



CS3219 - Software Engineering Principles & Patterns

**Project Report
Group G05**

Fatin Nabilah Bte Suhaimi (A0187945J)

Jiang Jiahui (A0185088R)

Jordan Tan Rei Yao (A0226495W)

Tay Hui Chun (A0170109N)

1. Introduction	6
1.1 Purpose	6
1.2 Intended Audience	6
1.3 Project Scope	6
1.4 References	6
2. Overall Description	7
2.1 Product Perspective	7
2.2 Product Features	8
2.3 User Classes and Characteristics	9
2.4 Operating Environment	10
2.5 Design and Implementation Constraints	10
2.6 Assumptions and Dependencies	10
3. System Features - Functional Requirements	11
3.1 Usability Enhancements	11
3.1.1 Functional requirement 1.1	11
3.1.2 Functional requirement 1.2	11
3.1.3 Functional requirement 1.3	11
3.1.4 Functional requirement 1.4	11
3.1.5 Functional requirement 1.5	11
3.1.6 Functional requirement 1.6	12
3.2 Persistence & Management of Multiple Conference Data for Multiple Conferences	12
3.2.1 Functional requirement 2.1	12
3.2.2 Functional requirement 2.2	12
3.2.3 Functional requirement 2.3	12
3.2.4 Functional requirement 2.4	12
3.3 Flexible Data Schemes	13
3.3.1 Functional requirement 3.1	13
3.3.2 Functional requirement 3.2	13
3.3.3 Functional requirement 3.3	13
3.3.4 Functional requirement 3.4	13
3.4 User Interface Enhancements	13
3.4.1 Functional requirement 4.1	13
3.4.2 Functional requirement 4.2	14
3.4.3 Functional requirement 4.3	14
3.4.4 Functional requirement 4.4	14
3.5 Prioritization and Release Plan	14
4. External Interface Requirements	16
4.1 User Interfaces	16
4.1.1 Home Page (FR1.3, FR1.4, FR1.5, FR1.6)	16
4.1.2 Import Data (FR3.1, FR3.2, FR3.3, FR3.4)	18
4.1.3 Analyse Data	21
4.1.4 Manage Data (FR2.1, FR2.2, FR2.3, FR2.4)	23

4.1.5 Track Conferences (FR1.1, FR1.2)	24
4.1.6 User Guide	26
4.2 Hardware Interfaces	27
4.3 Software Interfaces	27
4.4 Communication Interfaces	27
5. Nonfunctional Requirements (NFR)	27
5.1 Performance Requirements	27
5.1.1 Non - Functional requirement 1.1	27
5.1.2 Non - Functional requirement 1.2	27
5.1.3 Non - Functional requirement 1.3	28
5.1.4 Non - Functional requirement 1.4	28
5.1.5 Non - Functional requirement 1.5	28
5.2 Data Requirements	28
5.2.1 Non - Functional requirement 2.1	28
5.2.2 Non - Functional requirement 2.2	28
5.2.3 Non - Functional requirement 2.3	28
5.3 Security Requirements	29
5.3.1 Non - Functional requirement 3.1	29
5.3.2 Non - Functional requirement 3.2	29
5.3.3 Non - Functional requirement 3.3	29
5.3.4 Non - Functional requirement 3.4	29
5.3.5 Non - Functional requirement 3.5	29
5.4 Software Quality Attributes	29
5.4.1 Non - Functional requirement 4.1	29
5.4.2 Non - Functional requirement 4.2	30
5.4.3 Non - Functional requirement 4.3	30
5.5 Prioritization and Release Plan	30
6. Introduction	32
7. Setting up	32
8. Development Process	33
8.1. Process Steps	33
8.2. Process Details	34
8.3. Schedule	34
9. Overall Architecture	36
10. Backend Design	37
10.1. Architecture	37
10.2. Common component	38
10.2.1. Data Transfer	38
10.2.2. Entity	39
10.2.3. Exception	42
10.2.4. Util	42

10.3. Storage component	43
10.3.1. General Structure	43
10.4. Logic component	44
10.4.1. General Structure	44
10.4.2. Details	45
10.5. UI component	47
10.5.1. Advice	47
10.5.2. Controller	48
10.5.3. General Structure	48
10.5.4. Details	49
10.6 Implementation of features	51
10.6.1. Personalized Dashboard (FR1.2, FR1.3, FR1.4, FR1.5, FR1.6)	51
10.6.2. Conference Notifications (FR1.1)	52
10.6.3. Management of multiple conference data (FR2.1, FR2.2, FR2.3, FR2.4)	53
10.6.4. Flexibility of data schemes (FR3.1, FR3.2, FR3.3, FR3.4)	54
11. Frontend Design	55
11.1. Architecture	55
11.1.1. Components	56
11.1.2. Actions	56
11.1.3. Mutations	56
11.1.4. States	56
11.1.5. Packages	57
11.2. Store	57
11.2.1. Data	57
11.2.2. Helpers	57
11.2.3. Modules	57
11.3. Views and Components	58
11.3.1. Analyze.vue	58
11.3.2. ConferenceSection.vue	59
11.3.3. Home.vue (FR1.3, FR1.4, FR1.5, FR1.6)	59
11.3.4. ImportData.vue (FR3.1, FR3.2, FR3.3, FR3.4)	60
11.3.5. ManageData.vue (FR2.1, FR2.2, FR2.3, FR2.4)	61
11.3.6. NewConference.vue (FR1.1, FR1.2)	62
11.3.7. NewPresentation.vue	63
11.3.8. PresentationSection.vue	63
11.3.9. UserGuide.vue	65
11.3.10. ViewConference.vue	65
12. Additional Suggestions	66
12.1. ChairHub	66
12.1.1. Networking	66
12.1.2. User Authentication	66
12.1.3. Conference Events	66
12.2. User Interface Improvements	67

12.2.1. Google Calendar Integration	67
12.2.2. Colour Coding Schemes	67
12.2.3. Visualisations	67
12.2.4. Step by step guide	68
12.3. Application Improvements	68
12.3.1. Testing	68
12.3.2. Use another front-end design toolkit to support responsive design.	68

Software Requirements Specification

1. Introduction

1.1 Purpose

This section of the report is intended to provide a comprehensive overview of the development of ChairVise 4.0, an online conference management system. The purpose and complete declaration for the system's development will be outlined. This document is specifically meant to be proposed for the approval of the CS3219 teaching team and as a guide for the development team during the implementation phase.

1.2 Intended Audience

This project is a prototype for the ChairVise conference management system and it is restricted within the premises of NUS. This has been implemented under the guidance of the university's professors and teaching assistants. This project is useful for conference chairpersons.

1.3 Project Scope

ChairVise is an online conference management system developed for conference chairs to simplify their conference management duties. This is accomplished by enabling conference chairs to manage their uploaded conference data easily. This data would serve as the basis for conference chairs to create convenient presentations in the form of visual charts and diagrams.

ChairVise will be hosted on a web server which can be accessed by mainstream browsers. The website seeks to provide conference chairs with an easy-to-use program on a user-friendly interface. All information about the system is contained in a database hosted on a web-server. The system also has the ability to send email notifications to conference chairs about upcoming conferences.

1.4 References

- [1] IEEE Software Engineering Standards Committee, "IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications", October 20, 1998.
- [2] Davis M A, "Just Enough Requirements Management: Where Software Development Meets Marketing", New York, Dorset House Publishing, 2005.

2. Overall Description

2.1 Product Perspective

The ChairVise 4.0 database system stores the following information:

- Record Data:
It includes author records, review records and submission records. Each record has a version. The submission and author records are linked together in the form of an author-submission record if they contain matching data fields.
- User Information:
It includes user emails and nicknames for the purpose of sending notifications, as well as information required to maintain access control to presentation data.
- Presentation Data:
It includes presentations generated by the user and has an access control to link the presentations to the users who created them.
- Conference Data:
It includes upcoming and past conference events that have been added by the user to the conference calendar on ChairVise.

2.2 Product Features

The major features of the ChairVise 4.0 database system are as shown in the entity-relationship diagrams (ERDs) below:

