building the system should be able to extend its functionalities with ease.

2.1.4 Software Constraints

1. Data files uploaded have to be formatted.
2. Only .csv files are supported.
3. The application only works with an active internet connection.

2.2 Functional Requirements

2.2.1 Requirements

FR 2.2.1.1 The user can create presentations.

FR 2.2.1.2 The user can collaborate with other users on presentations.

FR 2.2.1.3 The user can import conference data onto the program.

FR 2.2.1.4 The user can generate analysis to see a word cloud for all submissions keywords.

FR 2.2.1.5 The user can generate analysis to see submission ranks of authors.

FR 2.2.1.6 The user can generate analysis to see submission ranks of countries.

FR 2.2.1.7 The user can generate analysis to see accepted submission organization rank.

FR 2.2.1.8 The user can generate analysis to see review score distribution amongst submissions.

FR 2.2.1.9 The user can generate analysis to see a review weighted evaluation score statistic summary.

FR 2.2.1.10 The user can generate analysis to see a reviewer expertise level statistic summary.

2.2.2 User Stories

1. User Story 1

As a user, I would want to upload data files and generate visualisations of the data

2. User Story 2

As a user, I would want to be able to download the visualisations into a PDF document.

3. User Story 3

As a user, I want to be able to edit my visualisations.

4. User Story 4

As a user, I want to share my presentation of visualisations with others.

2.2.3 Use Case

1. Use Case 1: Uploading of data File

a. Main Scenario

- i. User uploads a .csv file on the Import Data tab
- ii. ChairVisE displays the mappings for that .csv file and prompts the user for confirmation
- iii. Use Case ends

b. Alternative Scenario

- i. User uploads multiple .csv files at once
- ii. ChairVisE chooses one of the .csv files and displays the mappings for that .csv file and prompts the user for confirmation
- iii. Use Case ends

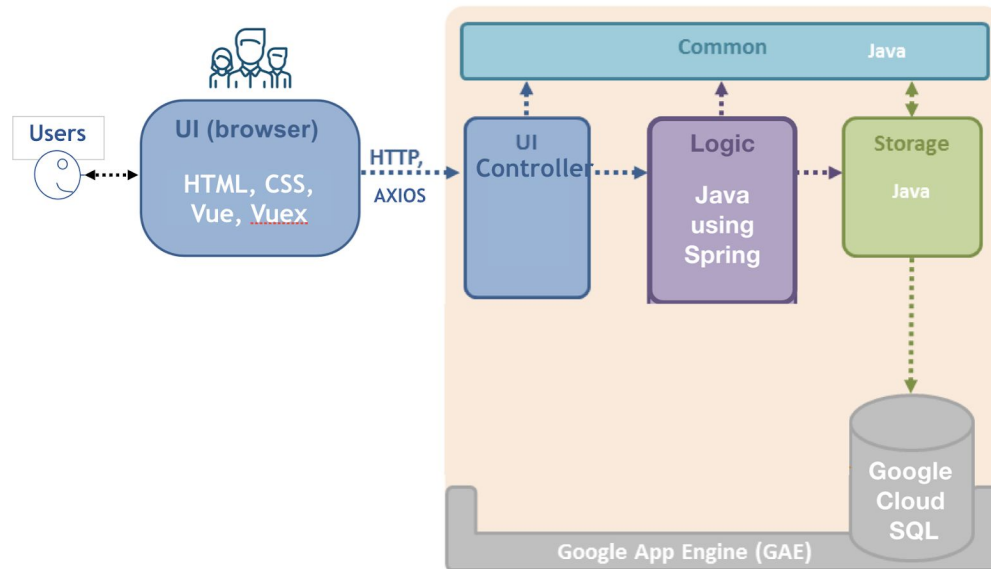
2. Use Case 2 : Adding a presentation

- i. User selects New on the Presentation List on the Analyze tab
- ii. User fills in the Name and Description of the presentation they want to create and clicks the Save button

- iii. Use Case ends
- 3. Use Case 3: Editing a presentation
 - i. User selects the presentation they want to edit on the Presentation List on the Analyze tab
 - ii. User fills in the new name and description of the presentation and clicks the Save button
 - iii. Use Case ends
- 4. Use Case 4: Sharing a presentation
 - a. Main Scenario
 - i. User selects the presentation they want to share on the Presentation List on the Analyze tab
 - ii. User fills in the email address of the user they want to share it with and the permissions they should have
 - iii. User clicks the Add button to allow the user access to the presentation
 - iv. The other user obtains the presentation's URL through external means
 - v. Use Case ends
 - b. Alternative Scenario
 - i. User selects the presentation they want to share on the Presentation List on the Analyze tab
 - ii. User sets the appropriate permissions under the Shareable Link section
 - iii. The other user obtains the presentation's URL through external means
 - iv. Use Case ends
- 5. Use Case 5: Downloading a presentation
 - i. User selects the presentation they want to download on the Presentation List on the Analyze tab
 - ii. User clicks on the Download as PDF button
 - iii. Use Case ends
- 6. Use Case 6: Deleting a presentation
 - i. User selects the presentation they want to delete on the Presentation List on the Analyze tab
 - ii. User clicks on the Delete button
 - iii. Use Case ends

2.3 Design

Overall Architecture



The application is designed with a layered design pattern. **Single Responsibility Principle (SRP)** is applied here. Each layer/classes has its own responsibility.

- **UI:** The component consists of API controllers and WebPage controllers. API controllers are responsible for handling API calls by the frontend. WebPage controllers are responsible for serving static production Vue files.
- **Logic:** The main logic of the application is in Java using the Spring framework.
- **Storage:** The storage layer of the application uses the persistence framework provided by Google App Engine, using MySQL 5.6.
- **Common:** The Common component contains utility code (data transfer objects, helper classes, etc.) used across the application.

More detailed explanation can be found in the developer documentation in Appendix A

3. New Version: ChairVise 3.0

3.1 Non Functional Requirements

Most of the requirements are the same. Except the following:

3.1.1 Security Requirements

1. Ensure all third-party dependencies do not have moderate or higher security issue according to npm.js data

3.1.2 Software Quality Attributes

1. Backend testing coverage should be greater than 88%.
2. Software should be maintainable and follow the code style rules implemented through the lint tools.

3.2 Functional Requirements

3.2.1 Requirements

FR 3.2.1.1 Enhance existing visualizations

1. There should be a downloadable PDF which should contain only one chart per page making it render as a presentation.

FR 3.2.1.2 Add more Visualizations

1. The user can generate overviews for Review, Submission and Author record.

FR 3.2.1.3 Enhance User Interface

1. There should be a warning before deletion of presentations or sections.
2. The user should be allowed to reorder the visualizations without having to add and remove them.
3. The user can easily see and access all presentation that were shared with them.
4. The user can differentiate the downloadable PDF from the download time and date stated in the document.
5. ChairVisE should prevent users from creating presentations with duplicate names.
6. The user can filter presentations he has access to by name.
7. The user can easily remove all sections that do not have data.

FR 3.2.1.4 Persistence of multiple conference data

1. The user should be able to store data for multiple conferences at the same time
2. The user should be able to create a presentation with analysis graphics from different conference data

FR 3.2.1.5 Co-authorship Queries

1. The user should be able to generate a network of countries based on the co-authorship of submissions.
2. The user should be able to generate a network of organisations based on the co-authorship of submissions.
3. The user should be able to view the number of papers collaborated on for top few country groups (e.g., China-USA, India-China-Germany, etc.)

FR 3.2.1.6 Technical Enhancements

1. Technical debt is reduced by updating all deprecated code.
2. Update npm libraries to remove moderate and high level security issues.
3. Add backend tests for critical path coverage.
4. Update code quality including static analysis rules and architectural design metrics.

3.2.2 User Stories

User Stories in ChairVise 2.0 is still relevant, the following are new user stories for version.

1. User Story 1
As a user, I want easy access to all the presentations that have been shared with me.
2. User Story 2
As a user, I want to be able to reorder the sections in my presentation after having selected my sections.
3. User Story 3
As a user, I would not want to lose a presentation or section the moment I first click the delete button.

3.2.2 Use Case

1. Use Case 1 : Adding sections in a presentation

- i. User selects the presentation they want to modify
 - ii. User selects the conference they want to create the section from
 - iii. User selects the type of section they want to create
 - iv. User confirms the selections
 - v. Use Case ends
- 2. Use Case 2: Reordering sections in a presentation
 - i. User selects the presentation they want to modify
 - ii. User finds the section they want to adjust the position of
 - iii. User shifts the section up or down as they require
 - iv. Use Case ends
- 3. Use Case 3: Deleting sections in a presentation
 - a. Main Scenario
 - i. User selects the presentation they want to modify
 - ii. User finds the section they want to delete
 - iii. User clicks on the delete button to remove the section
 - iv. Use Case ends
 - b. Alternative Scenario
 - i. User selects the presentation they want to modify
 - ii. User uses the Delete Sections without data button
 - iii. Use case ends
- 4. Use Case 4: Searching for a presentation
 - i. User types in the name of the presentation they would like to search for
 - ii. User selects the presentation they are looking for from the filtered list
 - iii. Use Case ends
- 5. Use Case 5: Deleting a presentation
 - i. User selects the presentation they want to delete
 - ii. User clicks the delete button
 - iii. User confirms their decision on the popup window
 - iv. Use Case ends
- 6. Use Case 6: Accessing a presentation shared with the user
 - i. Another user sets up permissions to allow the user access to their presentation
 - ii. User selects the presentation shared with them on the list of shared presentations
 - iii. Use Case ends

3.3 Design

Overall Architecture

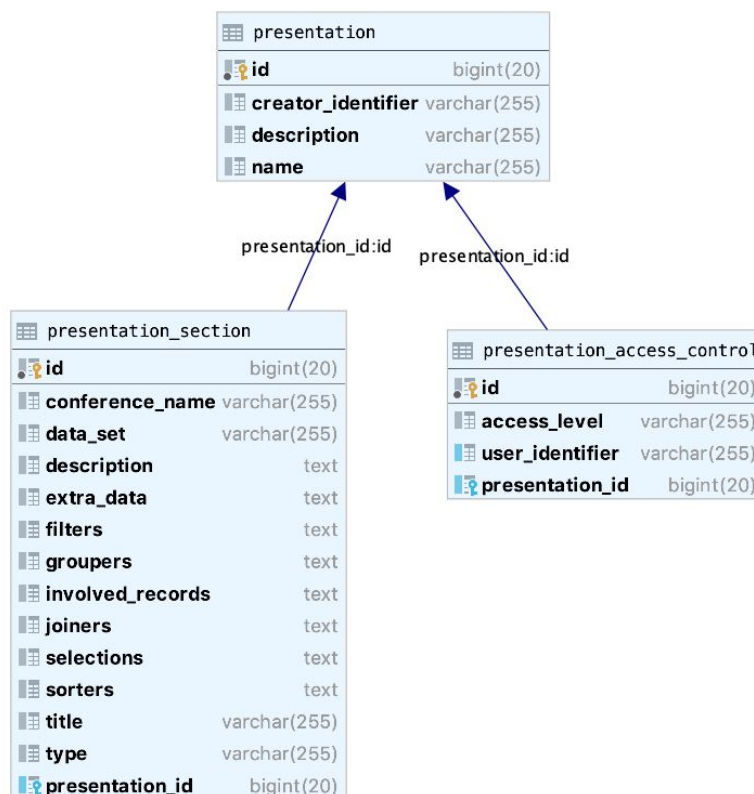
The overall architecture of ChairVise2.0 was mostly being preserved in ChairVise3.0, apart from a few new API endpoints being created and some new fields being added to the existing data models.

Database Schema

MySQL relational database is used to store the data used by backend to query and display data. The database schema can be divided into two parts:

1. Presentations

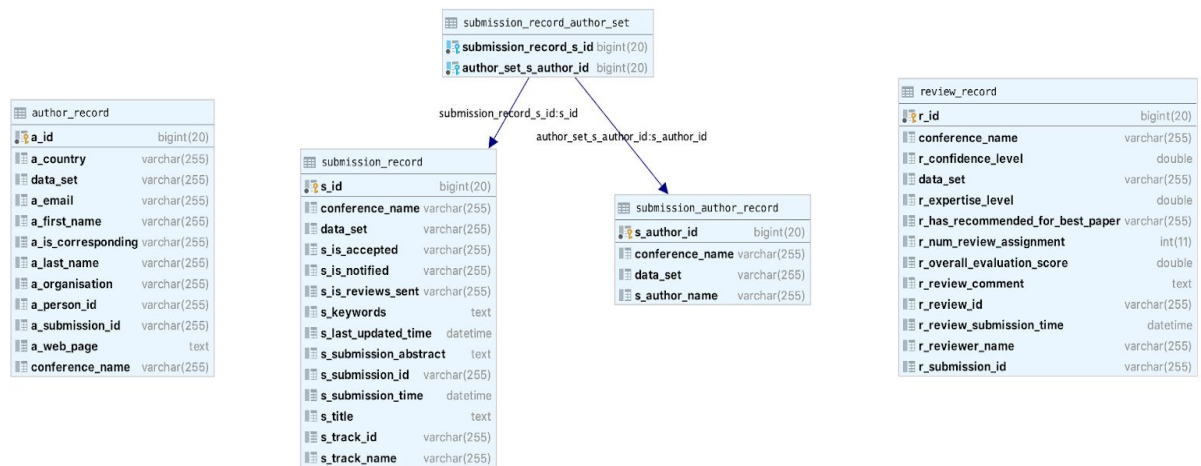
This mainly consists of presentation, presentation sections and data related to access control for presentation sharing between users. There are a couple of foreign key - primary key pairing used for matching data for easy lookup.



2. Record Entities

These entities mainly store the metadata related to conferences which are uploaded by users. A user can store data for multiple conferences.

The three entities are **submission**, **author**, and **review** records. Since submission and author records for the conference files usually have reference to each other, they are joined in the schema for having correct representation of data for easy analysis of data and faster querying.

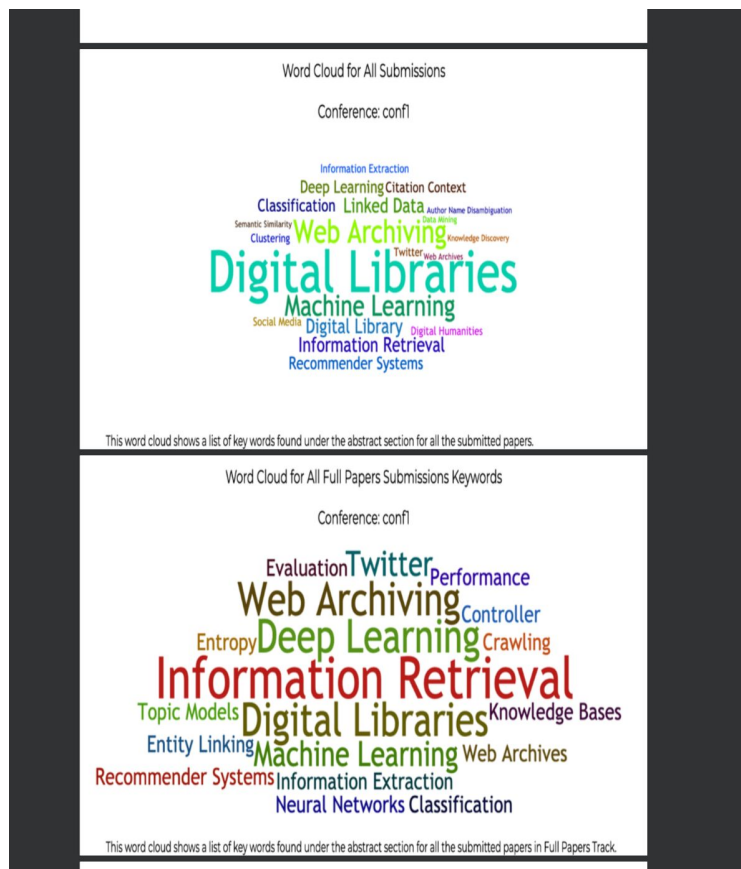


More detailed explanation can be found in the developer documentation in Appendix A

3.3.1 Enhance existing visualizations for easier understanding

3.3.1.1 Download the visualizations in a presentation format PDF

The common use case imagined for downloading the presentation is to be able to easily show and present to an audience. The given download is more focussed on formatting for keeping it as an offline record or to send a pdf copy. With the new implementation of being able to download the analysis presentation in a Presentation Format, the user can use it directly to present the given downloaded PDF to any conference.