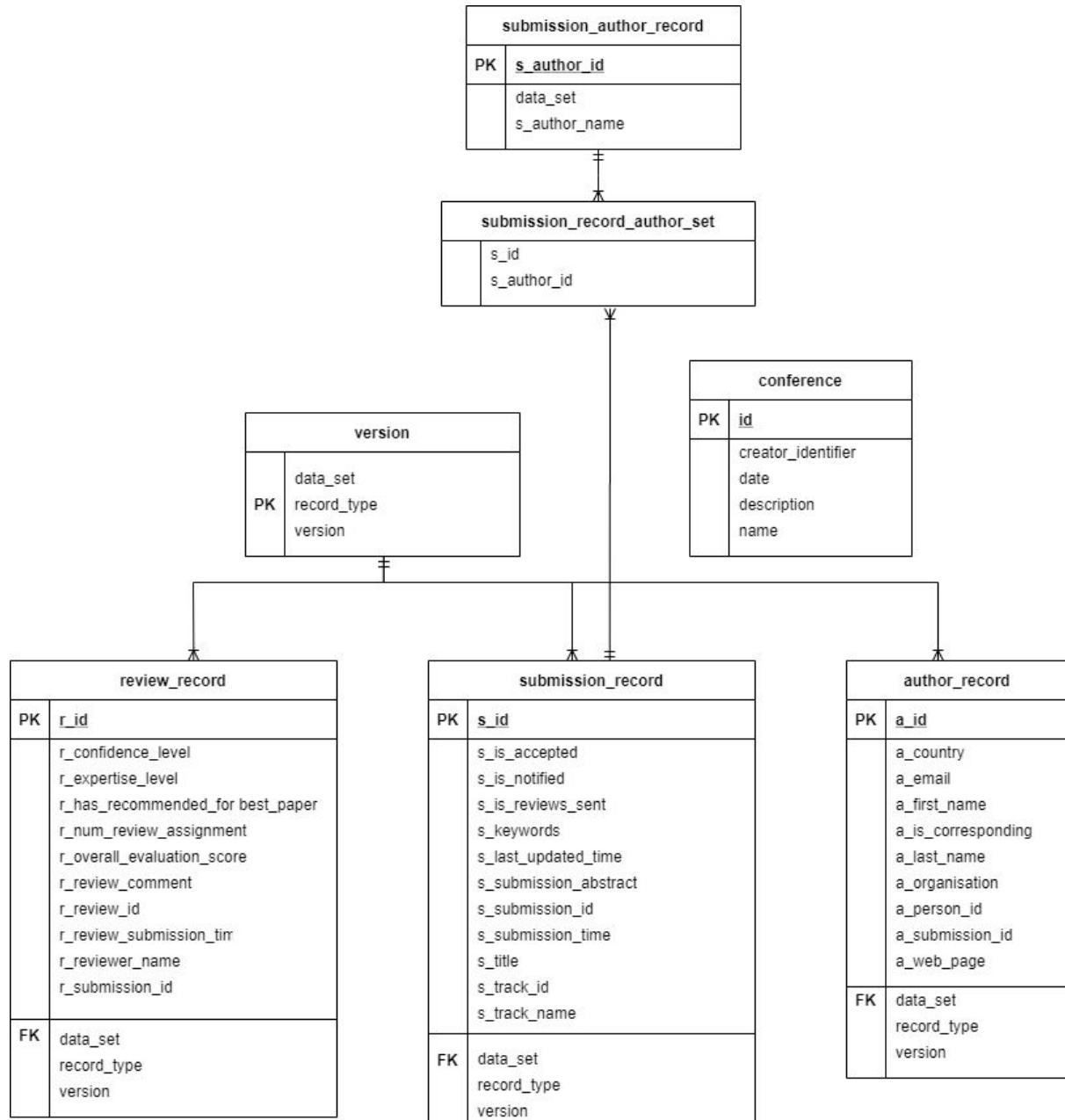


Figure 1. ERD for Record Data

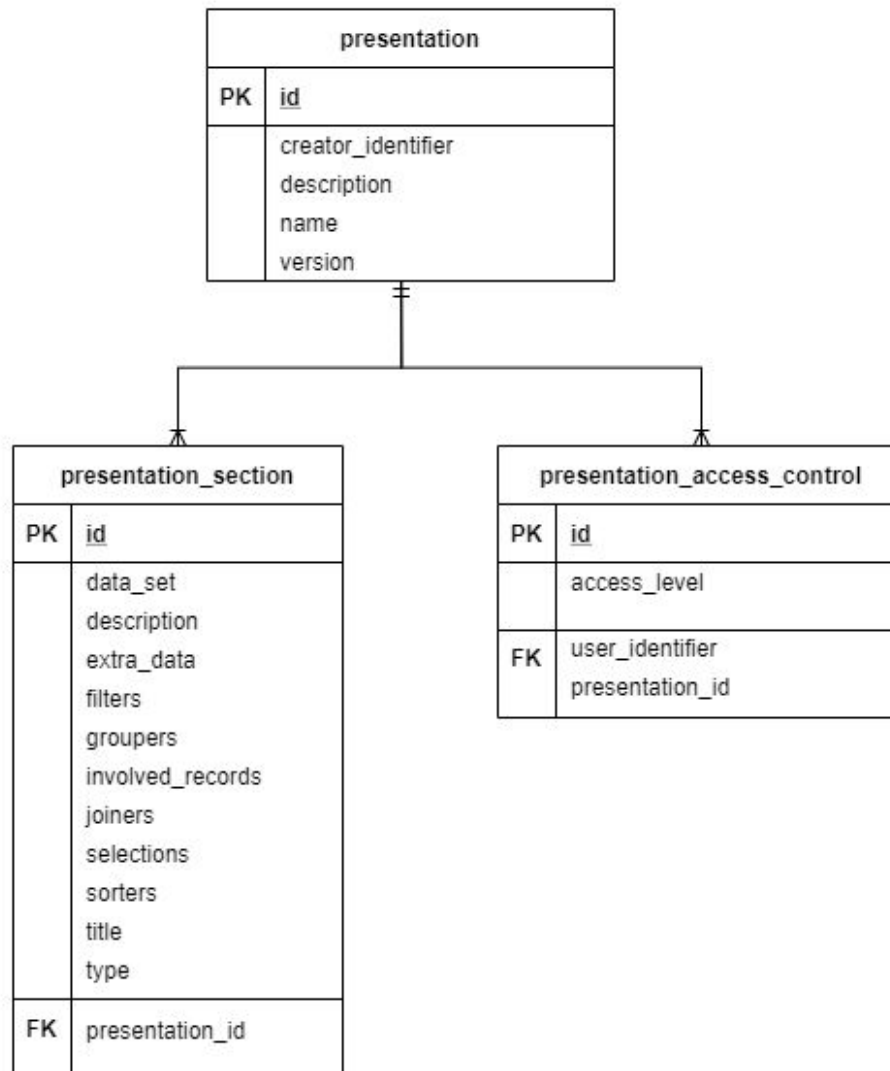


Figure 2. ERD for Presentation Data

2.3 User Classes and Characteristics

The system is catered for one specific type of user – conference program chairs. This user will be able to access all the features of the web application, namely, they are able to upload & manage conference submission data, and to generate various visualizations from the data. The web application is designed to be intuitive & convenient for these users, who may not have the time to learn to use a complex tool.

The visualizations generated may provide useful insights on top upcoming research focus and affinity to specific research areas, which could be very useful to the researchers in the field. Such insights could also help the organizers to make informed decisions to make the conference program more varied or more focused.

2.4 Operating Environment

The web application shall be accessible using any latest (or somewhat recent) mainstream browser on any operating systems.

The production database will be hosted on Google Cloud SQL.

The frontend & backend servers will be deployed on google cloud computing instances using Google App Engine (GAE).

2.5 Design and Implementation Constraints

As ChairVisE 4.0 is an extension of ChairVisE 3.0, its development process will continue to use the languages and frameworks used in ChairVisE3.0, which are:

- Vue + Vuex (Javascript)
- Spring Boot and Gradle (Java)
- Google App Engine
- Google Cloud SQL
- Continuous Development using Travis.

The data formats stored by the database and visualizations that can be generated are constrained by the author, review & submission records generated by the conference management systems. (EasyChair, SoftConf)

2.6 Assumptions and Dependencies

An assumption is that the user will be accessing the web application on the latest (or somewhat recent) version of a mainstream browser (Chrome, Firefox, Microsoft Edge, Opera or Safari). If an unsupported browser or an older version is used, the web application may not work as expected.

Another assumption is that the data formats for author, review, & submission records from the conference management systems (EasyChair, SoftConf) will remain the same. If their data formats change in the future, our system may need to be adjusted to cater to the new data formats.

Lastly, one assumption is that EasyChair & SoftConf will continue to be popular conference management systems in the future, so that our system continues to be relevant. If they someday become obsolete, or a new conference management system becomes more popular, our system will need to be adjusted to attract and retain users.

3. System Features - Functional Requirements

3.1 Usability Enhancements

3.1.1 Functional requirement 1.1

ID: FR1.1

TITLE: Conference Notifications

DESC: The system sends an email notification to the program chairperson when a conference event is approaching in 3 days time.

OBJ: Conference Notifications

3.1.2 Functional requirement 1.2

ID: FR1.2

TITLE: Conference - Style

DESC: The system must include relevant informations related to the upcoming conference within the email notification

OBJ: Conference Notifications

3.1.3 Functional requirement 1.3

ID: FR1.3

TITLE: Personalized Dashboard - Author Records

DESC: The system must show the number of author records uploaded in the dashboard.

OBJ: Personalized Dashboard

3.1.4 Functional requirement 1.4

ID: FR1.4

TITLE: Personalized Dashboard - Submission Records

DESC: The system must show the number of submission records uploaded in the dashboard.

OBJ: Personalized Dashboard

3.1.5 Functional requirement 1.5

ID: FR1.5

TITLE: Personalized Dashboard - Review Records

DESC: The system must show the number of review records uploaded in the dashboard.

OBJ: Personalized Dashboard

3.1.6 Functional requirement 1.6

ID: FR1.6

TITLE: Personalized Dashboard - Upcoming Conferences

DESC: The system must show the information for the next upcoming conference (if any) in the dashboard.

OBJ: Usability Enhancements

3.2 Persistence & Management of Multiple Conference Data for Multiple Conferences

3.2.1 Functional requirement 2.1

ID: FR2.1

TITLE: Show all Uploaded Data

DESC: The system is able to display the data that the program chair has uploaded in a separate view.

OBJ: Persistence and Management of Data

3.2.2 Functional requirement 2.2

ID: FR2.2

TITLE: Record Type

DESC: The system must be able to display the type of data record file (i.e. Author, Submission, Review record)

OBJ: Persistence and Management of Data

3.2.3 Functional requirement 2.3

ID: FR2.3

TITLE: Conference System Type Type

DESC: The system must be able to display which conference management system each record file belongs to.

OBJ: Persistence and Management of Data

3.2.4 Functional requirement 2.4

ID: FR2.4

TITLE: Version Number

DESC: The system must be able to display the version number of the record file.

OBJ: Persistence and Management of Data

3.3 Flexible Data Schemes

3.3.1 Functional requirement 3.1

ID: FR3.1

TITLE: Universal Support - Author Record

DESC: The system should be able to accept uploaded Author Record files from any conference management system - EasyChair, SoftConf and others.

OBJ: Flexible Data Schemes

3.3.2 Functional requirement 3.2

ID: FR3.2

TITLE: Universal Support - Submission Record

DESC: The system should be able to accept uploaded Submission Record files from any conference management system - EasyChair, SoftConf and others.

OBJ: Flexible Data Schemes

3.3.3 Functional requirement 3.3

ID: FR3.3

TITLE: Universal Support - Review Record

DESC: The system should be able to accept uploaded Review Record files from any conference management system - EasyChair, SoftConf and others.

OBJ: Flexible Data Schemes

3.3.4 Functional requirement 3.4

ID: FR3.4

TITLE: Standard Template

DESC: The system should provide a standard template for the program chair to map columns from the uploaded dataset to the template.

OBJ: Flexible Data Schemes

3.4 User Interface Enhancements

3.4.1 Functional requirement 4.1

ID: FR4.1

TITLE: Meaningful Visualizations

DESC: The visualizations created by the program chair must present data information in a readable and informative manner that is meaningful and provides value to the reader.

OBJ: User Interface Enhancements

3.4.2 Functional requirement 4.2

ID: FR4.2

TITLE: Data Processing

DESC: The system should correctly capture and interpret the data columns in the codebase and visualizations.

OBJ: User Interface Enhancements

3.4.3 Functional requirement 4.3

ID: FR4.3

TITLE: Relevant Visualizations

DESC: The system should present the user with a list of visualizations to choose from that only include those that can be created from the data files uploaded.

OBJ: User Interface Enhancements

3.4.4 Functional requirement 4.4

ID: FR4.4

TITLE: Presentation Layout

DESC: The system should allow the user to render different layouts for presentation according to the program chair's preferences

OBJ: User Interface Enhancements

3.5 Prioritization and Release Plan

FR1. Usability Enhancements	Priority	Deadline	Sprint
Objective: Conference Notifications			
FR1.1 - Conference Notifications	High	Week 7	1
Objective: Personalized Dashboard			
FR1.2 - Personalized Dashboard (Style)	High	Week 7	1
FR1.3 - Personalized Dashboard (Author Records)	High	Week 7	1
FR1.4 - Personalized Dashboard (Submission Records)	High	Week 7	1
FR1.5. - Personalized Dashboard (Review Records)	High	Week 7	1
FR1.6 - Personalized Dashboard (Upcoming Conferences)	High	Week 7	1

FR2. Persistence & Management of Multiple Conference Data for Multiple Conferences	Priority	Deadline	Sprint
Objective: Management of multiple conference data			
FR2.1 - Show all Uploaded Data	High	Week 9	2
FR2.2 - Record Type	High	Week 9	2
FR2.3 - System Type	High	Week 9	2
FR2.4 - Version Number	High	Week 9	2

FR3. Flexible Data Schemes	Priority	Deadline	Sprint
Objective: Flexibility of Data Types			
FR3.1 - Universal Support - Author Record	Medium	Week 11	3
FR3.2 - Universal Support - Submission Record	Medium	Week 11	3
FR3.3 - Universal Support - Review Record	Medium	Week 11	3
FR3.4 - Standard Template	Medium	Week 11	3

FR4. User Interface Enhancements	Priority	Deadline	Sprint
Objective: Improvements to Visualizations			
FR4.1 - Meaningful Visualizations	Low	-	-
FR4.2 - Data Processing	Low	-	-
FR4.3 - Relevant Visualizations	Low	-	-
FR4.4 - Presentation Layout	Low	-	-