New UI of the downloaded presentation

3.3.1.1.1 List of Changes

1. Modify the logic for deciding which page gets which presentation section
2. Modify the orientation and margins for the presentation all of which were done by the function calls of js2pdf library

3.3.1.1.2 Logic for Changes

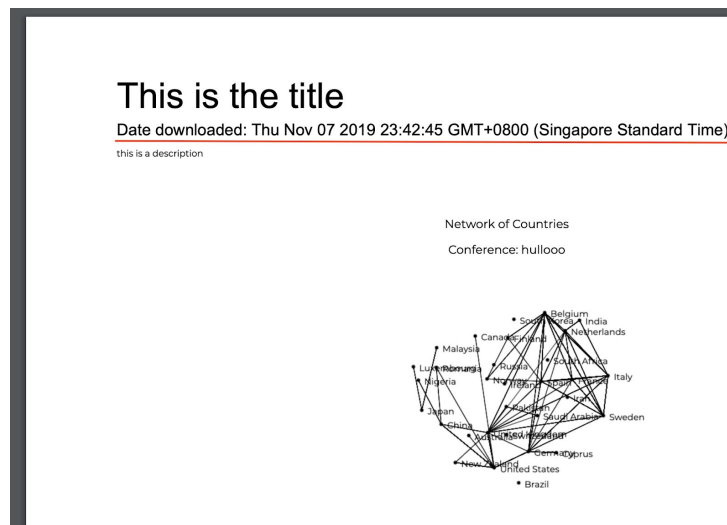
1. When the `Download as Presentation` button is clicked, the js2pdf library does the work and converts the presentation section into images which fit into a landscape size sheet
2. Then each presentation section is assigned a page based on index and hence it forms each slide having one analysis which can be presented easily

3.3.1.1.3 Design Consideration

1. Since the normal download and new download had a lot of similar reusable code, it had to be ensured that there is high cohesion and it can be extended properly and only those properties which are exclusive to each type as separated while trying to reuse most of the code.
2. It had to be considered that if the product decisions change and more extra types of download or change in format is requested, it wouldn't require rewrite of code and can be easily integrated into the current functionality. The **Maintainability** aspect of NFR had to be considered when coding out the extended feature.

3.3.1.2 Download time & date to the PDF

Users are allowed to download PDF version of their presentation. Since there could be data changes since the download of the PDF, we decided to add the download time & date to the PDF so that users can easily compare the presentation changes overtime.



Screenshot of sample PDF where date of download is shown

3.3.3.2.1 List of changes

1. Add today's time and date to PDF before download.

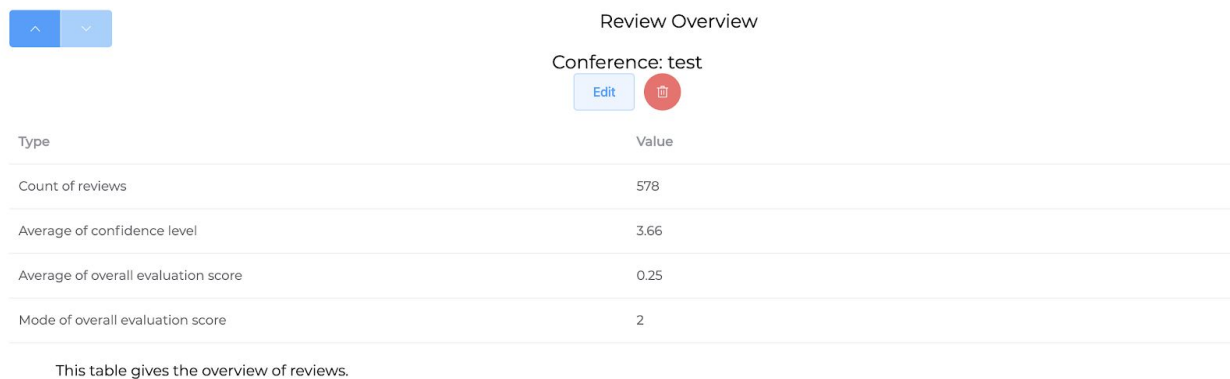
3.3.3.2.1 Logic Behind change

1. Before the start of every download, the title, description and date to the document before the rest of the visualisations are added.

3.3.2 Include new visualisations to provide new insights

3.3.2.1 Section for Overview of Reviews, Submissions and Authors

This new section gives users an overview of the data files. It breaks down the data to give simple information of the files such as the count, average, mode, unique count, sum, max, min ,median and categorical breakdown of columns specified.



Type	Value
Count of reviews	578
Average of confidence level	3.66
Average of overall evaluation score	0.25
Mode of overall evaluation score	2

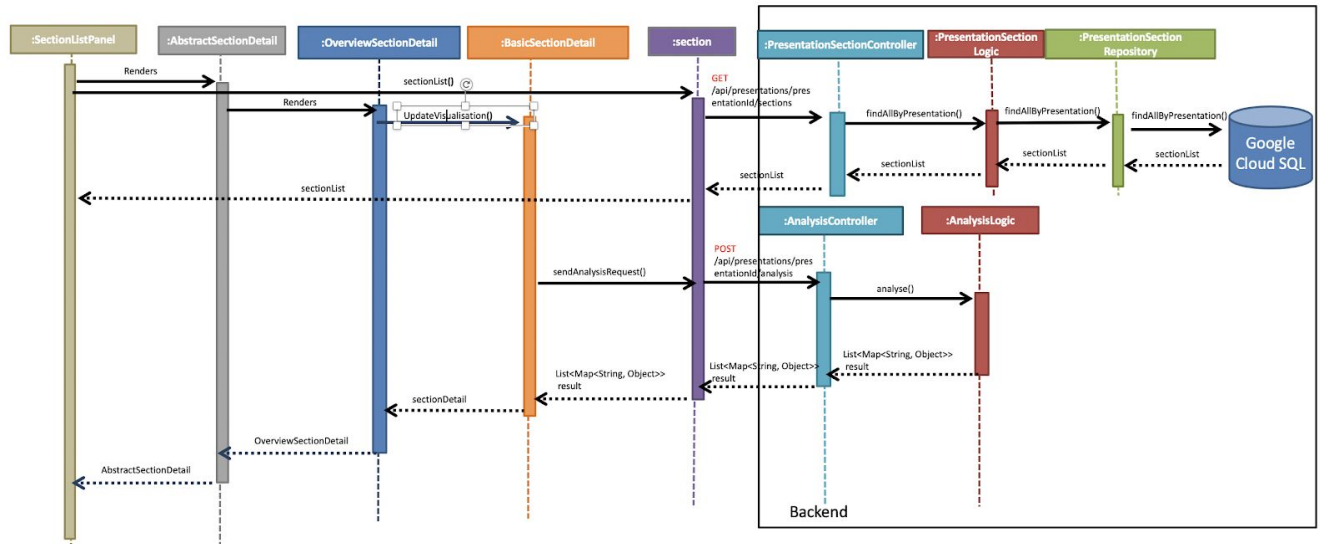
This table gives the overview of reviews.

Overview section for Review

3.3.2.1.1 List of changes

1. Added a new section detail called 'Overview Section Detail'
2. Added predefined queries: "Review Overview", "Submission Overview". "Authors Overview"
3. Added advance settings for 'Overview Section Detail' that allows users to customise their overview analysis

3.3.2.1.2 Logic Behind Change



Sequence Diagram for Overview analysis

1. When the user requests to add any of the “Overview” analysis, it will form its respective query from predefinedQueries.js.
2. Together with the data already obtained in sectionList, the query is passed to the Analysis Controller via POST `/api/presentations/{presentationId}/analysis`
3. Based on the query, it would return the required data in a list
4. Once the analysis is received, it would then passed to the Overview Section detail to be further processed.
5. The desired results would be placed into a table which will then be displayed.

3.3.2.1.3 Design Consideration

Type 0	reviews	Count	
Type 1	confidence level	Mode	
Type 2	overall evaluation score	Sum	
Type 2	overall evaluation score	Average	
Type 3	overall evaluation score	Count	
		Unique Count	
		Min	
		Max	
		Median	

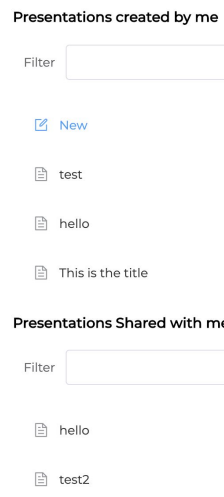
Settings for Overview Section

Under advanced settings, users can edit the type of queries that they want to be shown in the table. The “breakdown” function is different from the other functions, it is to deal with columns that has non-numeric categorization, it would return the count in each category.

3.3.3 Improve general user interface and experience

3.3.3.1 Shared Presentation list

In ChairViz, users are able to share their presentation via email. Users are now able to see a list of presentations that were shared with them via email. The user would not have to paste the shared presentation link.

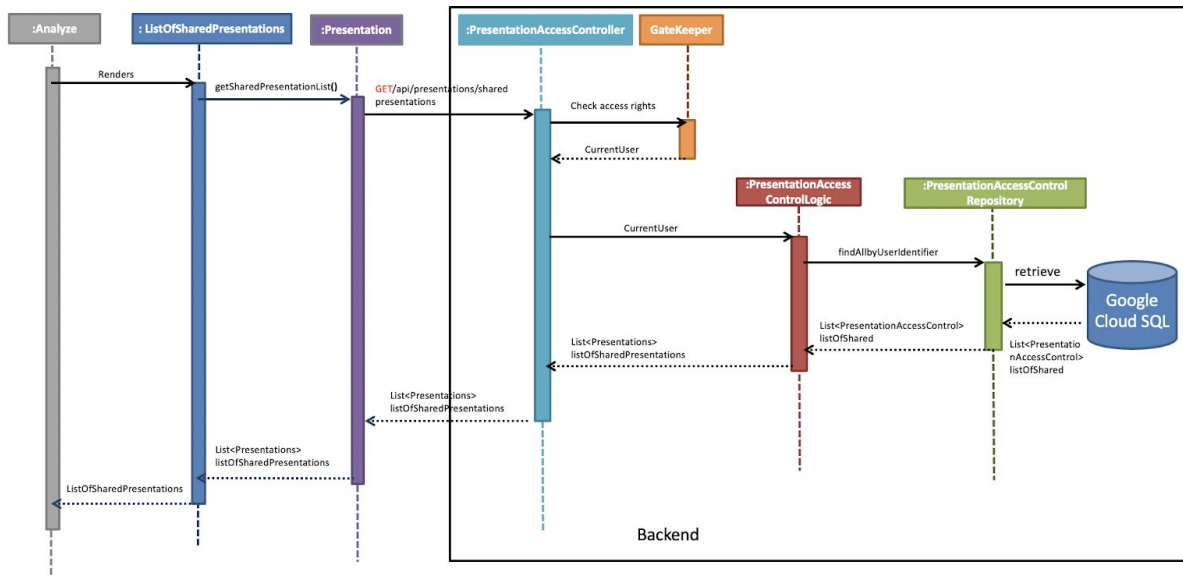


Shared Presentation list at the bottom of user's presentations list

3.3.3.1.1 List of changes

1. Added a component called "Presentation shared with me" below presentation created by me
2. Added API called "sharedPresentation" that returns a list of shared presentations
3. Created test cases for "sharedPresentation" API

3.3.3.1.2 Logic Behind Change



Sequence Diagram for Shared Presentations list

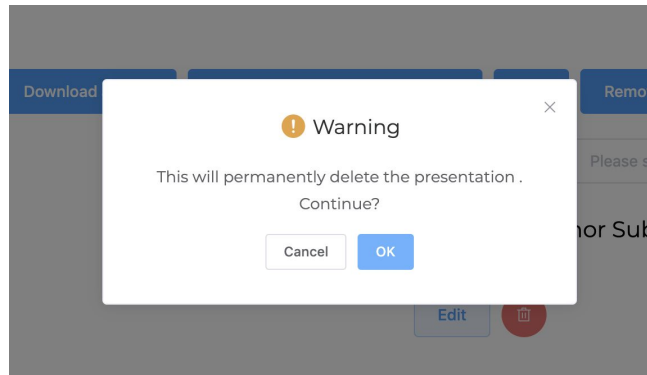
1. When the user clicks on Analyze tab, it will call the sharedPresentation API (GET/api/presentation/sharedPresentations) to the Presentation Access Controller
2. The controller will first get the user and check that the user is verified
3. Then, it would pass the Current User info to to the Presentation Access Control Logic to query the database.
4. The query will find a list of Presentation Access Control (PAC) based on the User Identifier (User's email).
5. It will then create a list and go through each PAC to get the respective presentation and add them to a list. Once done, it will return this list.
6. The list will then be displayed on ListOfSharedPresentation.vue

3.3.3.1.3 Design Consideration

The new API does not include the `userId` as when it is being called, as it will check if the user is authenticated before carrying out the query. Furthermore, we do not want malicious API queries to be made through valid user ids.

3.3.3.2 Delete Pop-up Before Deletion of Presentation/Section

Users are able to add and delete presentations and sections as they wish. However, users sometimes accidentally click a delete button that they don't intend to, and that could potentially delete an important customization in their analyses. Thus, we want to add a pop-up that asks for confirmation when a user clicks the delete button, such that such accidental deletions are minimized.

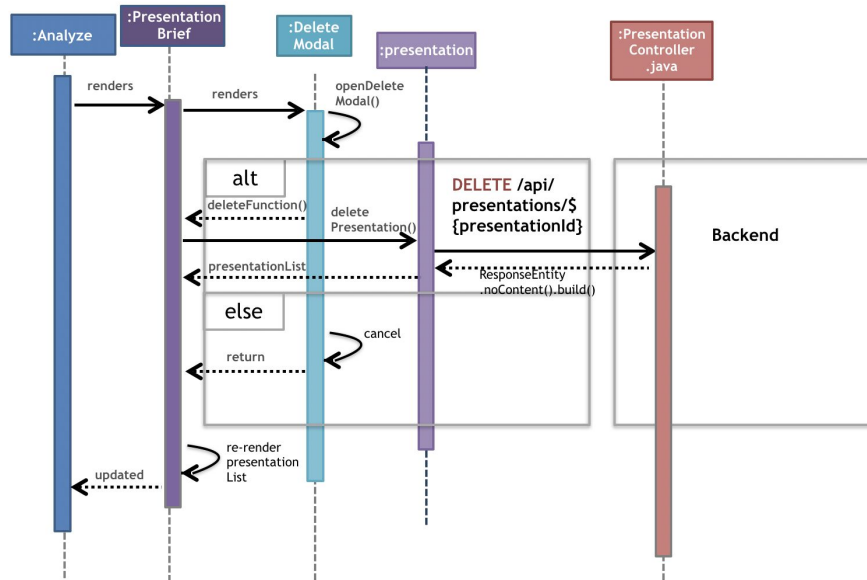


Delete modal that pops up when the delete button is clicked

3.3.3.2.1 List of changes

1. Created new Vue component DeleteModal, putting the existing design of delete buttons in PresentationBrief and BasicSectionDetail in the component.
2. Mount the DeleteModal on PresentationBrief and BasicSectionDetail, passing in which type of delete is being done each time, and the appropriate delete function.
3. Added Vue test for the new DeleteModal to ensure that the component renders the presentation and section delete buttons correctly.

3.3.3.2.2 Logic Behind Change



Sequence Diagram for Deletion of Presentation

The above sequence diagram illustrates what happens in this enhancement, using the user story of deleting a presentation, but this enhancement works for section detail deletion as well.

1. When user clicks the delete buttons in PresentationBrief and BasicSectionDetail, instead of immediately calling the delete function in the store, an elementUI MessageBox pop-up would appear, and ask the user if they would like to confirm the delete.
2. If the user clicks 'Cancel', the 'x' button, or anywhere outside of the pop-up, the pop-up would close and no action would be taken.
3. If the user clicks 'Ok', the `deletePresentation()` or `deleteSectionDetail()` function in the vuex store would be called.
4. The vuex store would send a DELETE request to the backend to delete the presentation/section, as per existing implementation.

3.3.3.2.3 Design Consideration

1. A new component, DeleteModal, was created instead of implementing the modal within the PresentationBrief and BasicSectionDetail so as to fulfill the separation of concerns and open-closed design principles.