4. External Interface Requirements

4.1 User Interfaces

4.1.1 Home Page (FR1.3, FR1.4, FR1.5, FR1.6)

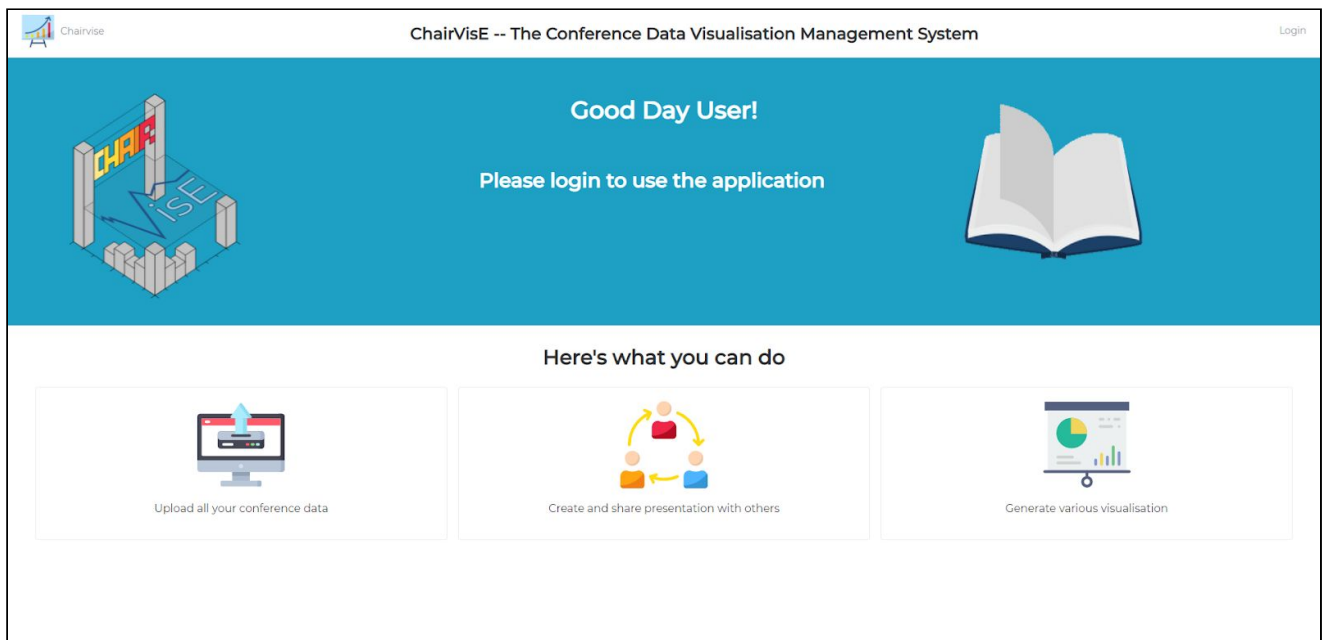
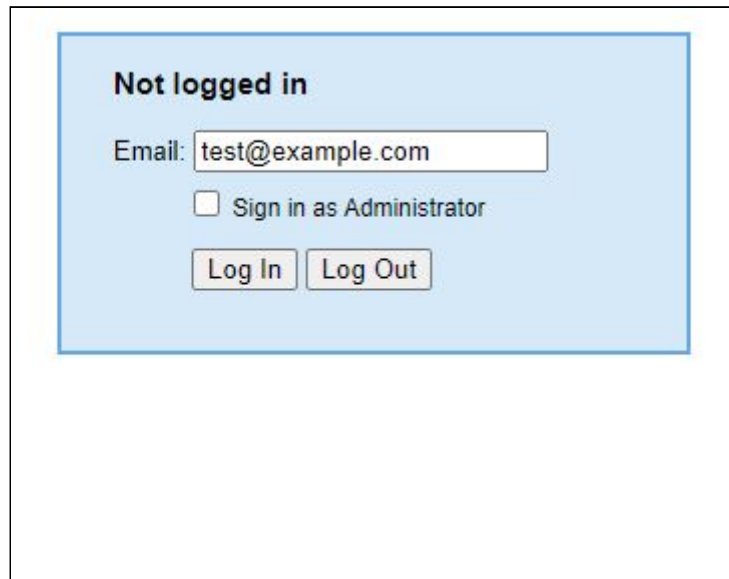


Figure 3. Home Page of ChairVisE

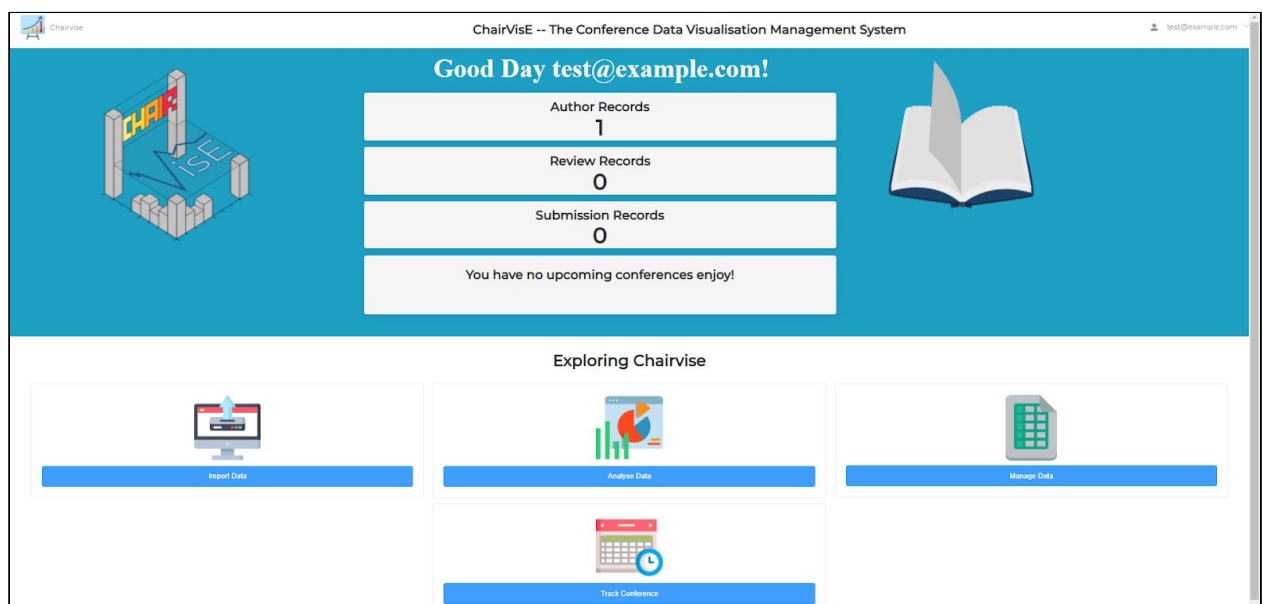
Figure 3 shows the landing page of ChairVisE when the user is visiting the web application for the first time. The user is greeted with a message and he is requested to start logging in in order to use the application. As such, the buttons below are disabled until he logs in successfully.



A screenshot of a login form titled "Not logged in". It features an email input field containing "test@example.com", a checkbox for "Sign in as Administrator", and two buttons labeled "Log In" and "Log Out".

Figure 4. Login View

When the user clicks on the “Login” button at the top right corner of the home page, he will be directed to the view as shown in Figure 4. The user can log in using his email address.

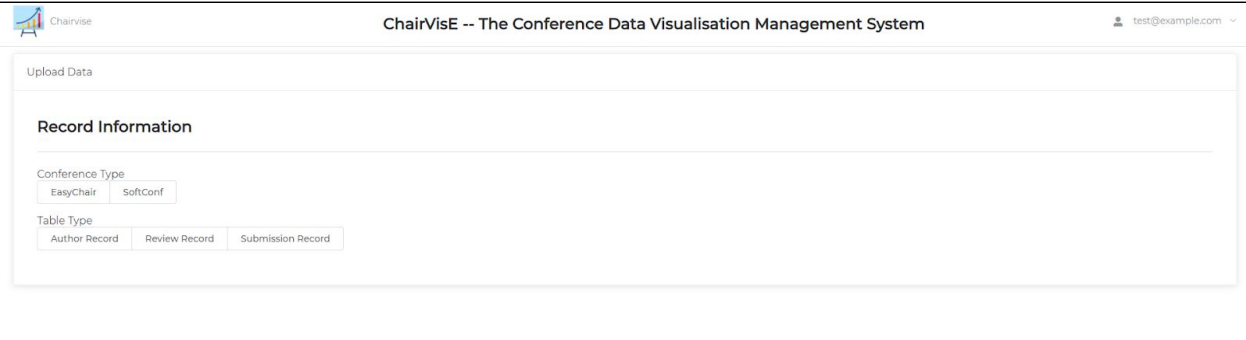


A screenshot of the authenticated home page of ChairVisE. The header includes the system name "ChairVisE -- The Conference Data Visualisation Management System" and the user "test@example.com". The main content area has a blue background with a 3D bar chart on the left and a book icon on the right. A central white box displays statistics: "Author Records 1", "Review Records 0", and "Submission Records 0", followed by the message "You have no upcoming conferences enjoy!". Below this is a section titled "Exploring ChairvisE" with four interactive tiles: "Import Data" (with a monitor icon), "Analysis Data" (with a bar and pie chart icon), "Manage Data" (with a document icon), and "Track Conference" (with a calendar icon).

Figure 5. Home Page of ChairVisE (Authenticated)

After the user successfully logs in to his account, the following buttons are enabled: Import Data, Analyse Data, Manage Data and Track Conference. The user can click on any of the said buttons to either upload conference data, analyse conference data via visualizations, manage uploaded data or track his conferences. On top of that, the user is also greeted with a message in the middle of the screen, which serves as a dashboard, informing the user the cumulative conference record files he has uploaded to date, as well as the upcoming conferences he has to manage.

4.1.2 Import Data (FR3.1, FR3.2, FR3.3, FR3.4)



ChairVisE

ChairVisE -- The Conference Data Visualisation Management System

test@example.com

Upload Data

Record Information

Conference Type

EasyChair SoftConf

Table Type

Author Record Review Record Submission Record

Figure 6. Upload Data View

Figure 6 shows what the user will see when he clicks on the “Import Data” button in the home page. In this view, the user has two options to the types of conference data he can upload to the system, EasyChair and SoftConf. For each Conference Management System, he can upload any of the three types of record file - Author Record, Review Record, Submission Record.

In Figure 7 below, upon selecting a particular conference type and record file type, the user can also include additional information about the files he wants to upload, i.e. header, predefined mapping and version of file. Header is about whether the record file has column headers included in it. Predefined Mapping is about using a set of predefined mapping between record files and the columns stored in the database. Version refers to the version of the file.

ChairvisE

ChairVisE -- The Conference Data Visualisation Management System

test@example.com

Upload Data

Record Information

Conference Type

EasyChair SoftConf Others

Table Type

Author Record Review Record Submission Record

Mapping Information

Note that the following column headers are expected to be in your csv file:

"submission #", "first name", "last name", "email", "country", "organization", "Web page", "person #", "corresponding?"

Version Information

Version

Input Version

Figure 7. Additional Upload Information

ChairvisE

ChairVisE -- The Conference Data Visualisation Management System

test@example.com

Mapping

String	Submission Id ▶ submission #	X
String	First Name ▶ first name	X
String	Last Name ▶ last name	X
String	Email ▶ email	X
String	Country ▶ country	X
String	Organisation ▶ organization	X
String	Web Page ▶ Web page	X
String	Person Id ▶ person #	X
String	Is Corresponding ▶ corresponding?	X

Database fields

Imported data fields

Upload Back

Figure 8. Mapping Tool View when an “EasyChair” author record is uploaded (mapping applied automatically)

ChairVisE -- The Conference Data Visualisation Management System

test@example.com

Mapping

String Submission Id ▶ Submission Number X

String First Name ▶ First Name X

String Last Name ▶ Last Name X

String Email ▶ Email X

String Country ▶ Country X

String Organisation ▶ Affiliation X

Database fields

Web Page Person Id Is Corresponding

Imported data fields

Passcode Date Received Submission Type Track(s)

Acceptance Status Contact Type Address1 Address2 Address3

City State/Province/Region Postal Code/Zip Biography Presenter

Upload Back

Figure 9. Mapping Tool View when a “SoftConf” author record is uploaded (mapping applied automatically)

ChairVisE -- The Conference Data Visualisation Management System

test@example.com

Mapping

No mapping specified!