3.3.3.3 Ability to Reorder Sections



Inter-Country Co-Author Submissions

Conference: easy

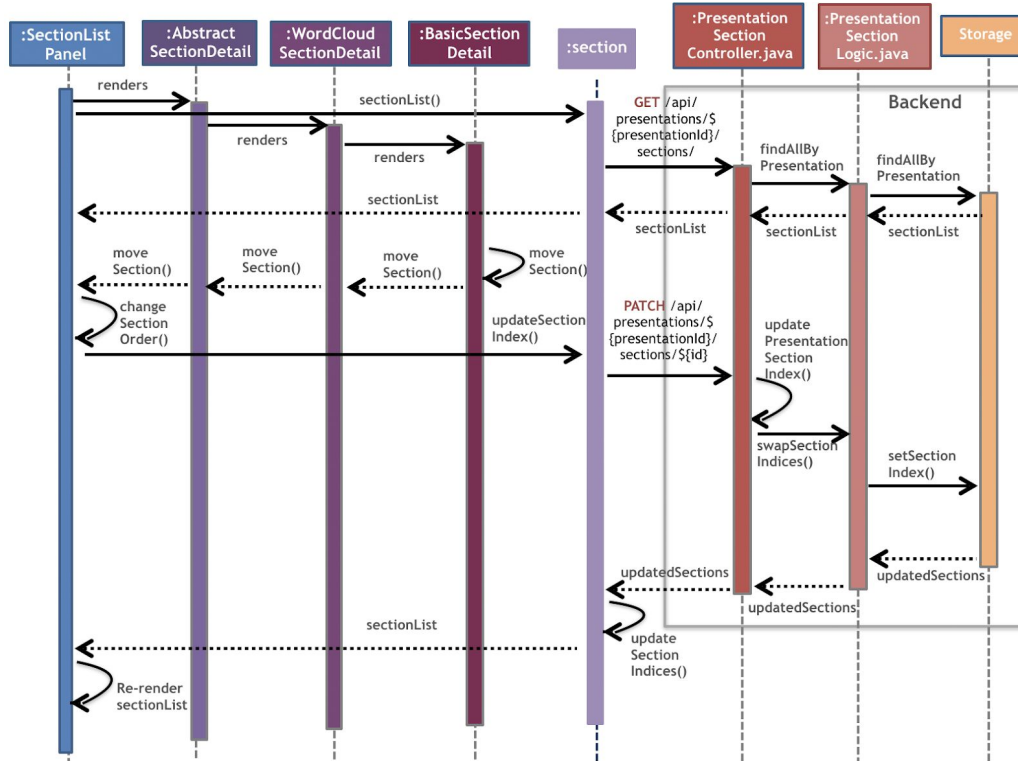


On the top left hand corner of each section are up and down buttons for reordering sections

3.3.3.3.1 List of changes

1. Added a new set of 'up' and 'down' buttons to the BasicSectionDetail component.
2. Created a method changeSectionOrder() in the SectionListPanel component and passed it through to the BasicSectionDetail where the buttons are.
3. Added new PATCH API endpoint that facilitates the swapping of just the section indices, along with method updatePresentationSectionIndex() in PresentationSectionLogic that then updates the section indices of sections.
4. Added new tests in PresentationSectionControllerTest.java for the new API endpoint.

3.3.3.3.2 Logic Behind Change



Sequence Diagram for Reordering Sections


1. When users click either of the reorder buttons on a section, it would trigger the `moveSection()` method, which propagates up to the `SectionListPanel` (as it requires knowledge of the neighbouring section).
2. The `changeOrderSection()` method is then called, which calls the `updateSectionIndex()` method in the `vuex` store.
3. The store will then call the `PATCH` API endpoint with the presentation, and the two sections which are to have their section indices swapped.
4. The backend would then update the section indices of the respective sections and save to the database, then return the updated sections.
5. The `vuex` store would then update its `sectionList` accordingly, and thus causing the `SectionList` to re-render, as shown in the sequence diagram.

3.3.3.3 Design Consideration


1. By creating a new `PATCH` API endpoint instead of using the existing `PUT` endpoint, we are creating a new, more efficient method that only updates the section index of the sections, instead of having to update all the attributes of a section, which would be the case if we used the existing `PATCH` endpoint.


3.3.3.4 Prevent duplicate presentation names

Presentations created by me

 New

 234

 aaa

 aaaa

✖ Cannot have duplicate presentation names

* Name 234|

Description

Save


i Please create presentation before adding sections


Error Message that appears when the user tries to add a presentation with the same name as another in the list

Presentations created by me

 New

 234

 aaa

 aaaa

✖ Cannot have duplicate presentation names

* Name 234|

Access Control Created by test@example.com

🔗 SHARE

Description

Save

Cancel

Delete

The same error message also appears if the user tries to edit an existing presentation to have a duplicate name

3.3.3.4.1 List of changes

1. Added a function `hasDuplicateName` that iterates through the list of presentations to check for duplicates
2. The Save button will now be disabled if a duplicate name is detected, preventing the user from saving a presentation with duplicate names
3. Added an error message that appears when a duplicate name is in the Name field, warning the user that duplicate names are not allowed.

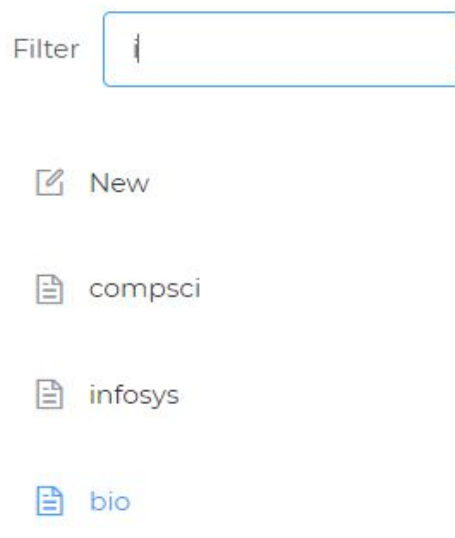
3.3.3.4.2 Logic Behind Change

1. `hasDuplicateName` first obtains the list of shared presentations from the store, if the selected presentation is a shared presentation, we ignore duplicate checks, since shared presentations naturally can have duplicate names as they are from different sources.
2. Otherwise, it obtains the list of presentations belonging to the user, and checks if there are any presentations matching the name of the selected presentation, but with a different id, if there are, it returns true, otherwise, it returns false
3. If `hasDuplicateName` returns true, `duplicatePresentationErrorMsg` will appear, and the Save button will be disabled until the user changes the name to be a non-duplicate one.

3.3.3.4.3 Design Consideration

1. An alternative to the current implementation is to have the presentation continue to be added as per normal, and only do the check while adding the presentation, if invalid, the presentation would then be deleted or edits reverted. While this is possible, changes to many different classes would have to be made, such that the `PresentationBrief` can obtain from the store that a presentation is about to be created with a duplicate name.
2. The above alternative is also unnecessary, since the key information that is needed to determine if there is a valid or a duplicate name comes from `PresentationBrief` itself, thus, there is no need to pass the name through to other classes when it could perform the check immediately with information it can obtain from the store.

3.3.3.5 Search Bar to Filter by name



Typing into the bar at the top of both the presentation list and the shared presentation list allows the user to filter the list to only display presentations whose names contain the specified string

3.3.3.5.1 List of changes

1. Added a form to both ListOfPresentation.vue and ListOfSharedPresentation.vue to allow users to input the term they wish to filter by
2. Modified presentations() in both ListOfPresentation.vue and ListOfSharedPresentation.vue to filter the presentations to display based on the respective lists found in store.

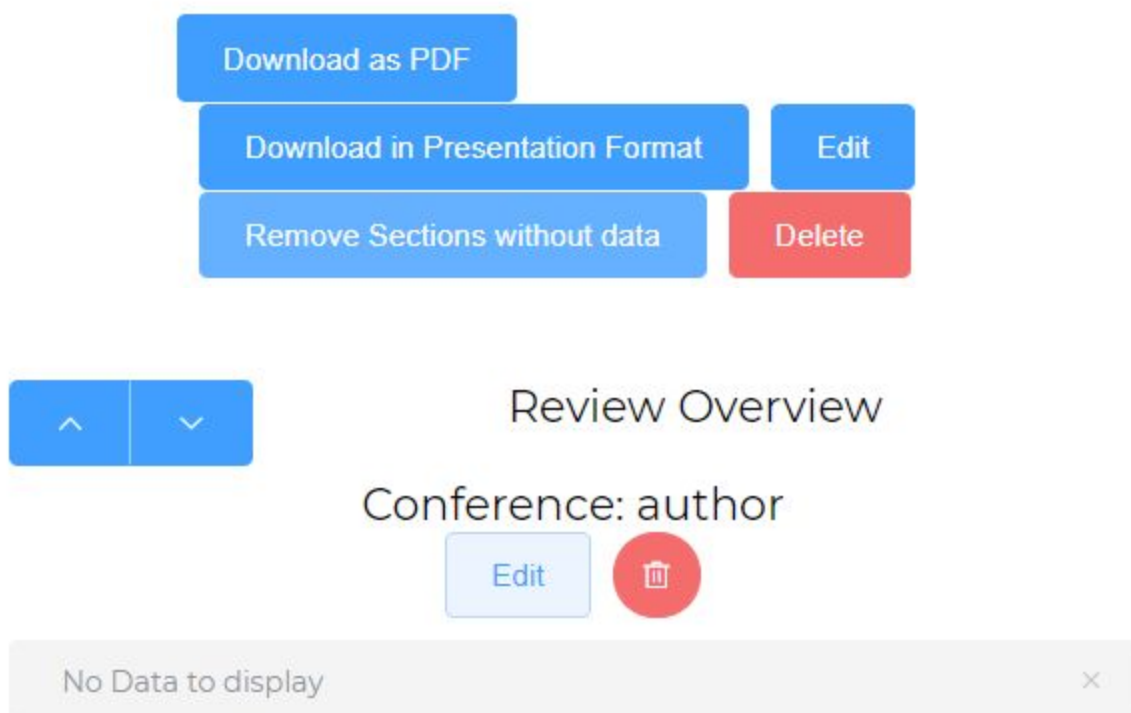
3.3.3.5.2 Logic Behind Change

1. Instead of returning the respective presentation lists straight away, presentation now takes in the presentation list as well as the string that the user input in the form and performs a filter on the presentation list. This new filtered list is then what is displayed to the user on the webpage.
2. If the user makes any change to the input in the form, the presentation method will run again, allowing the presentation list to be updated immediately without having any additional input required. An exception is if the form is empty, in which case, the entire presentation list will be returned.

3.3.3.5.3 Design Consideration

1. The main alternative is to design a search bar, which would be located in the same area as the name and the description of the PresentationBrief. Upon typing a query there and pressing a search button, the list of presentations will altered such that it only displays the presentations that have been searched for.
2. However, to support the single responsibility principle, it makes more sense for functionality that impacts each presentation list to be within the same module that is handling the display of the presentation lists, rather than having another module serve as the interface between the user and the presentation list.

3.3.3.6 Delete all sections that do not have data



This table gives the overview of reviews.

The button Remove Sections without data will delete all sections without any data to display such as the section above.

3.3.3.6.1 List of changes

1. A button deleteSectionsWithoutData has been added to PresentationBrief
2. All section details have been modified to contain the attribute hasData

3. PresentationSection and PresentationSectionLogic have also been updated to support hasData

3.3.3.6.2 Logic Behind Change

1. Originally, whether or not a section displays “No Data to display” is based off a variable named hasData. However, this hasData is not saved in the store, so there is no way for the PresentationBrief, which houses the button, to know whether or not a section has data. Thus, hasData is a new attribute that is added to each section in the store, and is being sent when the store gets updated.
2. On top of this, the data in each section does not get populated until each section has been created, however, by this point, the section has already saved the data into the store, so all freshly created sections in the store will have hasData be false. To combat this, after each section has been updated, the section saves its information to the stack again using updated(). This would trigger updated() again, so we have a variable firstRefresh which is set to true after this save to stop it from going into an infinite loop.

3.3.3.6.3 Design Consideration


1. Our original idea was to filter the possible selections of sections to add to remove all sections that would have any data. However, this idea was scrapped as it meant that every time a presentation is loaded, there would be a massive overhead as the system generates each possible section and checks whether it has any data, then removing it.
2. Another idea was to allow the user to add any section they want, but delete it immediately if there is no data. While this is possible, since a presentation section is being created regardless, we decided to keep the created section, if the user does wish to keep these sections in the presentation, but allow the user to have an easy way to remove these sections if they are unwanted.

3.3.4 Persistence and Management of multiple conferences data

3.3.4.1 Persist Data of Multiple Conferences

In order to allow scalability of the application where instead of only being able to store data for one conference at any point of time and even other functionalities only being supported for one conference, this implementation was focussed on making the app support storing of multiple conference data for each user. This eventually allowed other

functionalities also to take advantage of multiple conference data and supporting analysis and visualization of many conference data at the same time possible.



The screenshot shows a web form with a validation error message at the top: "Conference Name length should be of minimum of 3 characters". Below the message is a text input field containing the letter 'c'. To the right of the text field are four dropdown menus: "EasyChair", "Author Record", "Yes", and "Default Author Mapping".

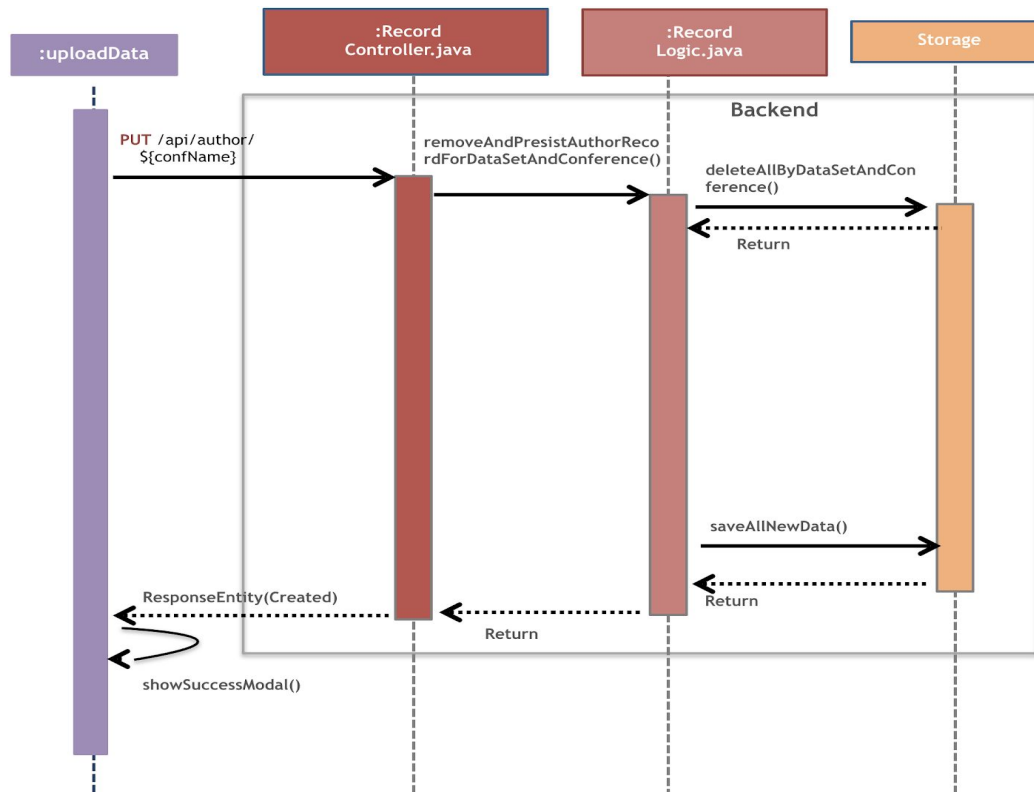
Add conference name field when uploading data

3.3.4.1.1 List of Changes

1. Migrate database entities to support each record type to have metadata of conferenceName
2. Front-end changes to support adding/updating for new or existing conference
3. Backend updates for supporting new schema and handling checks when uploading new data
4. Tests to ensure working of previous functionality and new functionalities

3.3.4.1.2 Logic Behind Change

1. The user can now either add data for a new conference or update data for existing conference. Since the user can see existing conference names the user experience helps in preventing user errors
2. Consideration was put into what's the best way to extend the functionality while supporting open-closed principle. Migration was found to be the logical way as it involves minimal changes to existing code and easily extended upon existing working of data. Moreover, it also allowed open-closed principle to be used if this data storage of conference had to be further extended for any more new functionalities
3. Test Driven Development was followed as this was a functionality that could potentially break existing functionality when new changes are introduced. Hence, by writing more tests to cover those scenarios, it was more foolproof when those tests along with tests for new functionality passed after supporting data for multiple conferences



Sequence Diagram in backend server when data uploaded

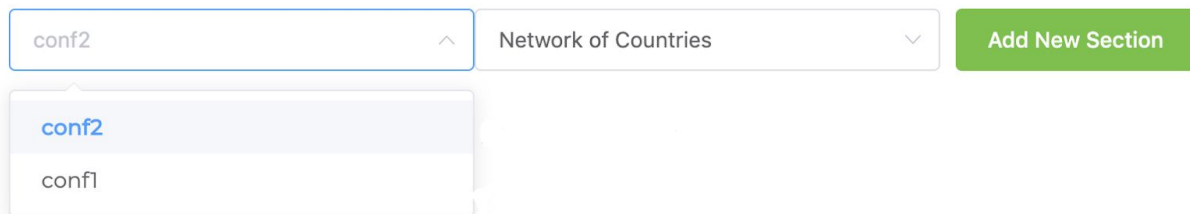
3.3.4.1.3 Design Consideration

1. In order to uniquely identify each conference for a user, the logical way seemed some kind of unique hash to store but this would be a bad experience for the user and moreover require a reliable manual hash function which wouldn't break things. Hence the decision was made to go with conference names as they would be unique in itself.
2. Since now the data is going to increase rapidly given each user can store multiple data, the decision was made to index to ``conferenceName`` and ``testData`` columns as they would improve performance a lot. The only con was that write would be slower but since over time the read:write ratio would be 1000:1, it wasn't considered an issue.
3. The names of conferences for each user are stored in database. When the Import Page is queried, this data is stored in Vuex state management library so

that no repetitive queries are made to backend querying the same data. This also gives us **Single Source of Truth**.