Database fields

Submission Id First Name Last Name Email Country Organisation Web Page

Person Id Is Corresponding

Imported data fields

submission # first name last name email country organization web page

person # corresponding?

Upload Back

Figure 10. Mapping Tool View when “Other” conference management type & “Header” information is selected (no mapping applied)

ChairVisE -- The Conference Data Visualisation Management System

test@example.com

Mapping

No mapping specified!

Database fields

Submission Id First Name Last Name Email Country Organisation Web Page

Person Id Is Corresponding

Imported data fields

Column 1 Column 2 Column 3 Column 4 Column 5 Column 6 Column 7

Column 8 Column 9

Upload Back

Figure 11. Mapping Tool View when “Other” conference management type is selected & “Header” information is not selected (no mapping applied)

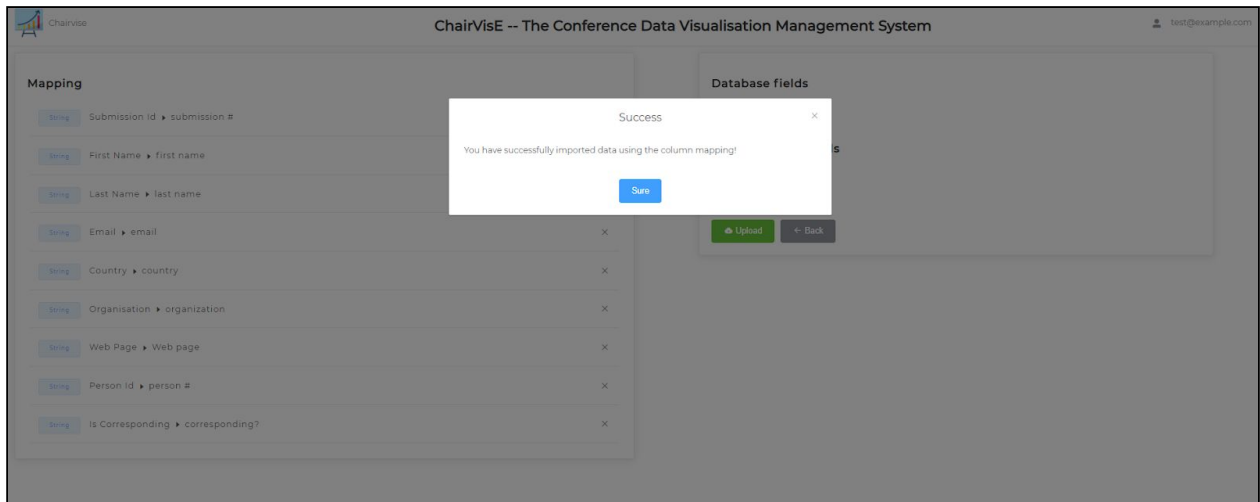


Figure 12. After successfully uploaded a record file

4.1.3 Analyse Data

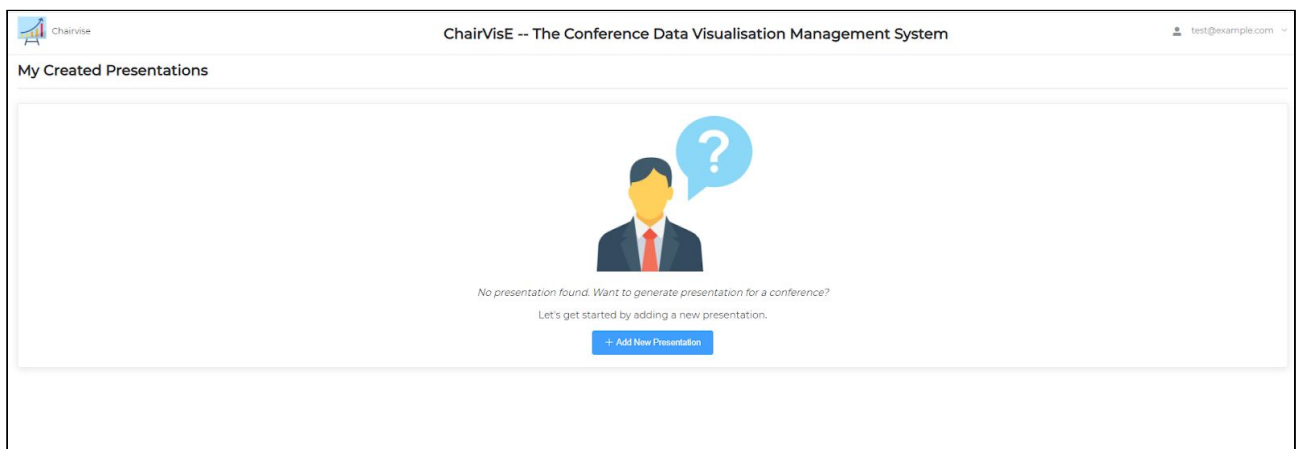


Figure 13. Analyse Data

Figure 13 shows the view that the user will see when he clicks on the “Analyse Data” button back in the home page. In this view, the user is able to view the presentations (visualizations) that he had generated previously. However, since this user does not have any presentations, he can proceed to add a new presentation via the “Add New Presentation” button in the middle.

The screenshot shows the 'Create New Presentation' form in the ChairVisE system. The header includes the ChairVisE logo, the title 'ChairVisE -- The Conference Data Visualisation Management System', and a user profile icon with the email 'test@example.com'. The form has two input fields: 'Name' with a placeholder 'Enter name' and 'Description' with a placeholder 'Enter description'. A blue 'Save' button with a checkmark icon is at the bottom left of the form.

Figure 14. Create New Presentation View

Figure 14 shows what happens when the user clicks on the “Add New Presentation” button. In this view, the user can create a new presentation by typing the name of the presentation and description.

The screenshot shows the 'My Created Presentations' view. The header is the same as Figure 14, but with a '+ Add New Presentation' button on the right. Below the header is a table with one row of data.

	Presentation12
#1	abcd

Figure 15. My Created Presentations View

After creating a new presentation, the presentation appears in the user’s records and the user can proceed to edit presentation details such as adding visualizations, share the presentation, edit it or delete it.

Figure 16 below shows the view for editing the presentation details.

The screenshot shows the 'Editing Presentation Details' view. The header is the same as previous figures. The main content area has a 'Presentation Details' section with fields for 'Name' (filled with 'Presentation12'), 'Access Control' (filled with 'Created by test@example.com'), and 'Description' (filled with 'abcd'). Below these fields are two buttons: 'PDF' and 'Powerpoint'. To the right of these buttons are three buttons: 'Share', 'Edit', and 'Delete'. Below the 'Presentation Details' section is a large area with a question mark icon and the text 'No section found. Want to visualise data? Let's get started by adding a new section on the right side of the panel.' To the right of this area is an 'Add section' panel with a dropdown menu 'Please select a section to add' and an 'Add New Section' button.

Figure 16. Editing Presentation Details View

4.1.4 Manage Data (FR2.1, FR2.2, FR2.3, FR2.4)

ChairvisE -- The Conference Data Visualisation Management System

Select the criteria below to specify dataset to display

Record Type Record Version Conference

Import Data

Figure 17. Manage Data View

Figure 17 shows the view that the user will see when he clicks on the “Manage Data” button in the home page. In this view, the user can view and manage the data that he has uploaded previously by specifying the record file type (Author, Review, Submission), file version, conference management system.

ChairvisE -- The Conference Data Visualisation Management System

Select the criteria below to specify dataset to display

AuthorRecord 1 EasyChair

Import Data

id	version	type	submissionid	firstName	lastName	email	country	organisation	webPage	personid	isCorresponding
625	1	AuthorRec	2	12013211511...	11512612611...	fradinalen...	United Sta...	Indiana Uni...		465	yes
626	1	AuthorRec	3	13413913211...	12115132123...	tyreicegari...	United Sta...	Virginia Tec...		90	yes
627	1	AuthorRec	4	12611911513...	12712913412...	leasimoto...	Pakistan	Informatio...		163	no
628	1	AuthorRec	4	13312312713...	11619118133...	simritabed...	Pakistan	Informatio...		164	yes
629	1	AuthorRec	4	12711513212...	11611512121...	mariadejes...	Pakistan	Informatio...		165	no
630	1	AuthorRec	4	11511512613...	11813513212...	aalyciadur...	United Kin...	University ...		166	no
631	1	AuthorRec	4	13612912412...	12211513213...	vojmirhartl...	Saudi Arab...	King Abdul...		167	no
632	1	AuthorRec	5	12413511811...	12012913312...	jude-jasso...	United Sta...	University ...		168	yes
633	1	AuthorRec	5	12411913313...	12612913212...	jesslinlorio...	United Sta...	University ...		169	no
634	1	AuthorRec	5	12813512771...	126119137139	numaalew...	United Sta...	Chattanooc...		170	no
635	1	AuthorRec	6	13211512611...	12513212311...	ralenkribb...	United Sta...	University ...		39	yes
636	1	AuthorRec	7	13413513211...	12712912313...	turekamoit...	United Sta...	Microsoft		171	yes
637	1	AuthorRec	7	13211913812...	11512512512...	rexineakkn...	United Sta...	Microsoft		172	yes
638	1	AuthorRec	7	11913212312...	12512913411...	erindakote...	United Sta...	Microsoft		173	yes
639	1	AuthorRec	8	13612912412...	11713213512...	vojmircru...	Canada	University ...		92	yes
640	1	AuthorRec	9	13011913413...	11512611613...	pettaalbre...	France	Lab L3I, Uni...		376	yes
641	1	AuthorRec	9	121115136132...	11812913513...	gavrielado...	France	Laboratoire...		377	yes
642	1	AuthorRec	9	121115128121...	11811512612...	gangelrida...	France	University ...		75	yes
643	1	AuthorRec	10	12611513412...	12212312814...	latoyrahinz...	Germany	Max Planck...		174	yes
644	1	AuthorRec	10	13211512412...	12011912611...	rajohnfeld...	Germany	Max Planck...		175	no

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

Figure 18. View Uploaded Data

Figure 18 shows a sample view when the user is requesting the web application to display all version 1 AuthorRecord files from EasyChair that he has uploaded.