3.3.4.2 Visualization Generated from Different Conferences

With support of multiple conference data being stored at the same time, this allowed for users to effectively use this data. This feature implementation allows user to create presentation section from data from any one of the conference data that they have uploaded. This helps in not losing old presentation due to data override and moreover gives them flexibility to create presentation of their own choice of data without having to upload and delete data multiple times



Choosing conference to get data from when adding section

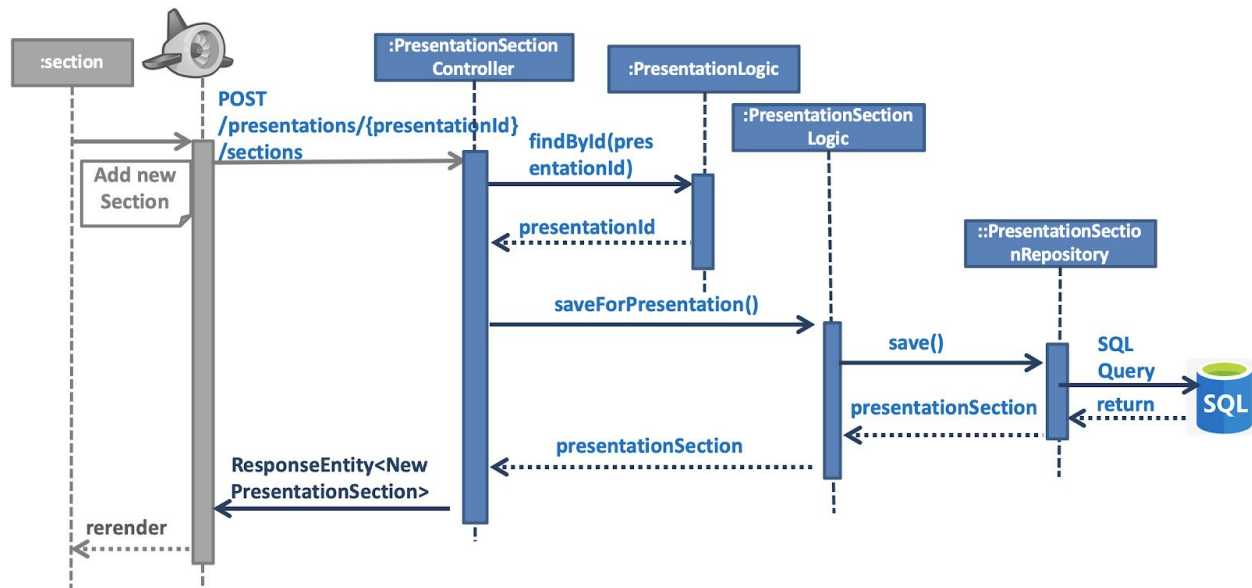
3.3.4.2.1 List of Changes

1. UI support for each presentation section to support data being taken from any one of the persisted conference.
2. Allowing user to edit section by changing the conference the data is chosen from.
3. Migrating database and backend queries to support considering filtering by conference names
4. Updating predefinedQueries to support conferenceName checks and filters

3.3.4.2.2 Logic behind Change

1. Since each presentation section now had data from one conference, it had to be able to support all the existing queries for visualization and easily extended for new visualization, so it was added as a key-value in `predefinedQueries`.

2. For every API call to perform analysis and return data, the `conferenceName` was added to JSON body and sent to backend to process it in the sql query.
3. Adding the `'conferenceName'` column to all record entities and `presentationSection` helped in easily querying the right data without any extra overload.



Sequence Diagram in backend server when new presentation section saved

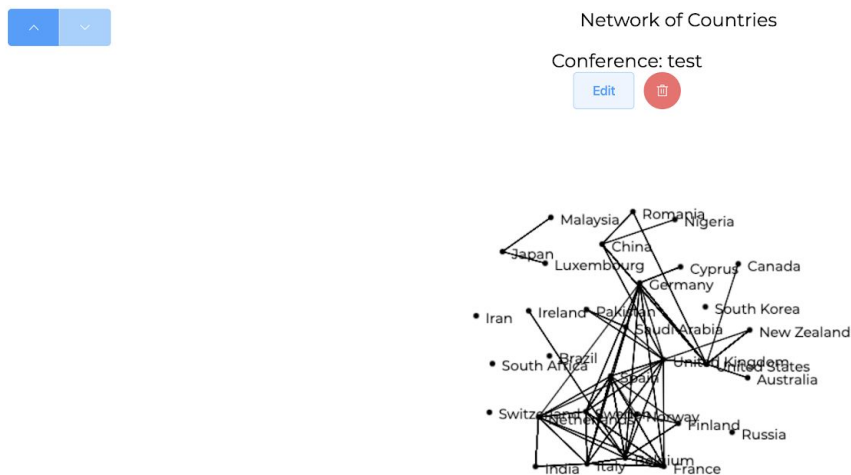
3.3.4.2.3 Design Consideration

1. One of the most important design decisions to be taken was whether to support 'one conference for one presentation' or 'one conference for one presentation section'. The decision involved input from product owners as they felt each presentation should be able to support more than conference data and moreover it went well along with people sharing their presentations and hence preventing from any privacy issue of data sharing.
2. Some queries for visualization are hardcoded for the `'involvedRecords'` and hence needed manual updates. The design decision was made to migrate these queries to formatted way so that when extending upon adding new visualizations, the developer doesn't need to worry about old design decisions made. However, this was only possible to an extent and rest were manually updated.

3.3.5 Co-authorship Queries

3.3.5.1 Section Network Graph for Co-Authorship

Since there are many submissions where authors come from different organisations and countries, the “Network of Countries” query will plot out the various countries and link them if there are submissions where their authors are from these countries have worked together on. On the other hand, the “Network of Organisations” query will plot out the various organisations and link them if there are submissions where their authors are from these organisations have worked together.



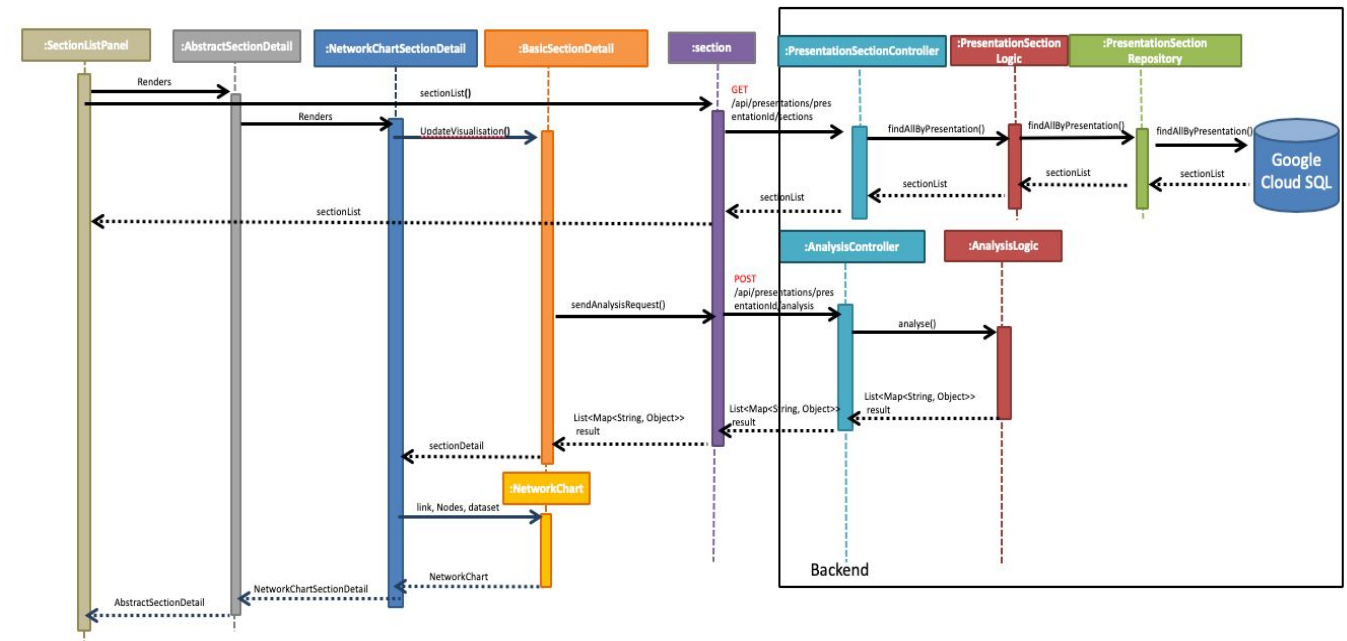
This network graph shows the countries linked based on submission

Network chart for Co-Author Submissions

3.3.5.1.1 List of changes

1. Added Library "vue-d3-network"
2. Added a new section detail called 'Network Chart Section Detail'
3. Added new chart 'Network Chart'
4. Added predefined queries: Network of Countries and Network of Organisations
5. Added advance settings for 'Network Chart Section Detail' that allow for editing of the graph

3.3.5.1.2 Logic Behind Change



Sequence Diagram for network chart analysis

1. When the user requests to add any of the “Network” analysis, it will form its respective query from predefinedQueries.js.
2. Together with the data already obtained in sectionList, the query is passed to the Analysis Controller via POST `/api/presentations/{presentationId}/analysis`
3. Based on the query, its would return the required data in a list
4. Once the analysis is received, it would then passed to the Network Section detail to be further processed.
5. The links and nodes and dataset will be passed to NetworkChart to generate the graph
6. The graph will then be returned and displayed.

3.3.5.1.3 Design Considerations

Under Advanced Settings, besides the basic edits such as changing the description and query, users are also able to edit the network graphs.

Links by

submission_id

Nodes

country

Force

Canvas Height

Node Size

Font Size

Labels

Labels off

Labels on

Settings for network graphs

Users are able to specify the relationship between the nodes and what the nodes are. The force of the graph would be how tightly packed the nodes will be. Since some graphs would be much larger, users will be able to increase the canvas height to show the entire graph. Users are also able to choose between showing and not showing the nodes labels as well as adjust the node and label font size.

3.3.5.2 Inter-Country Co-Author Submissions Bar Chart Visualization

To add to the new graph network to view co-authorship links, we also added a bar chart visualization that displays the top 10 (or more if the user chooses) co-authorship groups of countries and their respective number of submissions.



Bar chart of Inter-Country Co-Author Submissions

3.3.5.1.1 List of changes

1. Added predefined query: 'top_country_coauthors' which comprised a customized 'involvedRecords' query to extract out unique countries of co-authors and aggregate those who collaborated with people from other countries.

3.3.6 Flexible data schemes

Due to our hefty workload and the skills of our team members, we did not manage to implement any feature in this category. We decided to focus more on the other categories, mainly improving general user interface and experience. We also focused on some technical improvements like code quality and design metrics and technical debt. However, we did come up with some features that could fit in this category, such as using synonyms to allow users to upload data files with different column names.

3.3.7 Technical Improvements

As this project even though primarily designed for users who want to analyse conference data and presentations, for a group of developers maintaining a software engineering project, we found it necessary to include some improvements in the codebase for code design, style and software engineering practices and rules. Below are a few listed design improvements we have implemented in the project

3.3.7.1 ESLint Rule Configurations

It is important that the codebase is in a certain code style and follows that throughout as this improves the developer experience and saves time when extending certain functionalities. After our first two sprints we realised that we wasted a lot of time in fixing these issues through our code reviews when we could have automated it by ESLint rules and imposed them through CI. By researching and adding certain rules the rest of the development process was more pleasant and had clean code style. This would eventually lead to using the right programming language practices and eventually lead to fewer bugs due to **static analysis** of code.

3.3.7.2 Modularization and Configurations

As part of the project there were multiple third party tools that were used to get out of the box solutions and prevent re-inventing the wheel. Although this was very useful, it was noticed that this didn't follow the good practices that the third party libraries had suggested. As part of our effort in making the code **Modularized** we looked into some of them like BrowserList, ESLint, PostCss and followed on how these configurations should be setup in terms of best practices. It also followed **Separation of Concerns** as each tool had separated configs and not dumped into dependency management of package.json

3.3.7.3 Security Improvements

1. Our first main effort was put in updating our Third Party NPM libraries to version where they had at most low severity of security issue. It was inspired from a few technical talk one of which being <https://www.youtube.com/watch?v=0dgmeTy7X3I>. Hence, from having 15+ moderate and 2+ high severity security critical packages, we went ahead and updated them while ensuring that there are no breaking changes in our code through tests and manual inspections. It was necessary for us to check for **Regression** of our codebase as part of SE principles.
2. The current CD functionality exposed some user critical data which should be hidden from the internet. Hence, when implementing CD, we made a few changes to ensure that these files are encrypted when uploading to cloud and store the keys in a safe place where they can be decrypted from.

3.3.7.3 Other Improvements

1. Since the front-end of the code had a lot of hardcoded and manual inputs, we realised that for new developers touching this codebase and not knowing its deep understanding may not change everything as expected and break things. For this we had written some critical tests which would break the moment something hardcoded is changed and would ensure the code is always working. One of them was writing tests to ensure that the `predefinedQueries.js` format is always correct even when new data is added or if some developer changes the format. This indeed went on to help us when a couple of developers were working on the same part and made regressional changes.

2. Having been given an existing implementation, it was important for us to ensure that any feature we extended upon or added didn't end up violating any current principles/patterns. Hence, we made it compulsory in our sprint meetings for an informal RFC <https://whatis.techtarget.com/definition/Request-for-Comments-RFC>, which would talk about implementation details and the developer could go ahead with it if everything seemed fine to other team members.

4. Suggestions and Improvements

4.1 Access Control With Different Type of Privileges

Currently, there is only one level of access control for ChairVise. There might be a need for people who should not be generating visualizations to be downloading or accessing analyses. Thus, it could be nice to have some form of access control or different user groups such that certain capabilities of the system can be blocked off for certain user groups.

4.2 Improve Data Upload Interface

Currently, the 'Import Data' interface is a little manual as one has to manually type in the name of a conference to be uploaded. Thus, there is a chance of transfer error if, for example, I name my conference 'myFirstConference', and upload author.csv for that conference. Next, when uploading the submissions.csv, i type in 'miFirstConference' without noticing. It could be a slightly clunky experience when the user's visualizations for 'myFirstConference' do not get generated, and the user has to realize that he/she typed in the conference name wrongly.

4.3 Generic CSV support

The current version of the app only supports for EasyChair and SoftConf and has hardcoded column mapping and also would not work for other conference data types. In future development, it would be ideal to form a template and validate all uploaded files and user could be provided an online mapping view to map his input file fields to the template fields. This will provide a generic template with all necessary data required for database schema and will also form as a data constraint for external environment.

4.4 REST API Communication Pattern and Error Handling

Currently the app doesn't have a fixed interface/messaging pattern for communication between frontend and backend. Having no fixed pattern makes it complex as each entity defines the interface in its own manner. This causes no right way for front-end to get error messages and HTTP error codes. By forming a correct interface and proper error handling we can get high cohesion which will allow to reuse code. Moreover, the frontend currently didn't have a good error handling mechanism where server could send concise detail of error rather than the stack trace and the Vue app could show it as a popup for a good user experience

4.5 Sharing presentations and Sharing Conference Data

When the sharing presentation feature was integrated into the application, one use case wasn't taken into consideration that if a presentation can be edited by another user, he will be adding analysis from his own uploaded data. This wouldn't make sense as each presentation section isn't labeled or belongs to any user. This might not be intuitive to users and they wouldn't know from whose data the charts are generated. To solve this on a higher level, an architectural change will be needed where even the uploaded data is shared between users if allowed and hence it will allow more easy collaboration and even allow easy extension for new features without having to explicitly write edge cases for multiple use cases.

5. Development process

5.1 Set-up and Prioritization of Tasks

We took a week and a half to familiarise with ChariVise, get the existing implementation up and running on our own machines. We then came up with the different features to be added as enhancements, and divided the tasks based on frontend and backend features. We prioritized what enhancements to work on based on our individual skill-set.

5.2 Weekly Sprints

Subsequently, we held weekly “stand-ups” every monday evening to share our progress on the past week’s task, and what enhancement we plan to work on for the coming week. Over the course of the week, whenever anyone has a PR that is ready for review, we would flag it as that and just sound it off on our group chat. Anyone else can then pick up that PR to review and do user testing. Similarly, when we noticed any bugs in implementation or the existing application, our practise would be to create an issue on Github and assign it to the person who originally worked on the buggy feature.

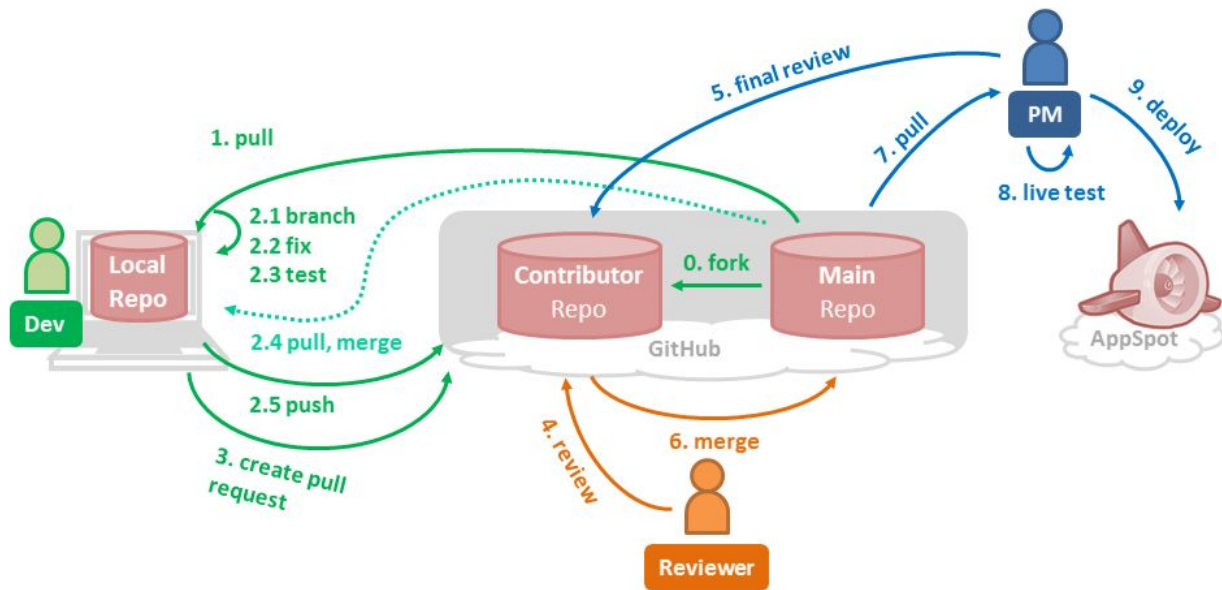
We followed a weekly sprint scheduled from Tuesdays to Sundays with sprint meetings on Monday to demo features, distribute work and prioritise requirements. A quick sanity check was done once all requirements for a particular sprint were pushed to master. In total we had 5 successful sprints with increasing productivity towards the latter as we were able to assign more accurate story points to our expected work. We used GitHub Project management tracker for keeping track of progress and to ensure all of us are achieving our goals and deadlines.

The project tracking board can be found here:

<https://github.com/CS3219-SE-Principles-and-Patterns/chairvise3-0-2019-team-17/projects/1>

5.3 Development Workflow

This workflow is an adaptation of the GitHub flow.



Appendix A - Documentation

Developer Documentation

Vue, Vuex⁴ and Spring Boot⁵

The existing front-end application is written in Vue. For backend, Spring Boot is chosen.

Google App Engine

Google App Engine (GAE) abstracts the low-level details such as HTTP server implementation. It lets the developer focus on high-level business logic and interaction and ease the process of deployment. Therefore, to simplify the development process, GAE is chosen to be the server.

Google Cloud SQL

As mentioned above, MySQL is chosen to be the backend database which is responsible for storing both application data and for analyzing queries sent by users. Setting up a MySQL server might be complicated and time-consuming. Fortunately, Google Cloud SQL provides pre-configured MySQL instance. The database also works well together with the Google App Engine.

Continuous Development (CD)

In the initial stages, Travis has been set up on Github to build and test the software automatically. Tests have been added for the backend and frontend functionalities. This helps to streamline the process of building, testing, and deploying new code efficiently in a live server. It minimizes the time for the project to build, test, and releases new features while ensuring that the product is maintained.

⁴ Vuex is a state management pattern + library for Vue.js applications. <https://vuex.vuejs.org/>

⁵ Spring Boot is designed to develop something as quickly as possible, with minimal upfront configuration of Spring. <https://spring.io/>

Design Diagrams

Overall Architecture

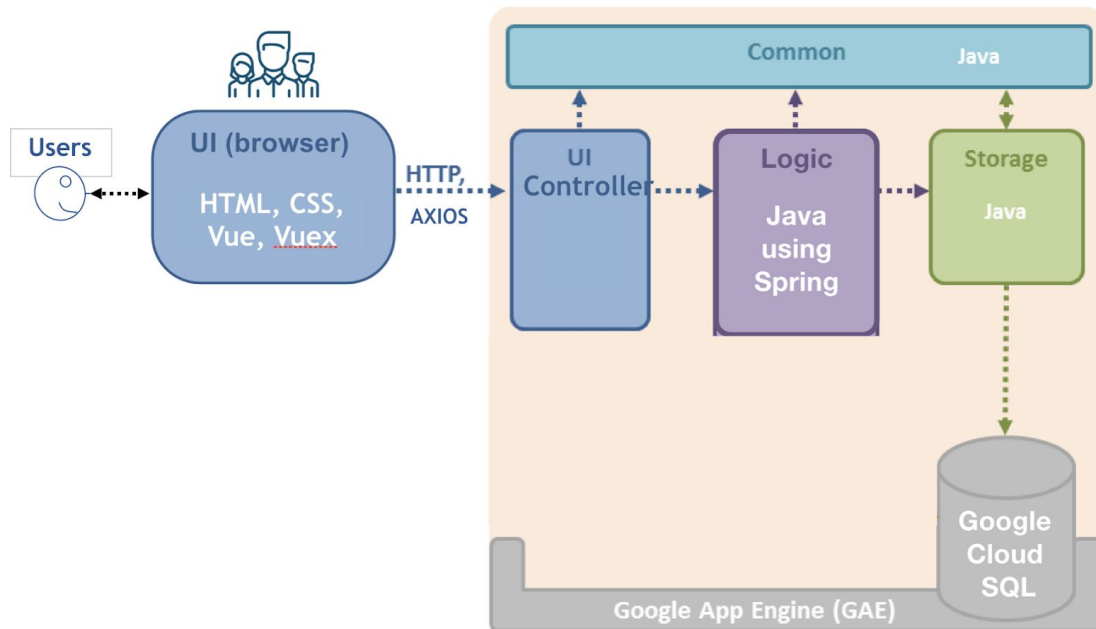


Figure 5

The application is designed with a layered design pattern. **Single Responsibility Principle (SRP)** is applied here. Each layer/classes has its own responsibility.

- **UI:** The component consists of API controllers and WebPage controllers. API controllers are responsible for handling API calls by the frontend. WebPage controllers are responsible for serving static production Vue Files.
- **Logic:** The main logic of the application is in Java using the Spring framework.
- **Storage:** The storage layer of the application uses the persistence framework provided by Google App Engine, using MySQL 5.6.
- **Common:** The Common component contains utility code (data transfer objects, helper classes, etc.) used across the application.

Sequence Diagram

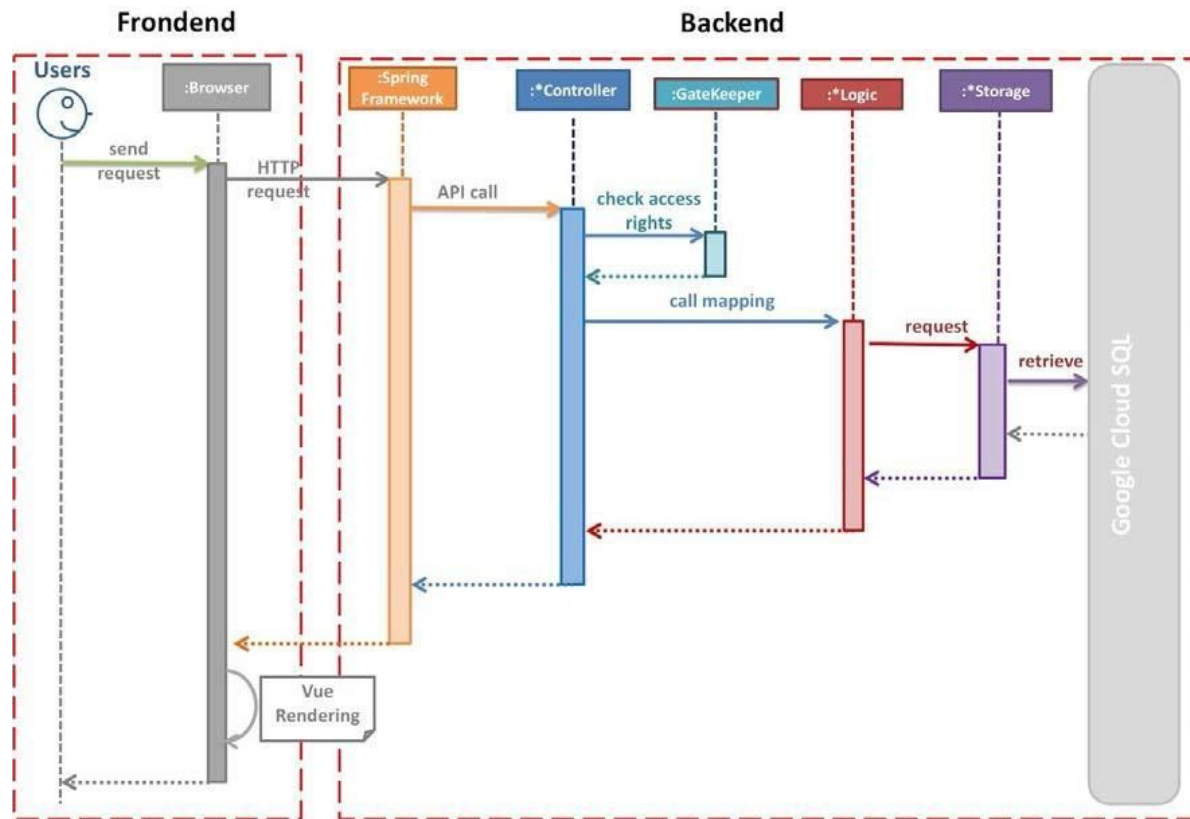


Figure 6

The diagram above showcases the workflow in the system of a typical request from the user.

1. A user sends a request to the browser for certain data or visualisation.
2. The browser sends an HTTP request to the backend. Spring framework will handle the HTTP request and pass to the correct controller based on URL mapping.
3. The controller checks the access rights of the user and executes the call by interacting with the **GateKeeper**.
4. Logic processes and requests the data from the storage component.
5. Storage retrieves the data from Google Cloud SQL.
6. The response will be sent back to the browser where it will use Vue to render the web page.

Backend Architecture

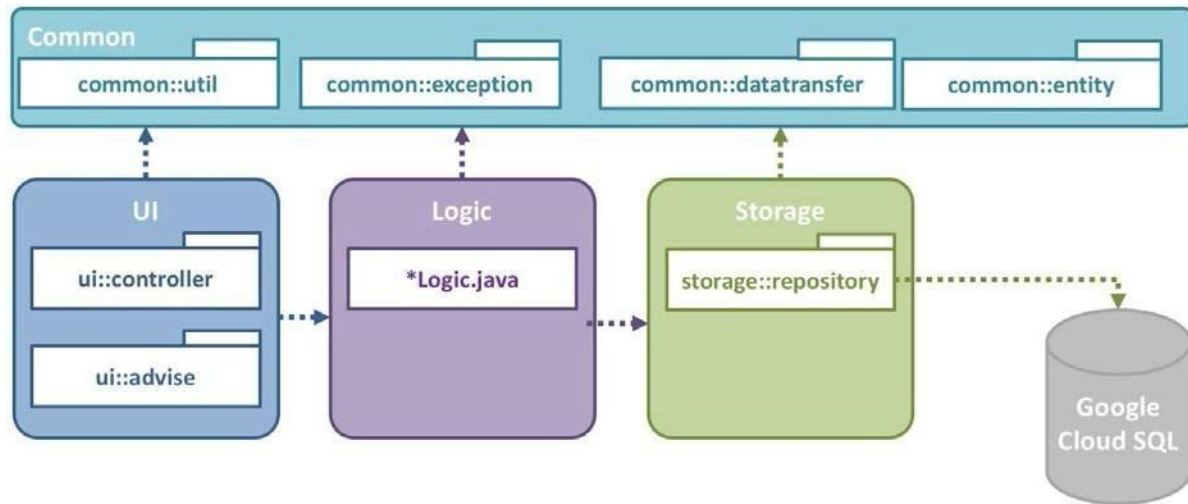


Figure 7

The diagram above illustrates the backend package overview of the existing system.

UI Component

The UI component is the first stop for all requests received by the backend of the web application.

- **ui.controller.api**: Provides backend Representational State Transfer (REST) API access to the users
- **ui.controller.data**: Contains helper objects to be sent to the client in JSON
- **ui.controller.webpage**: Handles static file requests for the users
- **ui.advice**: Handles exception thrown by the application

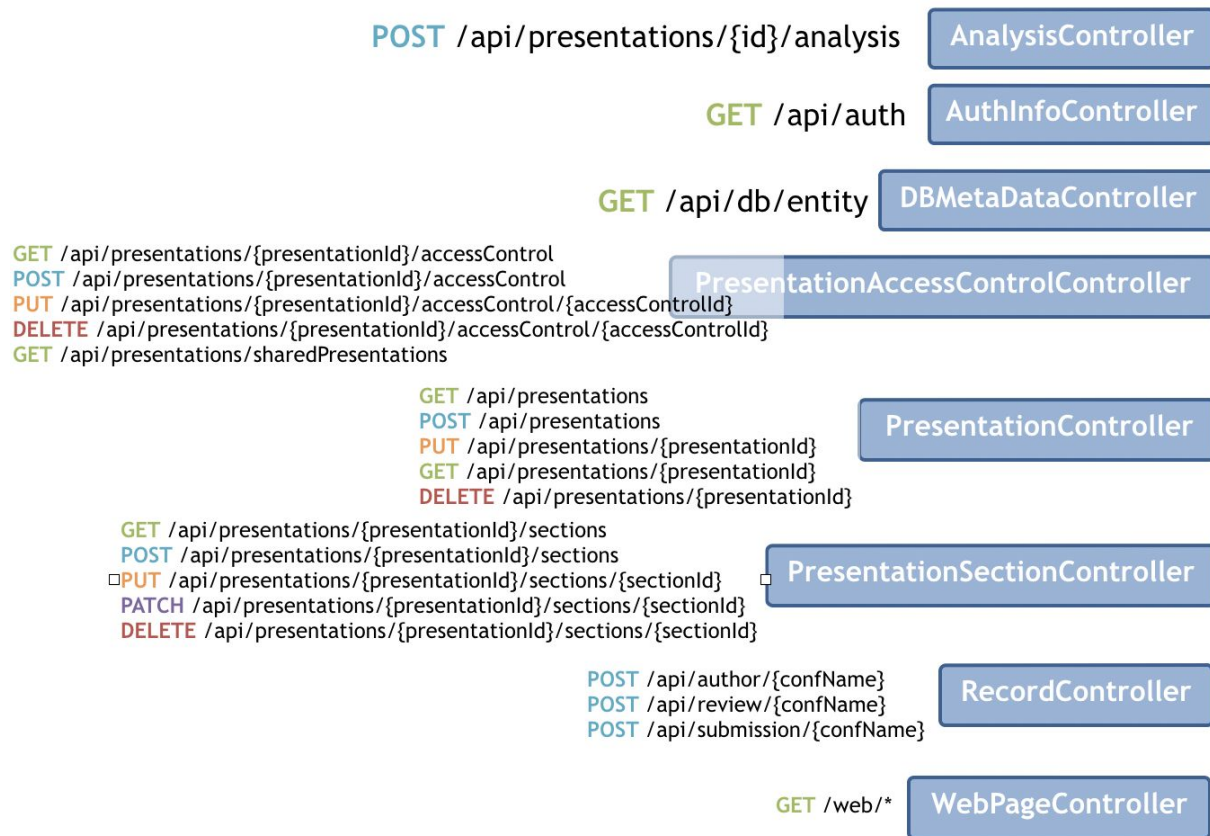


Figure 8

Figure 8 shows the mapping URLs and request method to the controller. In general, GET is used to retrieve data. POST is used to create data. PUT is used to update data. DELETE is used to delete data.