4.1.5 Track Conferences (FR1.1, FR1.2)

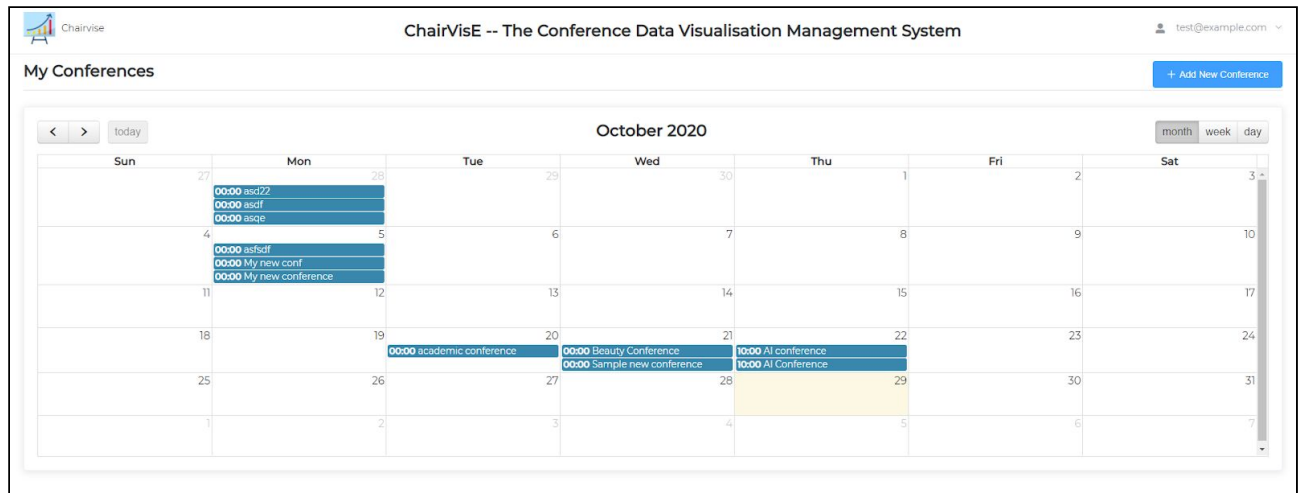


Figure 19. View My Conferences (Month)

Figure 19 shows what the user will see when he clicks on the “Track Conferences” button in the home page. He will be able to view all the conferences he has added previously in a calendar view that can be changed to show in month, week and day views. The user can also add a new conference by clicking on the “Add New Conference” button at the top right corner just above the calendar.

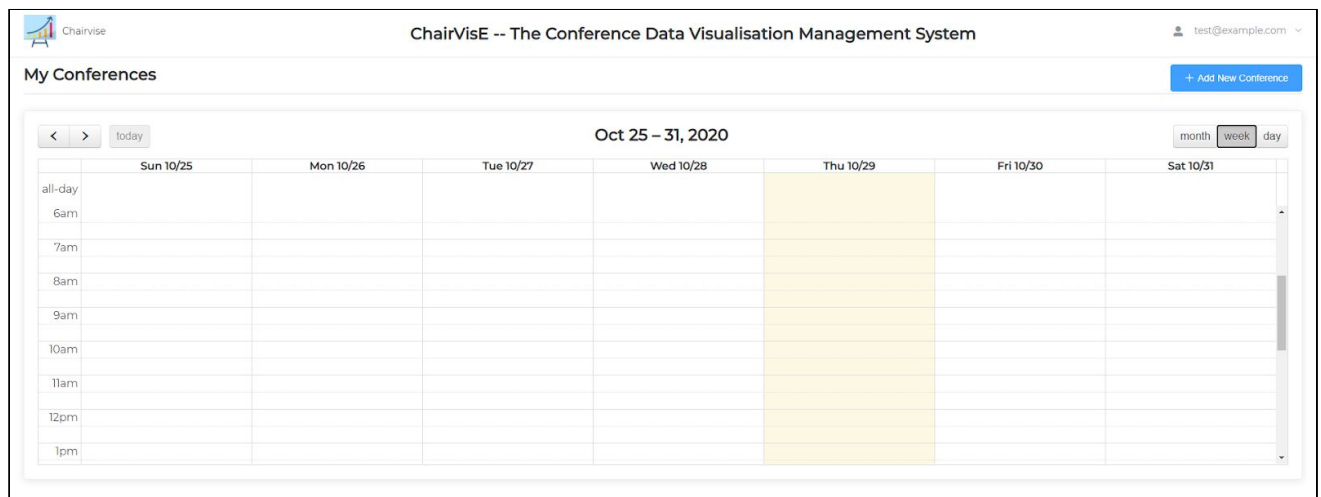


Figure 20. View My Conferences (Week)

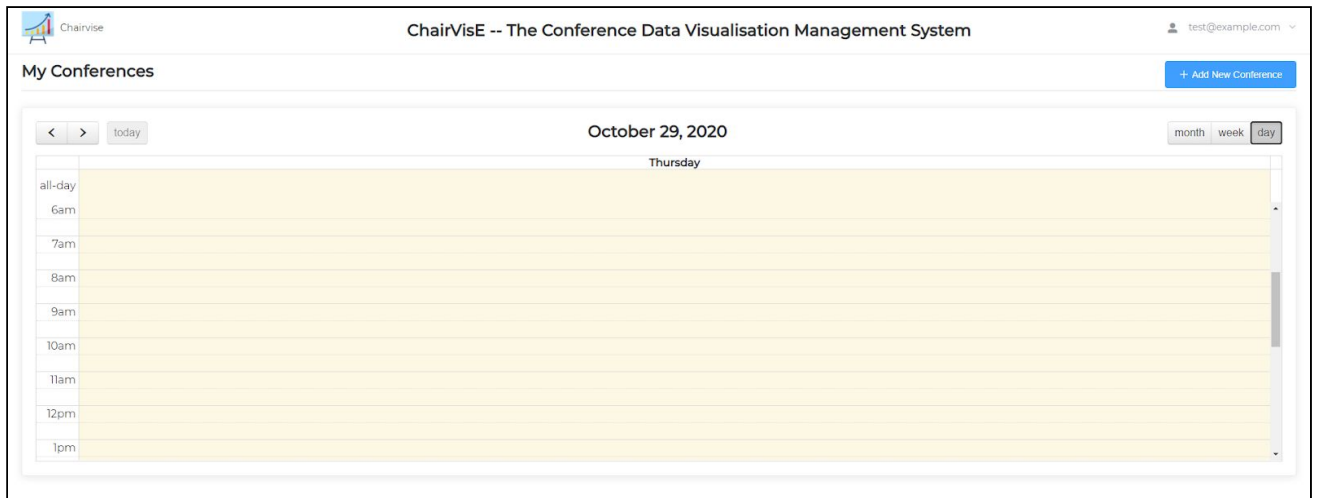


Figure 21. View My Conferences (Day)

The screenshot shows the 'Add New Conference' form in the ChairVisE system. The form has three main input fields: 'Name' (with a placeholder 'Enter name'), 'Description' (with a placeholder 'Enter description'), and 'Conference Date'. The 'Conference Date' field is currently selected, and a date picker is open, displaying the month of October 2020. The date picker shows a grid of days, with the 29th highlighted. Below the date picker, there are 'Now' and 'OK' buttons. A red border highlights the 'Select date and time' button and the 'Please select a conference date and time' message.

Figure 22. Add New Conference View

After clicking on the “Add New Conference” button, the user will need to fill in the fields as shown in Figure 22. The conference name, description and date.

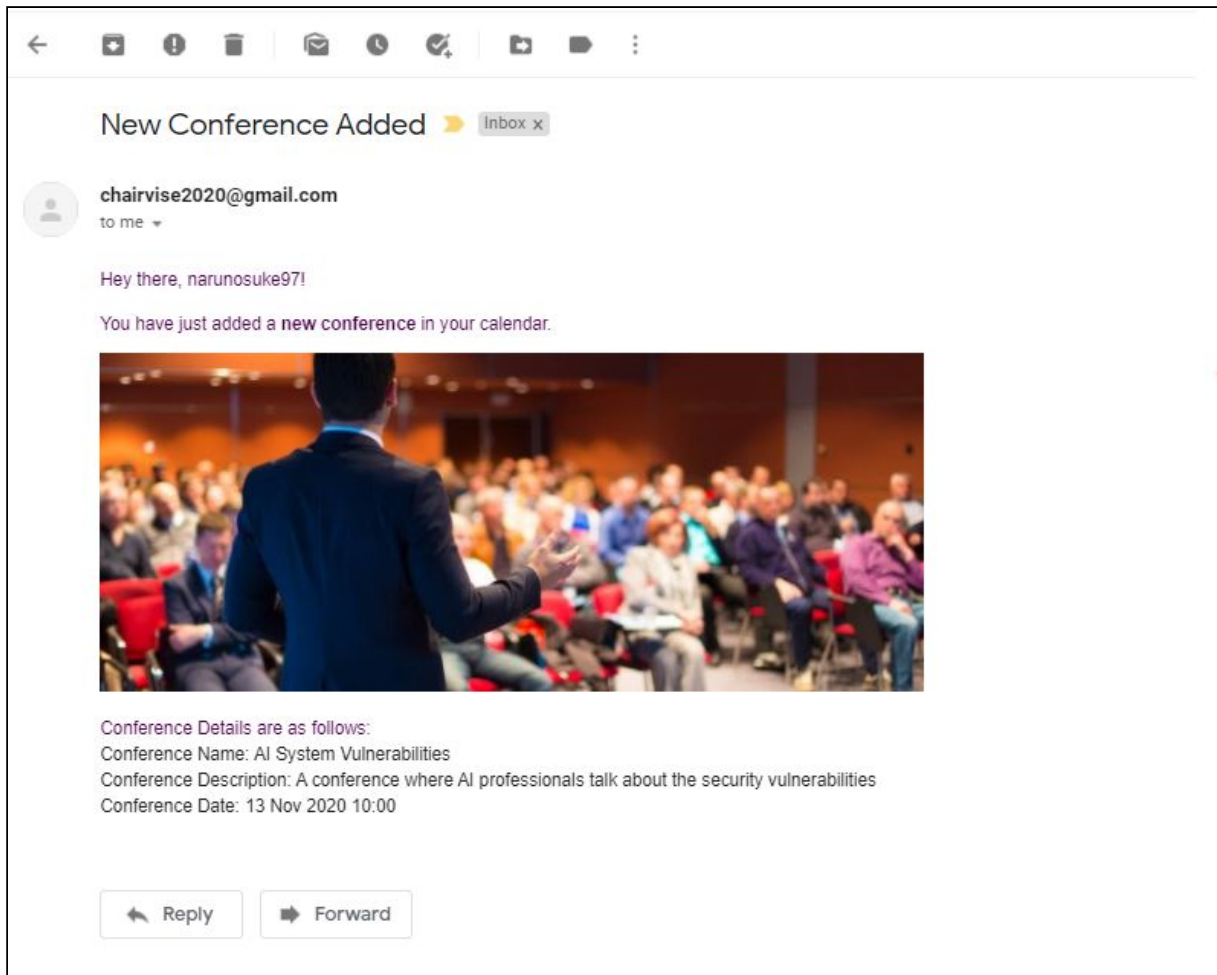


Figure 23: Screenshot of email notification when new conference is added

4.1.6 User Guide

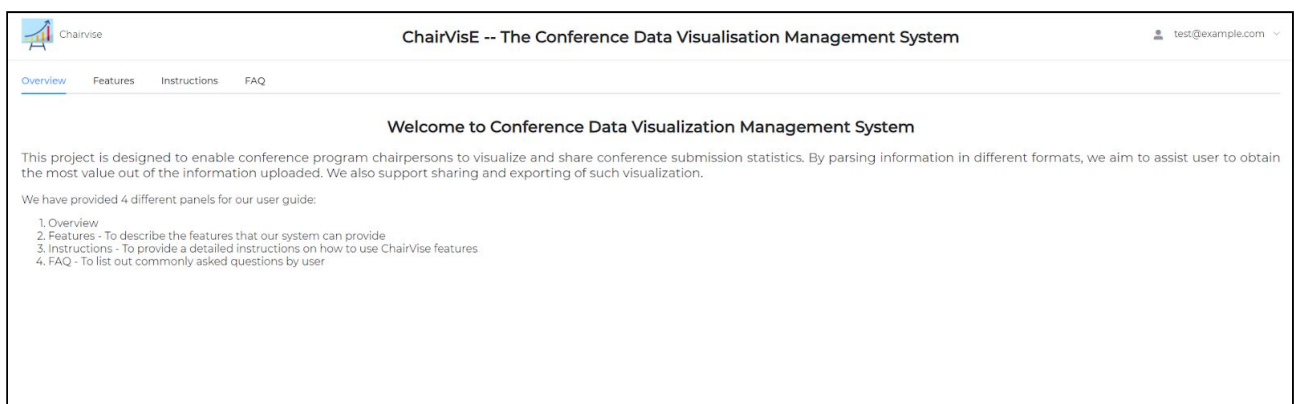


Figure 24. User Guide View

At the top right hand corner of the screen which shows the user's account, the user can also access shortcuts to "My Presentations" and "My Conferences" which will direct the user to view his presentations and conferences (Figure 15 and Figure 19). There is also a

user guide link for the user to have a walkthrough guide on how to use ChairVisE's features and common Frequently Asked Questions.