- **AnalysisController** is used to handle the analysis request sent by the frontend and issue SQL aggregation query to the backend.
- **AuthInfoController** is used to check the current authentication status of user. For example, it will give login URL if the user is not logged in and logout URL if the user is logged in.
- **DBMetaDataController** exposes the metadata, including name and type of the standard data template used in the application. This is useful when users try to match their own CSV file to the data template used in the application.
- **PresentationController**, **PresentationSectionController** and **PresentationAccessControlController** are responsible for the CRUD operations.
- **RecordController** accepts the CSV data imported by users and stores them in the database
- **WebPageController** serves the static production files built by Vue.


```

1 {
2   selections: [{
3     expression: "ROUND(SUM(CASE WHEN s_is_accepted = 'accept' THEN 1 ELSE 0 END)/COUNT(*), 2)",
4     rename: 'acceptance_rate'
5   }, {
6     expression: "a_organisation",
7     rename: 'a_organisation'
8   }, {
9     expression: "COUNT(*)",
10    rename: 'submitted'
11  }, {
12    expression: "SUM(CASE WHEN s_is_accepted = 'accept' THEN 1 ELSE 0 END)",
13    rename: 'accepted'
14  }],
15  involvedRecords: [{
16    name: 'author_record',
17    customized: false,
18  }, {
19    name: 'submission_record',
20    customized: false,
21  }],
22  filters: [],
23  joiners: [{
24    left: "a_submission_id",
25    right: "s_submission_id",
26  }],
27  groupers: [{
28    field: "a_organisation"
29  }],
30  sorters: [{
31    field: 'accepted',
32    order: 'DESC',
33  }, {
34    field: 'a_organisation',
35    order: 'ASC',
36  }]
37 }

```

Figure 9

Figure 9 shows a typical analysis request receive by **AnalysisController**. The request tries to rank organization based on the number of accepted submissions that organization has. It contains **selections**, **involvedRecords**, **filters**, **joiners**, **groupers** and **sorters**, which are enough to generate an SQL query for aggregation. For instance, the above analysis request will generate the below SQL query (the generation logic is inside Logic Component).

```

SELECT Round(Sum(CASE
WHEN s_is_accepted = 'accept' THEN 1 ELSE 0
END) / Count(*), 2) AS
`acceptance_rate`, a_organisation AS
`a_organisation`,
Count(*) AS `submitted`,
Sum(CASE
WHEN s_is_accepted = 'accept' THEN 1 ELSE 0
END) AS `accepted`
FROM author_record,
submission_record
WHERE author_record.data_set = 'user@email.com' AND
submission_record.data_set = 'user@email.com' AND
a_submission_id = s_submission_id
GROUP BY a_organisation
ORDER BY accepted DESC a_organisation ASC

```

The query result will be sent to the frontend to display visualization.

It should be noted that under such data aggregation framework. Any new visualization can be easily supported.

Logic Component

The Logic component handles the business logic. In particular, it is responsible for:

- Managing CRUD operations, ensuring the integrity of data.
- Providing a mechanism for checking access control rights.

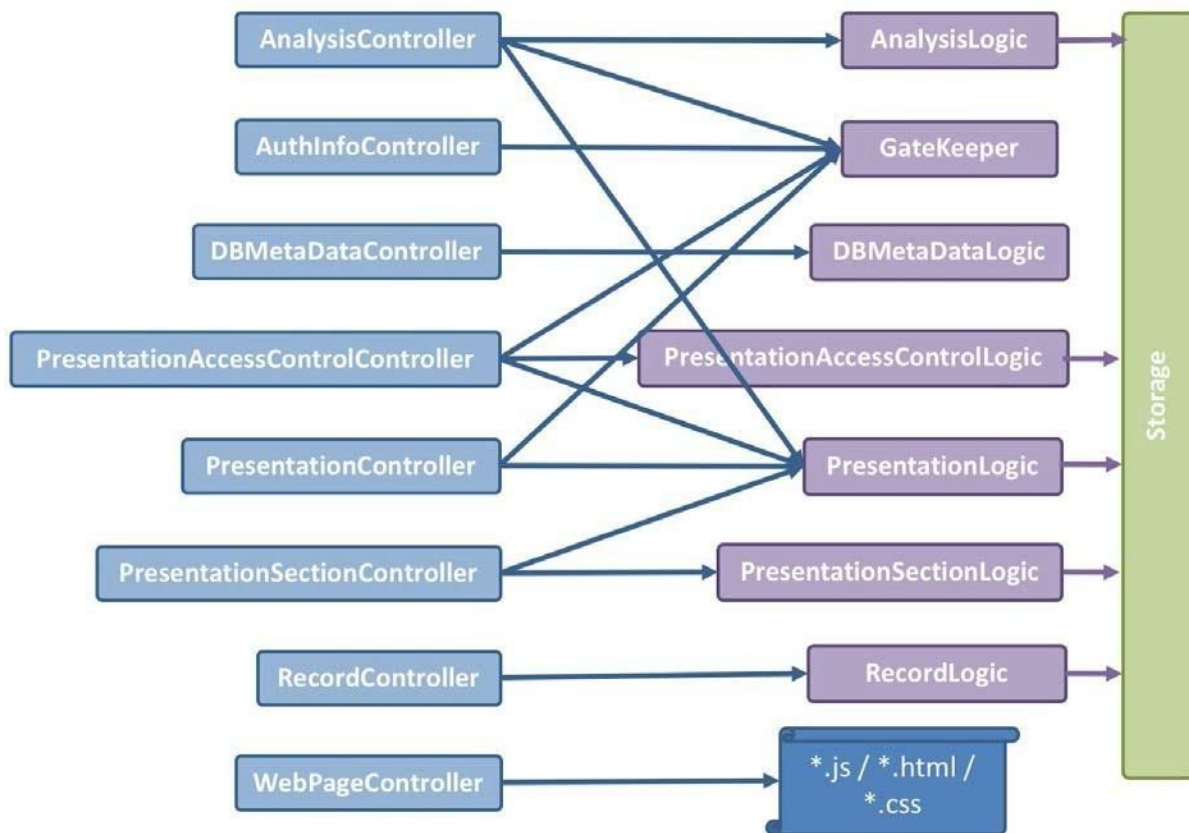


Figure 10

Figure 9 shows how each controller is making use of its corresponding logic. The dependency is injected (Dependency Injection) by Spring framework automatically.

Storage Component

The Storage component performs create, read, update and delete (CRUD) operations on data entities individually. It contains minimal logic beyond what is directly relevant to CRUD operations.

Figure 10 shows how logic makes use of the repository in the storage component.

- **AnalysisLogic** will issue SQL query directly to the Database to perform analysis query
- **GateKeeper** will use GAE internal APIs to check logged in user and also issue queries to database to check the access rights.
- ***RecordRepository** are responsible for persisting user uploaded CSV data to the database
- **Presentation*Repository** are responsible for storing user generated presentation, access control list and sections.

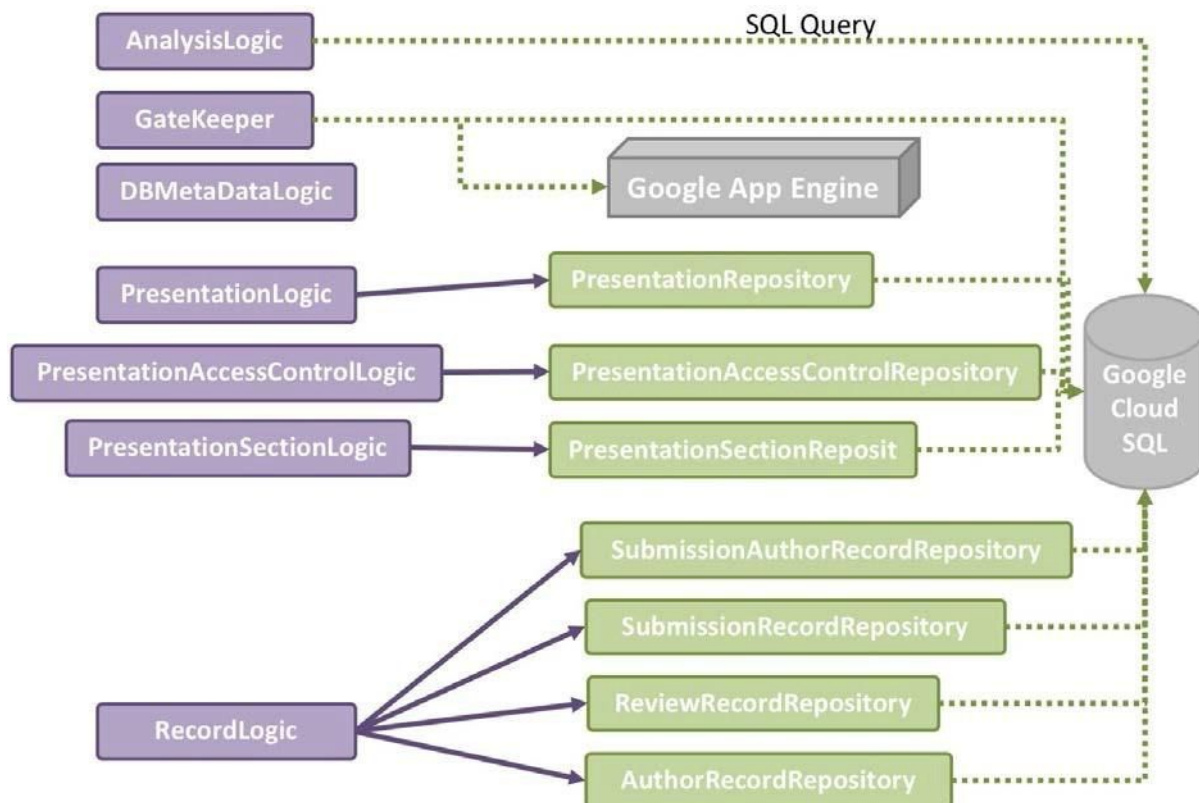


Figure 11

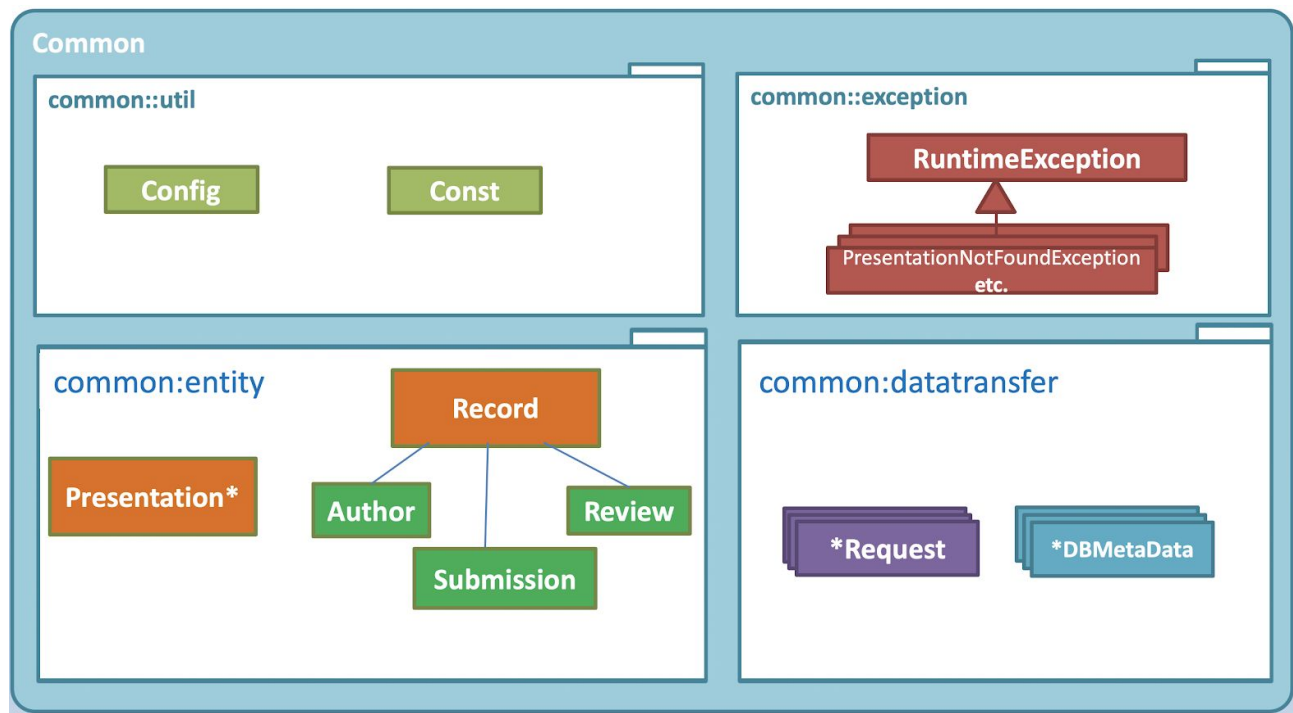
Common Component

The Common component basically contains common utilities used across the web application.

- `common.util`: Contains utility classes.
- `common.exceptions`: Contains custom exceptions.
- `common.datatransfer`: Contains data transfer objects (DTOs).
- `common.entity`: Contains entity stored in the database.

Figure 11 shows the entity relationship. Each entity has its corresponding table in the relational database.

- One presentation can have multiple sections or access controls.
- *Record is the data imported by users. There is a many-to-many relationships between `SubmissionRecord` and `SubmissionAuthorRecord` because one submission can be contributed by multiple authors. Each author can also contribute to multiple submissions



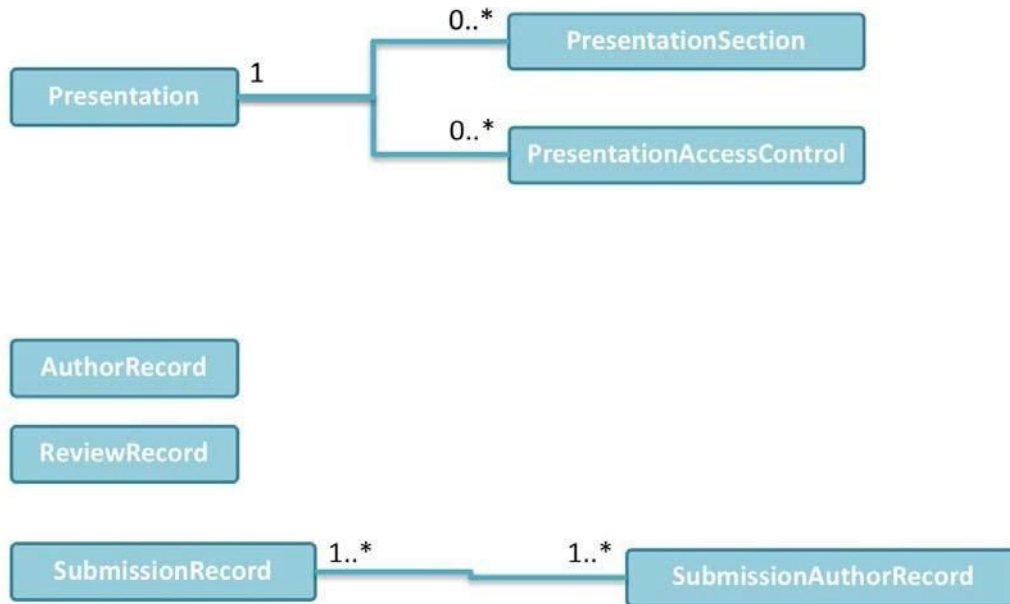
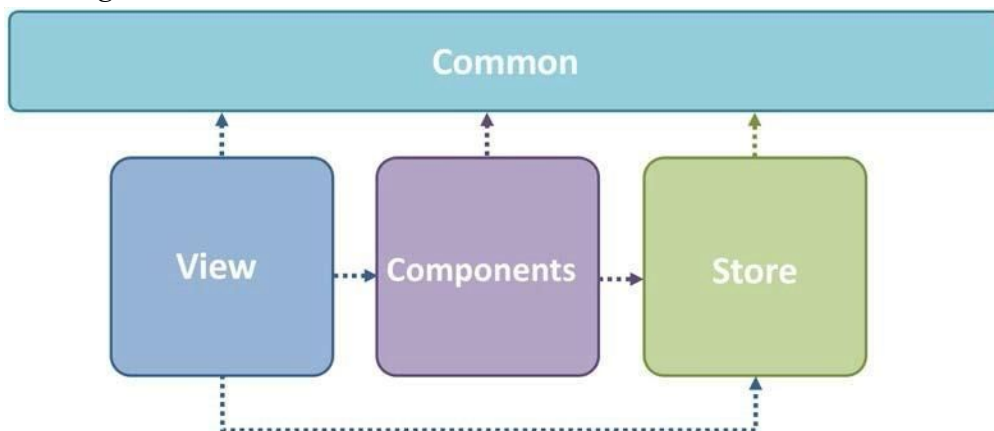


Figure 12

Frontend Architecture

Frontend Package



The diagram above illustrates the frontend package overview of the application.

- view: The view is mainly in charge of displaying pages of the application.
- component: The component contains reusable UI and display logic components which is called by multiple pages.

- store: The store contains core logic of the application. It shows the state of the application and handles mutation and action to interact within the application using Vuex.
- common: Contains utility function as well as constants used

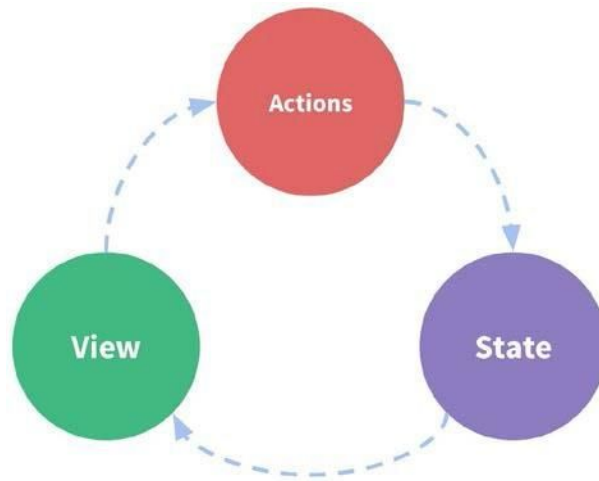


Figure 14. Source: Vuex Documentation

As shown in Figure 13. The view (components) will fire actions and the actions change the state. After that, as view is subscribed to the state (Observable Pattern), the UI (view) will be notified and updated. In doing this, the state becomes the single point of truth in the whole application, which increases the maintainability a lot and make it is easy to trace the change of UI.

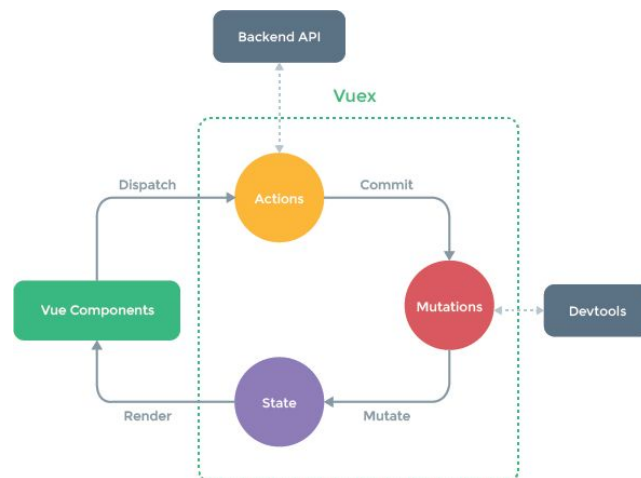


Figure 15. Source: Vuex Documentation

A more detailed diagram is shown in Figure 14. Typically, a vue component will dispatch an action. The action will communicate with backend API to do CRUD operations. After that, it will commit a mutation and the mutation will change the state. The change of state will be propagated to the component and component will render based on the data in the state. A devtool (e.g. Vuex.js devtools⁶) can be used to monitor the state for easy debugging.

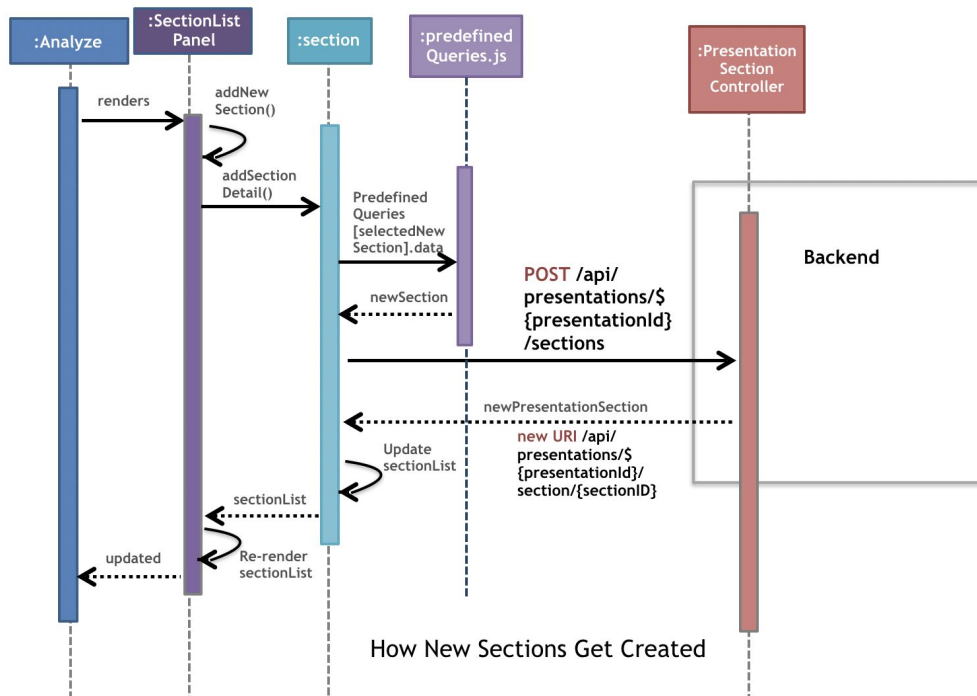


Figure 16: Example Sequence Diagram for Creation of New Section

A concrete illustration of the vuex workflow is shown in Figure 16, from the view component (SectionListPanel) dispatching an action in vuex, which is the section.js file, which then communicates with the backend via the POST API url. After the backend returns a new presentation section, the action method commits a mutation to change the sectionList, which updates the state. When the state of the program is updated, the view component is re-rendered.

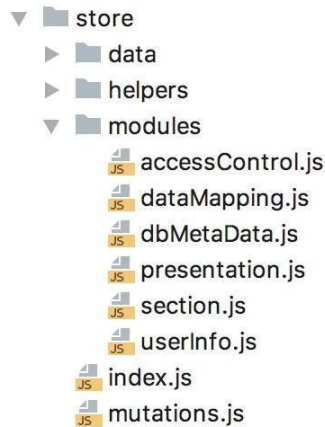


Figure 17

Indeed, in the store component, we have the following modules as shown in .

- ◆ `accessControl.js` is responsible for handling logic related to the access control of a presentation
- ◆ `dataMapping.js` is responsible for handling logic related to the mapping of CSV and built-in data schema
- ◆ `presentation.js` is responsible for handling logic related to the presentation
- ◆ `presentation.js` is responsible for handling logic related to the presentation section
- ◆ `userInfo.js` is responsible for handling logic related to the authentication

In summary, with the help of Vuex, the overall architecture of the front end ensures that the state can only be mutated predictably.

The components are designed to be highly reusable. The diagrams below break down each page (view) into different components.

⁶ Vuex.js devtools is available as a Chrome Extension

<https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjimejigglicpcpnannhbledajbpd?hl=>

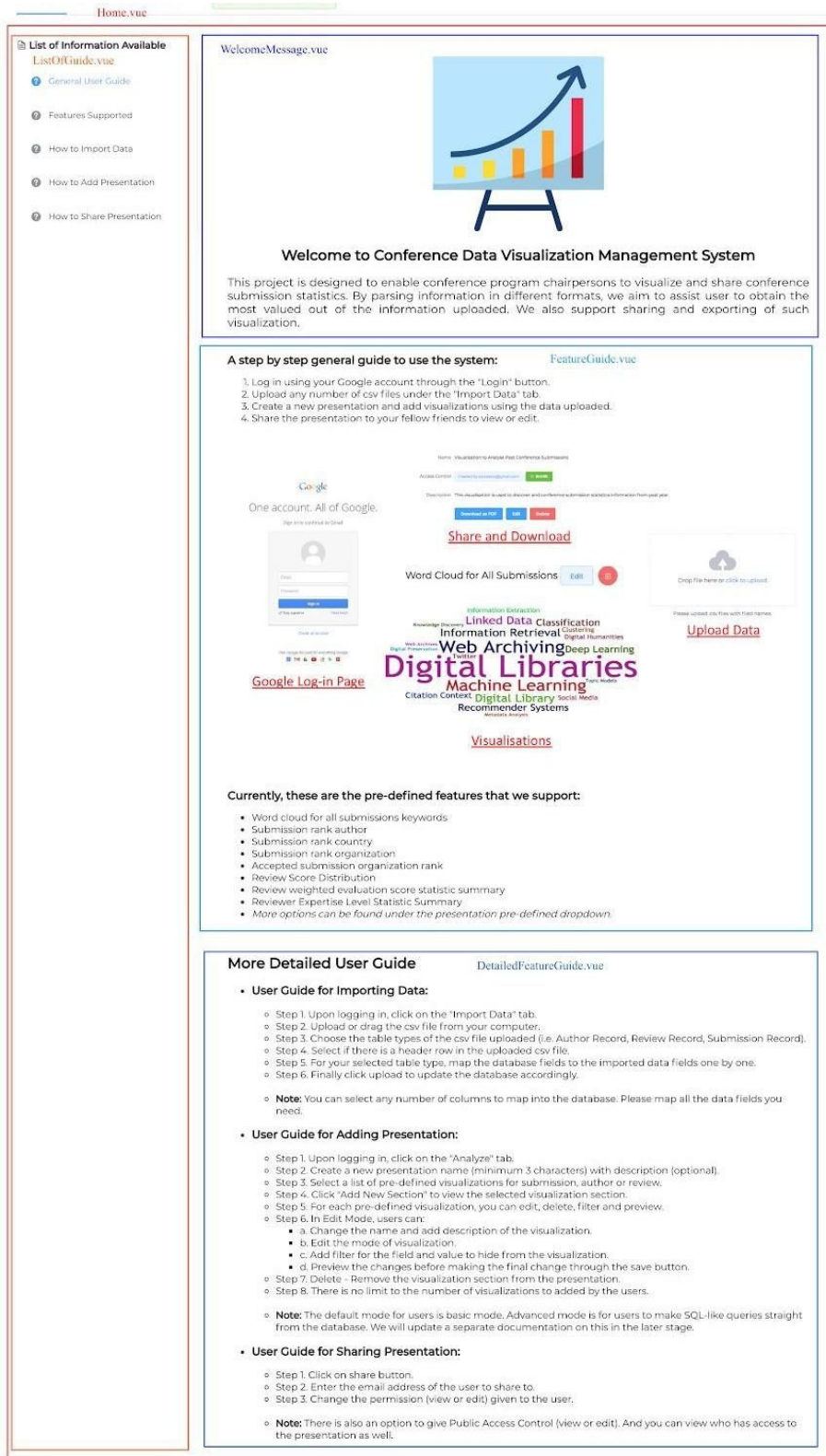


Figure 18: Breakdown of Home view

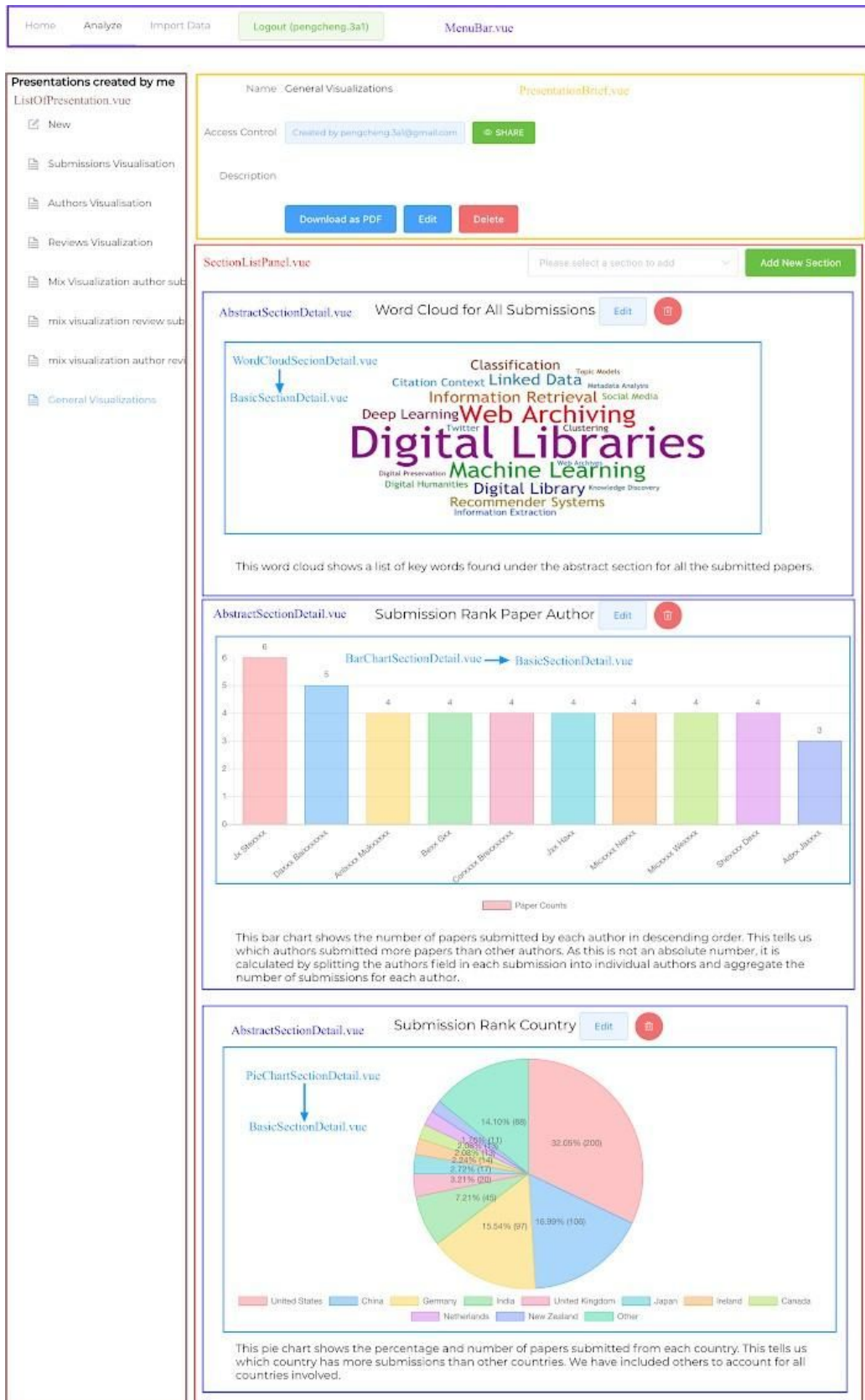


Figure 19: Breakdown of Analyse

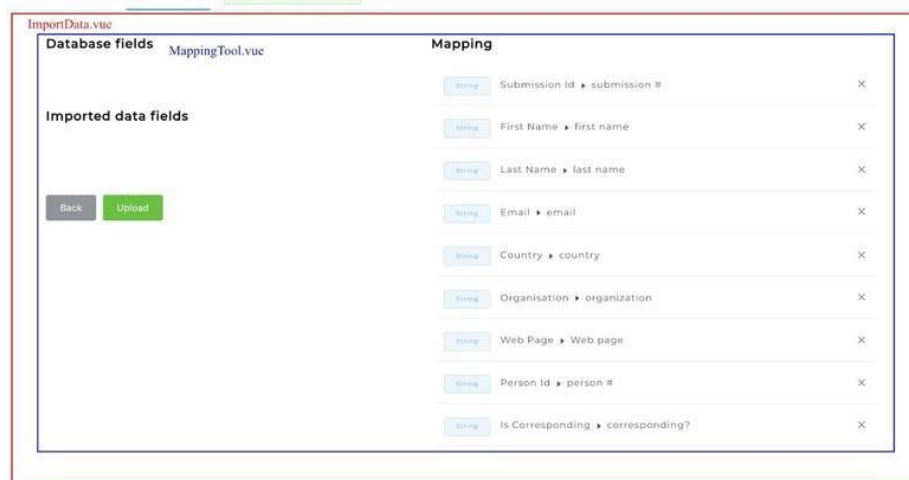


Figure 20: Breakdown of *ImportData* view

Figure 20 shows the breakdown of a specific visualization. The component outlined in blue are controlled by `BasicSectionDetail.vue`. The components outlined in orange are `BarChartSectionDetail.vue`. All the logic related to the title, description and filtering of the section are handled in the `BasicSectionDetail`. `BarChartSectionDetail` only cares about how to generate visualization based on given data. Such separation of concern tremendously increases the reusability of the frontend components. Any new type of visualization could be added with the help of `BasicSectionDetail`. Any existing visualization will remain untouched (**Open–Closed Principle**).

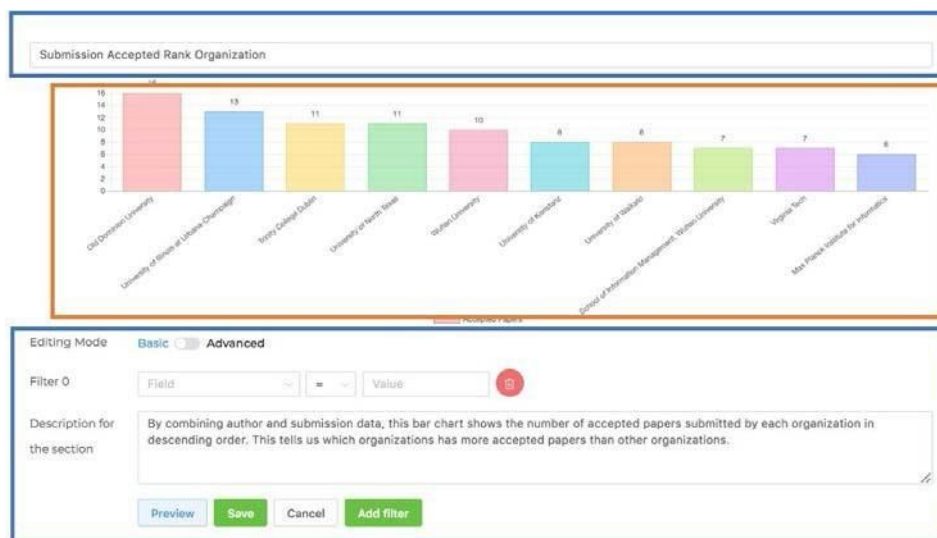


Figure 21

It should also be noted that the component `BarChartSectionDetail` is not specific to show “Submission Accepted Rank Organization”. Any visualization that is displayed in bar chart format can reuse the component.