4.2 Hardware Interfaces

The software interface for ChairVise 4.0 consists of a browser which supports HTML & Javascript. ChairVise 4.0 does not have any direct hardware interfaces and therefore, does not have any other designated hardware. The underlying operating system on the web server handles the hardware connection to the database server.

4.3 Software Interfaces

To save conference data uploaded by users and presentations created by users, we have chosen the MySQL database. The interactions between the database and the web server consists of conference data creation, reading and deletion operations. In addition, via the Thymeleaf engine implementation on the Spring Framework, the web server sends conference notifications to the user via email.

4.4 Communication Interfaces

This project supports all types of web browsers. As they rely heavily on each other, coordinating the interactions between the various components of ChairVise 4.0 is crucial. However, in what way the interactions take place is not important to the system and is instead managed by the underlying operating systems for the web server.

5. Nonfunctional Requirements (NFR)

5.1 Performance Requirements

5.1.1 Non - Functional requirement 1.1

ID: NFR1.1

TITLE: Supported Browsers

DESC: The website should be able to function on any mainstream browser (i.e. Chrome, Firefox, Microsoft Edge, Opera and Safari).

5.1.2 Non - Functional requirement 1.2

ID: NFR1.2

TITLE: Request Time

DESC: Each user request should be processed within the timeframe of 5 seconds.

5.1.3 Non - Functional requirement 1.3

ID: NFR1.3

TITLE: Upload Time

DESC: Data files should be uploaded in no less than 1 minute.

5.1.4 Non - Functional requirement 1.4

ID: NFR1.4

TITLE: Visualization Rendering Time

DESC: Rendering of graphs, pie charts and other visualizations should take no more than 10 seconds.

5.1.5 Non - Functional requirement 1.5

ID: NFR1.5

TITLE: Notification Time

DESC: The website shall be able to process and send a notification with a latency of no greater than 1 minute.

5.2 Data Requirements

5.2.1 Non - Functional requirement 2.1

ID: NFR2.1

TITLE: Date Format

DESC: The Date format, wherever relevant in the application, must be in the format of DD/MM/YYYY.

5.2.2 Non - Functional requirement 2.2

ID: NFR2.2

TITLE: Numerical Accuracy

DESC: Any numerical information available in the data visualization diagrams should be accurate up to 1 decimal place.

5.2.3 Non - Functional requirement 2.3

ID: NFR2.3

TITLE: Data Precision

DESC: The precision of calculations with derived data shall be at the same degree of precision as the originating source data.

5.3 Security Requirements

5.3.1 Non - Functional requirement 3.1

ID: NFR3.1

TITLE: Anonymisation

DESC: The author names, together with any other column data containing sensitive information, must be anonymised in the form of a string of numbers.

5.3.2 Non - Functional requirement 3.2

ID: NFR3.2

TITLE: Password Visibility

DESC: Passwords should not be viewable at the point of entry or any point of time

5.3.3 Non - Functional requirement 3.3

ID: NFR3.3

TITLE: Record Login Attempts

DESC: Each unsuccessful attempt to log into a user's account shall be recorded.

5.3.4 Non - Functional requirement 3.4

ID: NFR3.4

TITLE: Notify Login Attempts

DESC: More than 3 unsuccessful login attempts will trigger the locking of the program chair's account for 10 minutes and an email notification will be sent to the user.

5.3.5 Non - Functional requirement 3.5

ID: NFR3.5

TITLE: Password Reset

DESC: Password resets due to forgotten passwords shall only be allowed if the user verifies the request through an email that shall be sent to them.

5.4 Software Quality Attributes

5.4.1 Non - Functional requirement 4.1

ID: NFR4.1

TITLE: MTBF

DESC: The website should achieve at most 100 hours MTBF (mean time between failure).

5.4.2 Non - Functional requirement 4.2

ID: NFR4.2

TITLE: Uptime

DESC: The website must achieve an uptime of 99.5%.

5.4.3 Non - Functional requirement 4.3

ID: NFR4.3

TITLE: Availability

DESC: Unless the system is non-operational, the website must present the program chair with a notification informing him that the system is unavailable at that point of time.

5.5 Prioritization and Release Plan

NFR1. Performance Requirements	Priority	Deadline	Sprint
NFR1.1 - Supported Browsers	Low	Week 11	3
NFR1.2 - Request Time	Medium	Week 9	2
NFR1.3 - Upload Time	Medium	Week 9	2
NFR1.4 - Visualization Rendering Time	Medium	Week 9	2
NFR1.5. - Notification Time	Medium	Week 9	2

NFR2. Persistence & Management of Multiple Conference Data for Multiple Conferences	Priority	Deadline	Sprint
NFR2.1 - Date Format	High	Week 7	1
NFR2.2 - Numerical Accuracy	High	Week 7	1
NFR2.3 - Data Precision	High	Week 7	1

NFR3. Flexible Data Schemes	Priority	Deadline	Sprint
NFR3.1 - Anonymisation	High	Week 7	1
NFR3.2 - Password Visibility	Low	-	-
NFR3.3 - Record Login Attempts	Low	-	-
NFR3.4 - Notify Login Attempts	Low	-	-
NFR3.5 - Password Reset	Low	-	-

NFR4. User Interface Enhancements	Priority	Deadline	Sprint
NFR4.1 - MTBF	Low	Week 11	3
NFR4.2 - Uptime	Low	Week 11	3
NFR4.3 - Availability	Low	Week 11	3

Design Document

6. Introduction

This document is intended to provide a comprehensive overview of the design of ChairVise 4.0, an online conference management system. The purpose and complete declaration for the system's development will be outlined. This document is specifically meant to be proposed for the approval of the CS3219 teaching team and as a guide for future development teams during the implementation phase.

7. Setting up

To get started, you should have cloned the repository on your local machine. Ensure that you have installed the following software with the specified versions:

- Python 3.8.3
- Node 12.18.0
- NPM 6.14.4
- Java 8
- MySQL 5.6.

Frontend:

1. Navigate to ``src/web/app``.
2. Install dependencies using ``npm install``.
3. Run ``npm run serve``.
4. Access the application through `http://localhost:4040`

Backend:

1. In MySQL Server, create a new database.
2. Configure the connection to local database MySQL
 - a. Navigate to `src/main/resources/application-local.properties`
 - b. Input the correct information to connect to your local database:
 - i. `spring.datasource.username=[YOURDATABASENAME]`
 - ii. `spring.datasource.password=[YOURPASSWORD]`

3. Run application backend
 - a. Run `./gradlew appengineRun``
 - b. Access the application through `http://localhost:8080/_ah/admin`

Note: This project works best in Firefox

8. Development Process

ChairVise 4.0 was developed using the Agile development method, with each sprint lasting 2 weeks. This allowed for sufficient time for development before each deadline, while taking into account other external factors such as assignments and projects for other modules. Each sprint consisted of 4 processes:

8.1. Process Steps

1. Plan
 - a. Discuss which features to develop and/or bugs to fix
 - b. Decide if there are any features to continue working on.
2. Design
 - a. Make considerations towards the frontend and backend mechanisms of the features to be implemented.
 - b. Come up with a rough sketch of the required workflows of the features to be implemented.
 - c. Allocate the workload amongst team members.
3. Build
 - a. Implement the codebase needed for the feature.
 - b. Integrate the frontend and backend code together.
4. Review
 - a. Evaluate the progress made on the implemented features.
 - b. Basic manual testing of the implemented features to unearth any bugs within the system.
 - c. Discuss any improvements or bug fixes to be made.

8.2. Process Details

In total, there were 4 sprints. The schedule was planned in mind with regards to which of the features were deemed more difficult to tackle. This gave our team a better start to familiarize ourselves with the code base, by completing the simpler features first such as the personalized dashboard and conference notifications. As our knowledge of the codebase increased, we moved on to tackling more challenging features such as the management of multiple conference data and the flexibility of data schemes.

Additionally, we adopted the pair programming agile software development technique. This entails two programmers working together, which was done over video calls. One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in. The two programmers switch roles frequently.

We allocated functional requirements to each pair based on the amount of work required for each of the functional requirements, and how interdependent they were. For example, FR2.1 was able to be developed independently by one pair and required more work. On the other hand, FR2.2, FR2.3 and FR2.4 each required less work and were highly dependent on each other, which made it more fitting for one pair to tackle these features together.

8.3. Schedule

The details of the work done (the requirements satisfied as well as their respective functional requirement IDs) for each sprint are as follows:

Feature(s)	Member Assigned	Deadline	Sprint ID
Personalized dashboard (FR1.3, FR1.4, FR1.5, FR1.6)	Fatin and Jordan	Week 7	1
Conference notifications (FR1.1, FR1.2)	Hui Chun and Jia Hui		
Management of multiple conference data (FR2.1)	Fatin and Jordan	Week 9	2
Management of multiple conference data (FR2.2, FR2.3, FR2.4)	Hui Chun and Jia Hui		
Flexibility of data schemes (FR3.1, FR3.2, FR3.3)	Fatin and Jordan	Week 11	3
Flexibility of data schemes (FR3.4)	Hui Chun and Jia Hui		
General enhancements to existing features and Bug fixes	Hui Chun and Jia Hui	Week 13	4
Documentation	Fatin and Jordan		

Note: For detailed descriptions of each of the Functional Requirements, please refer to Section 3 (Software Requirements Specifications - Functional Requirements).

9. Overall Architecture

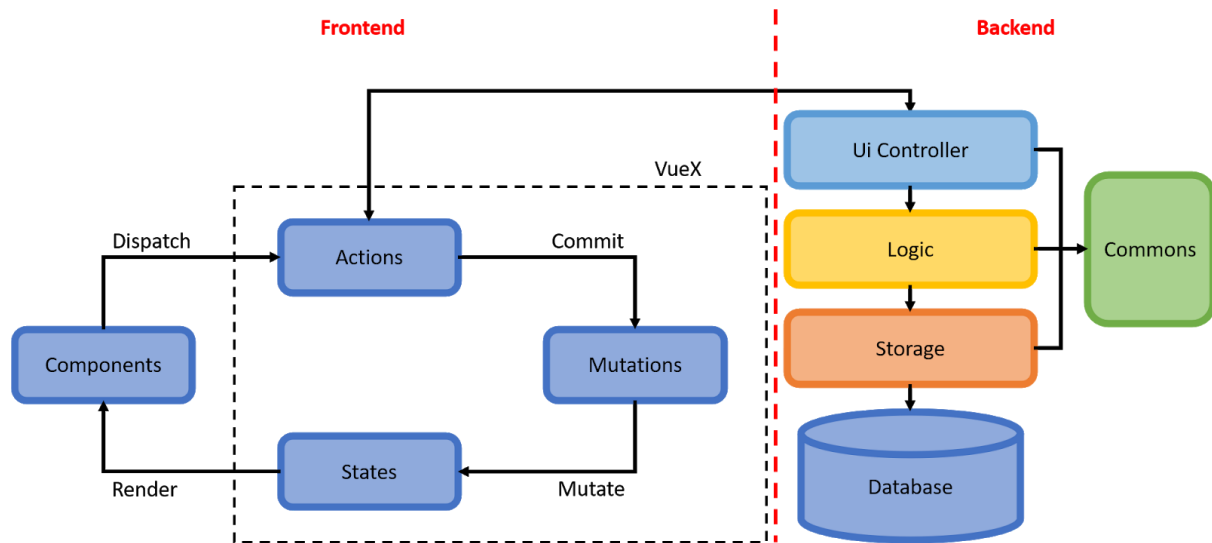


Figure 25: Overall Architecture of ChairVise

Note: Actions does not interact directly with the UI Controller due to encapsulation. However, their interactions have been simplified for ease of visualization within the diagram.