Setting Up Development Environment

This is a step-by-step guide for setting up a development environment on your local machine. Using this environment, you can contribute to the project by working on features, enhancements, bug fixes, etc.

All the instructions in this document work for Linux, OS X, and Windows, with the following pointers:

- Replace `./gradlew` to `gradlew.bat` if you are using Windows.
- All the commands are assumed to be run from the root project folder, unless specified otherwise.
- When a version is specified for any tool, install that version instead of the latest version available.

Getting Started

To get started, you should have installed the latest version of IntelliJ on your local machine. Then please follow the steps.

We also recommend to download the latest version of VS Code for front-end development to make use of their great debugging and development tools.

Step 1: Get a Copy

Install Git and Clone the repo to your local machine.

Step 2: Install necessary tools and languages

1. Install JDK1.8
2. Install NodeJS 8.11.1
3. Install mysql5.6. You can use either of the following
 - a. For Mac users you can download from HomeBrew
 - b. Download GUI from <https://www.mysql.com/>
 - c. Install Sequel Pro from <https://www.sequelpro.com/> (optional)
4. Configure your IntelliJ by installing the following plugins
 - a. Google Cloud Tools
 - b. Vue.js (Optional)
5. Configure your VS Code by installing the following plugins
 - a. ESLint
 - b. Vue.js Dev Tools
 - c. Vetur

6. Install Google Cloud SDK (minimum version 222.0.0). Follow the directions given at <https://cloud.google.com/sdk/downloads>
 - a. Run the following command at the Google Cloud SDK directory
./install.sh --path-update true (Linux/OS X)
install.bat --path-update true (Windows)
 - b. Run a gcloud command (e.g. `gcloud version`) in order to verify that you can access the SDK from the command line
 - c. Run the following command to install App Engine Java SDK bundled with the Cloud SDK:

```
gcloud -q components install app-engine-java
```

7. Install npm. We suggest using NVM, which stands for Node Version Manager. You can follow instructions <https://www.npmjs.com/get-npm>
8. Navigate into `src/web/app` and run the following commands
 - a. `npm install`

Step 3: Setup Project specific settings and configuration

1. `src/main/resources/application-local.properties` for SQL server connection setup (If you are using your own MySQL server, you need to enter the correct connection string here)
2. Import the project into IntelliJ
 - a. Click Import Project (or File → New → Project from Existing Sources... if a project is open).
 - b. Select the local repository folder and click Open.
 - c. Select Import project from external model and then Gradle.
 - d. Click Next.
 - e. Check Use auto-import and uncheck Create separate module per source set.
 - f. Ensure Create directories for empty content root automatically is unchecked.
 - g. Ensure Use default gradle wrapper is selected.
 - h. Ensure for Gradle JVM: that a JDK 8 is selected.
 - i. Click Finish. Wait for the indexing process to complete.
 - j. You should see a dialog box with the message:
 - k. Frameworks detected: Google App Engine Standard
3. You can either run the server by either of the following:
 - a. Click the run button (a green arrow) to run the application
 - b. Run the application by issuing command `$./gradlew appengineRun`
 - c. Access the backend admin by accessing <http://localhost:8080>

4. Import the front-end code into VS Code by navigating into `src/web/app`
5. You can start the front end by the following commands:
 - a. `npm run serve`
 - b. Access the frontend application through <http://localhost:4040>

Step 4: Test the project

Here are two main components in this project's testing stage; front-end testing and backend testing.

- Frontend tests:
 - Run the following commands under root directory.
`$ cd src/web/app`
`$ npm run test:unit`
- Backend tests:
 - Run the following commands under root directory.
`$./gradlew check`

Step 5: Deploy to GCP (Optional)

You have two ways of deploying: locally and through continuous deployment.

- Local deployment:
 1. Make sure that npm is properly installed on your local machine. You can verify it by calling
`$ npm -v`
 2. Make sure that Google Cloud SDK is properly installed on your local machine. You can verify it by calling
`$ gcloud --version`
 3. Authorize your Google Cloud SDK locally through
`$ gcloud auth login`
 4. Make sure `application-prod.properties` contains correct configurations
 5. Modify the template with corresponding information.
 6. Build the frontend for production
`$ cd src/web/app`
`$ npm run build`
 7. Go to project root directory and run

```
$ ./gradlew appengineDeploy
```

- Continuous deployment:

Every time when master branch is updated, the new stable version will be automatically deployed to Google Cloud.

In order to enable this follow

<https://docs.travis-ci.com/user/deployment/google-app-engine> and update the credentials.

Appendix B - A Peek in the Data

Let's see how a conference management system looks like.

6	Jim Hahn. <i>The Navigation of Topic Space</i>			
8	Ali Shiri. <i>Methodological Considerations in Developing Cultural Heritage Digital Libraries: A Community-driven Framework</i>			
9	Nhu Van Nguyen, Mickaël Coustaty and Jean-Marc Ogier. <i>An adaptive document recognition system for lettrines</i>			
15	Wen Lou and Hui Wang. <i>BSOnto: A Binary Similarity Calculation Method on Ontology Fusion in Knowledge bases</i>			
23	Robert Allen and Yoonhwan Kim. <i>Semantic Modeling with Foundries</i>			
26	Lihong Zhou, Xinyu Lu and Tim Zijlstra. <i>Building a Theoretical Framework for the Development of Digital Scholarship Services in China's Universities</i>			
32	Ke Yuan, Liangcai Gao, Zhuoren Jiang and Zhi Tang. <i>Formula Ranking within an Article</i>			

Figure 1: Conference Management System

Figure 1 shows a snapshot of submitted papers with submission#, author name(s) followed by paper title. Figure 1 shows only minimal required information and operations.

However, for each submission there are multiple other data like author list with their affiliations, assigned reviewers for each submission, reviewers who may have a conflict of interest with a submission etc. In fact there are more than a dozen of CSV files generated (see Figure 2).



























-  assignment.csv
-  author.csv
-  bidding.csv
-  comment.csv
-  committee_topic.csv
-  committee.csv
-  conflict.csv
-  easychair_k...phrases.csv
-  metareview.csv
-  relevance.csv
-  review_field_score.csv
-  review_form_field.csv
-  review_requests.csv
-  review_score.csv
-  review_text.csv
-  review.csv
-  submission...ld_value.csv
-  submission_file.csv
-  submission...ld_track.csv
-  submission...rm_field.csv
-  submission_form_file.csv
-  submission_topic.csv
-  submission.csv
-  superseded_review.csv
-  track.csv
-  watchlist.csv

Figure 2: List of CSV files

submission #	first name	last name	email	country	organization
2	Laura	Kahn	lkahn@indiana.edu	United States	Indiana University Bloomington
3	Yi	Shen	yishen18@vt.edu	United States	Virginia Tech

In Figure 2, the structure of **author.csv** and **submission.csv** are shown in Figure 3 and 4. **author.csv**:

Figure 3: Structure of author.csv

submission.csv:

#	track #	track name	title	authors
2	3	Posters and Demos	Quantitative Approach for Coffee Rust, Production and Futures	Laura Kahn
3	3	Posters and Demos	Digital library System in Intelligent Infrastructure for Human-Centered Communities	Yi Shen

Figure 4: Structure of submission.csv

There are multiple conference management systems available and the conferences can choose which management system they want to use. The structure of the files may vary between different conference management systems. The two conference management systems whose data ChairVisE supports are EasyChair and SoftConf. Hence, any conference which uses EasyChair or SoftConf as its conference management system is supported by ChairVisE. Following is the Structure of the three files: **author.csv**, **review.csv** and **submission.csv** for EasyChair and SoftConf formats:

Author csv :

EasyChair:

Col 1: submission#

Col 2: first Name

Col 3: last name

Col 4: email

Col 5: country

Col 6 : organisation

Col 7 : webpage

Col 8 : person# (author's unique id in this file; everyone who registers to conference system has a unique id)

Col 9: corresponding (is the author corresponding author for the paper)

SoftConf:

Col 1: Submission Number

Col 2: Passcode

Col 3: Date Received

Col 4: Submission Type

Col 5: Track(s)

Col 6: Acceptance Status

Col 7: Contact Type

Col 8: First Name
Col 9: Last Name
Col 10: Email
Col 11: Affiliation
Col 12: Address 1
Col 13: Address 2
Col 14: Address 3
Col 15: City
Col 16: State/Province/Region
Col 17: Postal Code/Zip
Col 18: Country
Col 19: Biography
Col 20: Presenter

Submission csv: EasyChair:

Col1: Submission #
Col2: Track the submission is submitted to like a full paper or just a poster
Col3: Name for the track referred in col2 (string)
Col4: Title of the submission
Col5: authors of the associated submission
Col6: time submitted
Col7: time last updated Col8: ignore
Col9: keywords associated with submissions as put by the authors
Col10: accept/reject decision
Col11: acceptance/rejection mail sent to authors or not?
Col12: review sent in the mails or not?
Col13: abstract of the submission.

SoftConf:

Col1: Submission ID
Col2: Passcode
Col3: Title
Col4: Author Track
Col5: Track Chairs
Col6: Acceptance Status
Col7: Conditions
Col8: Track Recommendation
Col9: Abstract
Col10: Submission Date
Col12: Submission Type
Col13: Subject Area
Col14: Keywords
Col15: n: First Name (n:1 to 10)
Col16: n: Last Name
Col17: n: Email
Col18: n: Affiliation
Col19: n: Presenter
Col65: Main Contact Title
Col66: Main Contact Firstname
Col67: Main Contact Lastname
Col68: Main Contact Affiliation

Col69: Main Contact Affiliation Dpt
 Col70: Main Contact Job Function
 Col71: Main Contact Phone
 Col72: Main Contact Mobile
 Col73: Main Contact Fax
 Col74: Main Contact Email
 Col75: Main Contact Street Address
 Col76: Main Contact City
 Col77: Main Contact State/Province
 Col78: Main Contact Zipcode
 Col79: Main Contact Country
 Col80: Main Contact Biography
 Col81: Authors with Affiliations
 Col82: All Author Emails
 Col83: procTitle
 Col84: procShortTitle
 Col85: paperNumPages
 Col86: copyrightSig
 Col87: jobTitle
 Col88: orgNameAddress
 Col89: Implementation?
 Col90: Data?
 Col91: Dataset_Author
 Col92: Final Attachments OK
 Col93: Final Tags
 Col94: Final Notes

Review csv : EasyChair:

Col1 review#
 Col 2 submission#
 Col 3 review_assignment# (Each reviewer is given a number for each track he/she is reviewing. For example Animesh reviewed 2 different tracks but 3 papers in total- one from Track 1 and two papers from Track 2. He therefore has 2 uniques numbers assigned.)
 Col 4 reviewer name
 Col 5 field# . (Reviewer selects a field 1-5 to indicate expertise while submitting the review. For example 5: expert, 1: passing knowledge)
 Col 6 review_comments
 Col 7 "overall evaluation - score" (ignore)
 Col 8 overall evaluation score
 Col 9-12 subreviewer info (ignore)
 Col 13-14 Date and time of review submission
 Col 15 - recommendation for best paper

SoftConf:

Col1: Submission ID
 Col2: Submission Title
 Col3: Acceptance Status
 Col4: Submission Type
 Col5: Subject Area

Col6: Track
 Col7: First Submission Time
 Col8: Latest Modification Time
 Col9: APPROPRIATENESS
 Col10: CLARITY
 Col11: ORIGINALITY
 Col12: ORIGINALITY
 Col13: ORIGINALITY
 Col14: ORIGINALITY
 Col15: ORIGINALITY
 Col16: IMPACT OF THE
 SURVEY
 Col17: EMPIRICAL SOUNDNESS / CORRECTNESS
 Col18: COMPREHENSIVENESS / CORRECTNESS
 Col19: IMPLEMENTATION AND SOUNDNESS
 Col20: SOUNDNESS / CORRECTNESS
 Col21: THEORETICAL SOUNDNESS / CORRECTNESS
 Col22: MEANINGFUL COMPARISON
 Col23: SUBSTANCE
 Col24: IMPACT OF IDEAS OR RESULTS
 Col25: IMPACT OF ACCOMPANYING SOFTWARE
 Col26: IMPACT OF ACCOMPANYING DATASET
 Col27: EVALUATION
 Col28: MEANINGFUL COMPARISON Col29:
 IMPACT OF IDEAS OR RESULTS
 Col30: IMPACT OF ACCOMPANYING SOFTWARE
 Col31: IMPACT OF ACCOMPANYING DATASET
 Col32: RECOMMENDATION
 Col33: REVIEWER CONFIDENCE
 Col34: PRESENTATION FORMAT
 Col35: RECOMMENDATION FOR OUTSTANDING PAPER AWARD
 Col36: MENTORING
 Col37: AUTHOR RESPONSE
 Col38: REVIEW DATASET
 Col39: Significance of Rebuttal

Appendix C - Join Schematics

An ideal joined table (for example of author.csv and submission.csv) should look like the diagram in Figure 5.

author.submission #	first name	author.last name	author.email	author.country	submission.#	submission.trai	submission.track nat	submission..correspon
2	Laura	Kahn	lkahn@indiana.edu	United States	2	3	Posters and Demos	yes
3	Yi	Shen	yishen18@vt.edu	United States	3	3	Posters and Demos	yes

Figure 5: Ideal joined table of author.csv and submission.csv

Note:

1. We don't suggest that you join the CSV files and process. We encourage you to have an internal parsed representation of data being in a form which emulates this. For example, suppose you decide to use a DB/Store, then there should be a way to perform such join by using where author.submission # == submission .#
2. The basic requirement is to be able to do this given what fields are shared between various files. Hence, if your team decides to provide visualizations for conflict.csv as well and given 'author.#' is the common key, you need not have to write all the code again.
3. Furthermore, you can take one more step by having no association of column and the column header. Which means that author.csv with column 1 switched with column 3, should still be treated as a correct file with valid columns (i.e., valid column means a column with a valid header) and the system should still perform agnostically to this change.

Appendix D FAQs

1. I do not want to spend money on Google Cloud Platform. What should I do?

Google Cloud Platform like most cloud platform provides a \$300 credit for sign up for one year. You can do a new sign up from google cloud and enjoy all the benefits of google cloud platform for free. Please be considerate in using the resources so that you don't shoot up the \$300 limit.

Alternatively, you do not need to use Google Cloud Platform(GCP) credits. The purpose for using GCP is to use their App Engine for deployment and the MySQL database. But, as mentioned in Appendix A, you will be downloading Google Cloud SDK. The SDK provides a local server which you can use for development. Also, you can use (recommended) local MySQL as the database.

2. My GCP credits are being used up quickly. What should I do?

Make sure you disable the application and pause the use of resources when it is not in use. This will save your resource usage to a large extent. Also, make sure that you do not do your testing solely on the deployed application. This increases the traffic on the app engine and hence incurs more cost.

3. I do not have credits in my Google Cloud account. What should I do?

The module does not mandate you to deploy the application on cloud. It is for your learning purpose. If you do not have credits, you can keep on working on your localhost and show the demo in local as well. However, it will be appreciated if you can deploy your website to the google app engine.

4. I am more comfortable with using SQL server local installation than using google cloud server proxy. Can I continue to use it?

Yes you may use it.

If you are using your own MySQL server, you need to enter the correct connection string. Please find the details in Appendix 1.

5. I do not have google credits in my account and I intend to demo the app on localhost. What should I do about SQL?

The use of Google Cloud SQL proxy is a design choice. You may use local SQL server and any client for data access. We would advise you to use Google Cloud Platform for ease of use. In case you opt for local server, please make sure that you change all the database connections in the

application.

6. ./gradlew appengineRun command is not working. What can be the issue?

There might be a few reasons. Please check the following things:

If you are using Google Cloud Platform, Cloud SQL currently supports: MySQL 5.5, 5.6, and 5.7. PostgreSQL 9.6 and 11.1 Beta, ensure that your database is one of these versions.

7. If you are using Windows OS, the alternate command is: ./gradlew appengineRun

Please check /src/main/webapp/WEB-INF/appengine-web.xml. The manual scaling script should be commented while running on local.

While other properties in this file are ignored while running in local, manual scaling is not. You need to manually comment/uncomment it accordingly. Manual scaling improves performance in the deployed application and hence is a recommended setting while deploying.

8. I am facing MYSQL timezone issue while running the project.

In *application.local.properties*, modify *spring.datasource.url* value and add parameter *serverTimezone =UTC* as highlighted below

```
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/v  
iz? useSSL=false&serverTimezone =UTC
```

9. I am facing an error while running npm install.

Try deleting node_modules folder and package_lock.json and rerun npm install command