As ChairVisE 4.0 is an extension of ChairVisE 3.0, its development process will continue to use the languages and frameworks used in ChairVisE3.0, which are:

- Vue + Vuex (Javascript)
- Spring Boot and Gradle (Java)
- Google App Engine
- Google Cloud SQL
- Continuous Development using Travis

We have also utilised the following additional libraries in the implementation of our newly-added features:

- Thymeleaf for the backend (MailContentBuilder, Section 10.4.2)
- Vue-notification for the frontend (ManageData.vue, Section 11.3.5)

The backend and frontend design are further detailed in their respective sections below.

10. Backend Design

This section serves to give a description of the major components in the backend architecture of ChairVise 4.0. Subsequent sections provide more information on the inner workings of individual components.

10.1. Architecture

The backend of ChairVise has been implemented using the Java Spring framework and adopts a layered architecture made up of 4 layers

The following diagram visualizes the relationship between each of the layers:

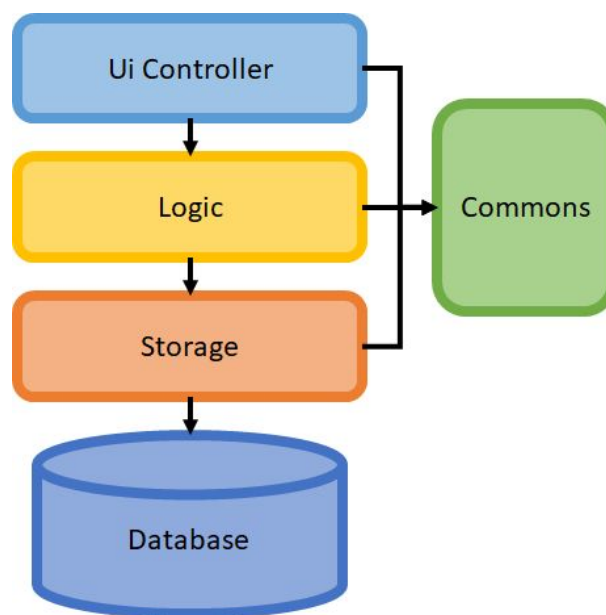


Figure 26: Backend Architecture of ChairVise

Note: For easier visualization of the interactions between each layer, diagrams under Section 5 - Backend Design will follow the same colour scheme as the above. Namely,

1. Commons - Green (Further details are available in Section 5.2)
2. Storage - Orange (Further details are available in Section 5.3)
3. Logic - Yellow (Further details are available in Section 5.4)
4. UI Controller - Blue (Further details are available in Section 5.5)

The application database is encapsulated using Google App Engine. It is stored using MySQL locally and Google Cloud SQL in production. Further details for each of the remaining layers are provided in their respective sections below.

10.2. Common component

Contains utility code used across the application.

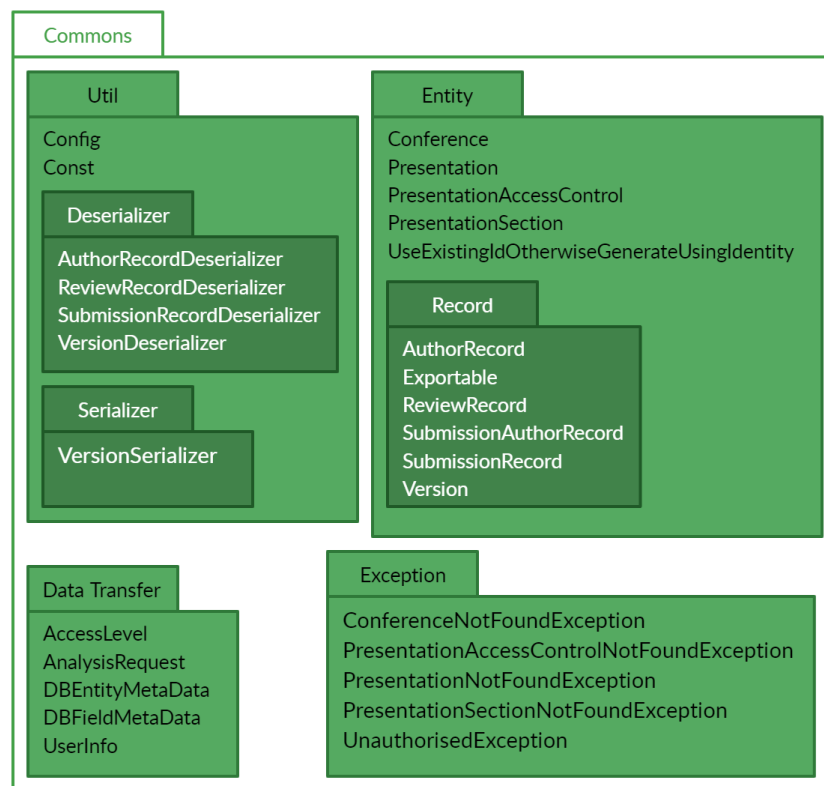


Figure 27: Package Diagram for Commons

10.2.1. Data Transfer

Contains classes that do not require persistent storage and are created and consumed in the Logic and Controller components.

- AccessLevel - Make sure users only have access to their own presentations (i.e. users cannot access presentations created by other users to read and write)
- AnalysisRequest - Used to obtain data from conference files and input into presentations (dataset and version id). May contain multiple PresentationSection objects.
- DBEntityMetaData - Represents metadata for conference files (name and description). Can contain multiple DBFieldMetaData objects

- DBFieldMetaData - Represents input fields of conference files (type, name and description)
- UserInfo - Represents user data (email and nickname)

10.2.2. Entity

Contains classes that require persistent storage. Database table schemas are defined in these classes.

- Record - data entity for conference record files (id, version, submissionid)
 - Author Record - data entity for author records
 - Submission Record - data entity for submission records
 - Submission-Author Record - data entity generated by matching each particular author in author record to their corresponding submissions in submission record
 - Review Record - data entity for review records
 - Exportable - describes what can be exposed to the user to import data
 - Version - data entity for the unique version number of each type of record
- Conference - data entity used for conference events (date, name, description, creatorID)
- Presentation - data entity used for presentation data (name, version, description, creatorID)
- PresentationAccessControl - data entity used to make sure users only have access to their own presentations. Associated with a single Presentation.
- PresentationSection - data entity used for storing each analysis of record files to be input into presentations. Associated with a single Presentation.
- UseExistingIdOtherwiseGenerateUsingIdentity - data entity to use the existing id of a record. If the current id doesn't exist (id is null), a new one is generated

The following entity-relationship diagrams shows the relationship between the different entities:

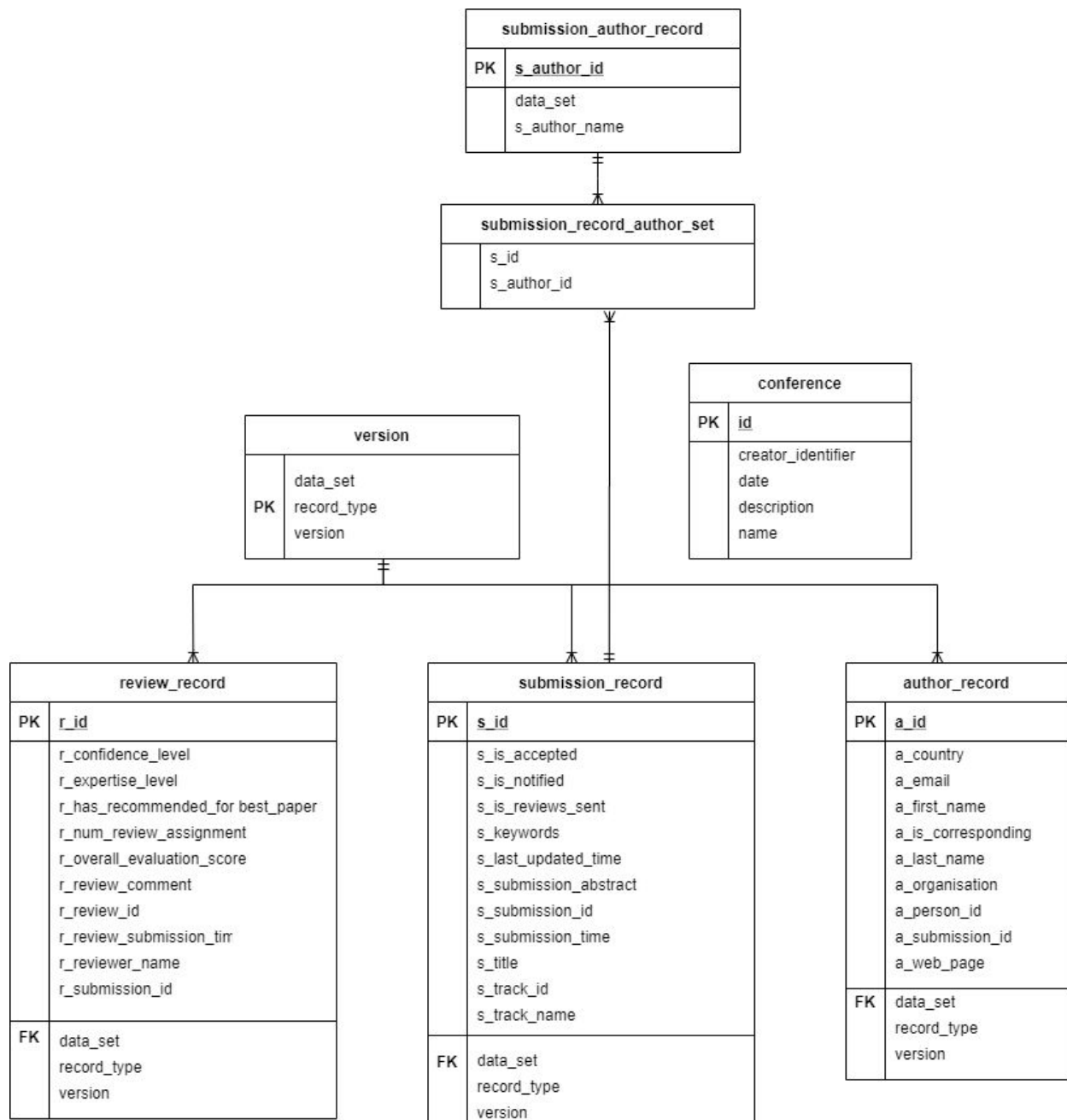


Figure 28: Entity - Relationship Diagram for Record related data

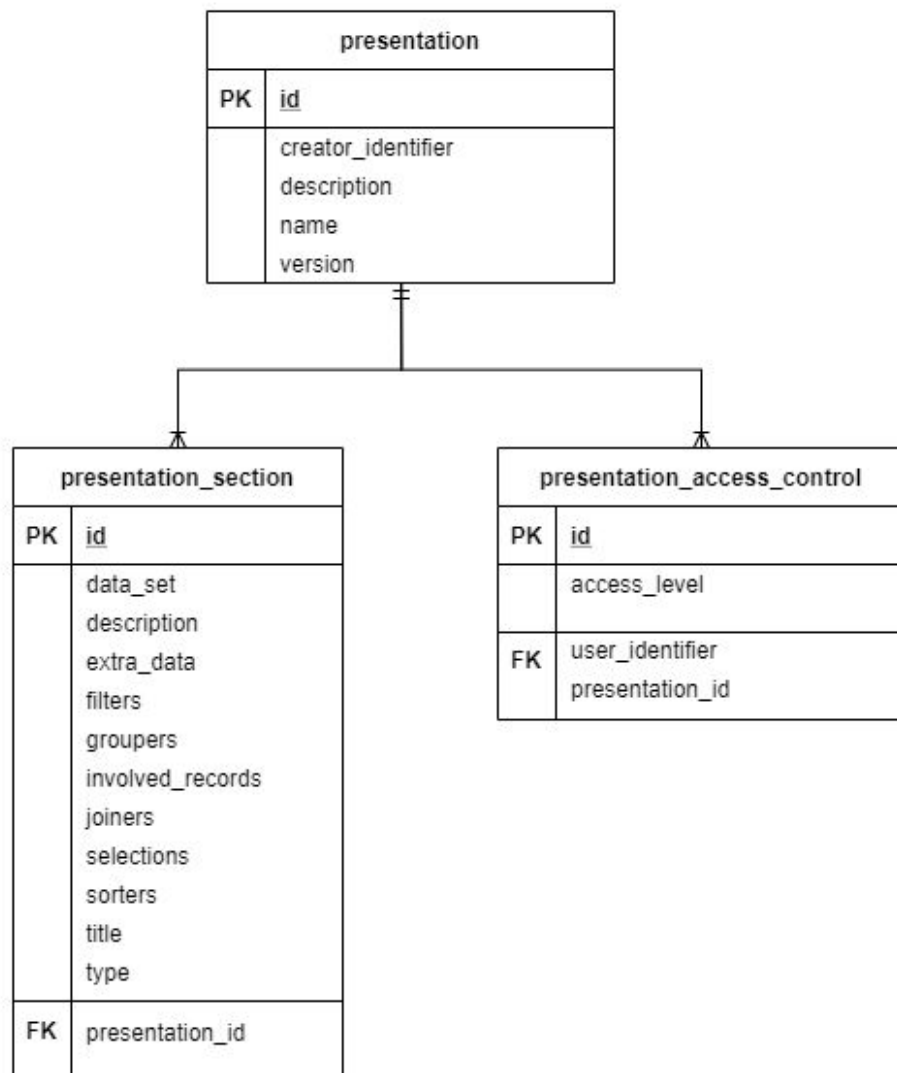


Figure 29: Entity - Relationship Diagram for Presentation related data

The following diagram illustrates the relationship between the entities and data transfer objects related to Presentations.

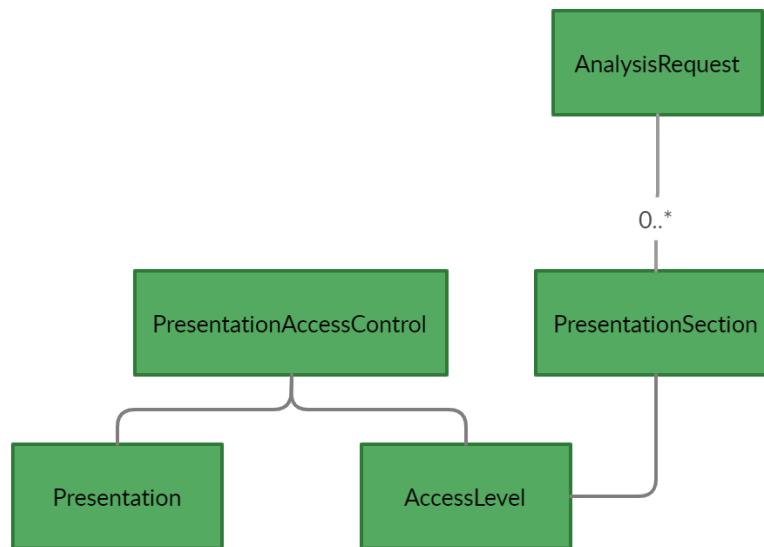


Figure 30: Class Diagram for Presentation related data

10.2.3. Exception

Contains classes that provides custom exception messages

- Used when an entity is not found (ConferenceNotFoundException, PresentationNotFoundException, PresentationSectionNotFoundException and PresentationAccessControlNotFoundException)
- Used when there is an attempt to access data that is unauthorised such as data from other users etc. (UnauthorisedException)

10.2.4. Util

- Serializer - formats data that is returned by backend (VersionSerializer)
- Deserializer - constructs entity classes (AuthorRecordDeserializer, SubmissionRecordDeserializer, ReviewRecordDeserializer and VersionDeserializer) from request bodies from frontend.

10.3. Storage component

Defines repository interfaces for appropriate transactions that extend the central JpaRepository, which is where the database connection happens.

The JpaRepository is a marker interface that allows the Spring Data Repository engine to recognize it and apply the necessary proxy classes to implement basic CRUD actions as well as some custom methods. Each method defined in the repository is interpreted and mapped to a unique SQL query on their respective database table. It utilises the persistence framework provided by Google App Engine, using MySQL.

10.3.1. General Structure

The typical relationship between each storage component and the entities that they store is as shown in the diagram below (where “**ABC**” refers to a particular entity):

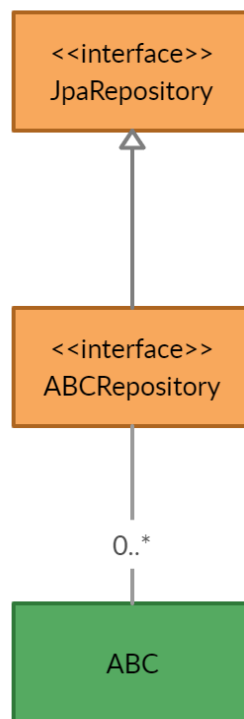


Figure 31: Class diagram for a typical Storage component

*Note: This structure is used by **all** Storage components.*

*All Storage components use a **Long** object as an **identifier** with the exception of **VersionRepository**, which uses **VersionPK***

10.4. Logic component

The main logic of the application which has been implemented in Java using the Spring framework. This is invoked by Controller components in the form of algorithms, database operations (transactions), etc. In transactions, the methods have access to the database through implementing repository code (which are basically interfaces with access to the actual database).

10.4.1. General Structure

The typical relationship between each logic component and their respective storage component is as shown below (where “**ABC**” refers to a particular entity):

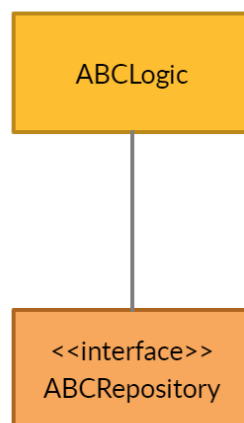


Figure 32: Class diagram for a typical Logic component

*Note: This structure is used by **ConferenceLogic**, **PresentationAccessControlLogic**, **PresentationLogic**, **PresentationSectionLogic** and **VersionLogic**.*

On the other hand, the **RecordLogic** component is associated with multiple repositories as shown below

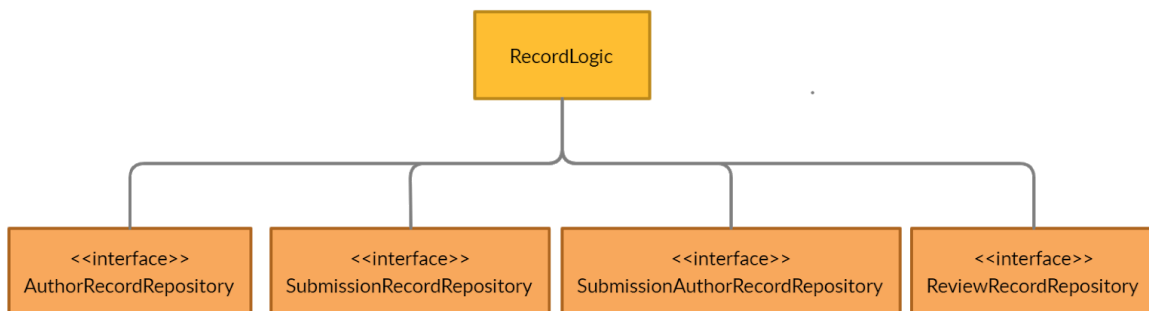


Figure 33: Class diagram for a **RecordLogic** component

Other exceptions to the general structure:

- AnalysisLogic - this component is associated with a JdbcTemplate.
- DBMetaDataLogic - since DBEntityMetaData does not require persistent storage, this component directly holds its respective DataTransfer object without requiring an intermediate repository
- MailContentBuilder - this component is associated with a TemplateEngine.
- MailSenderHelper - this component is associated with a JavaMailSender.
- GateKeeper - this component is associated with the PresentationAccessControlRepository.

10.4.2. Details

AnalysisLogic	Generates a map from field name to type in the form of an SQL query string from AnalysisRequest. It also performs and returns the result of the SQL query
ConferenceLogic	Performs CRUD actions on Conferences using ConferenceRepository
MailContentBuilder	Composes the conference notification to be sent to the user's email
MailSenderHelper	Sends the conference notification directly to the user's email.
GateKeeper	Verify user login status and access rights of user to Presentation

PresentationLogic	Perform CRUD actions on Presentations using PresentationRepository
PresentationAccessControlLogic	Performs CRUD actions on PresentationAccess using PresentationAccessControl
PresentationSectionLogic	Associate new PresentationSection with an existing Presentation and performs CRUD actions on PresentationSection using PresentationSectionRepository
RecordLogic	Overwrite/delete old entries and create new entries in the database based on the version specified in the record. Also creates many-to-many mappings between SubmissionRecord and SubmissionAuthorRecord
VersionLogic	Calls VersionRepository to persist version entries it receives from VersionController

10.5. UI component

Handles API calls made by the frontend and serves static production Vue Files.

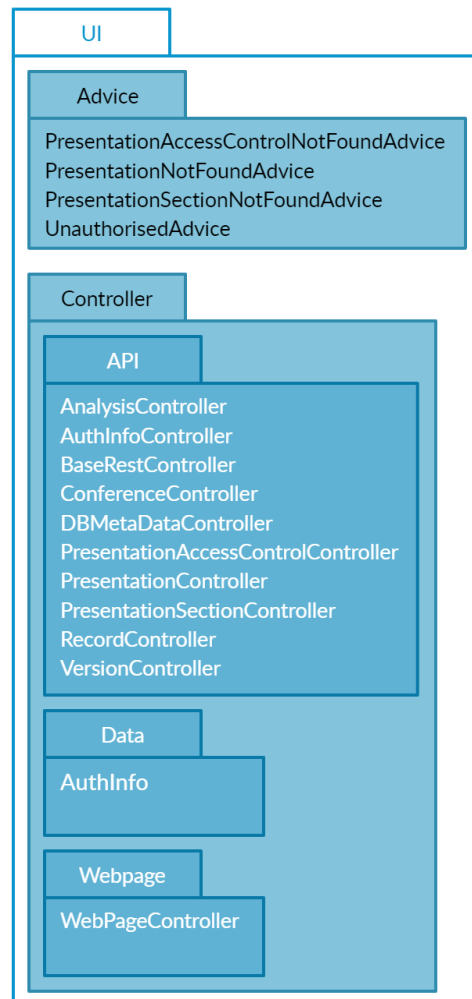


Figure 34: Package Diagram for UI

10.5.1. Advice

Controller advice classes are defined and used as centralized `ExceptionHandler`s (as opposed to having each controller define their own exception handlers).

10.5.2. Controller

API - provides backend REST API access by connecting the REST requests (incoming URLs) with the appropriate class and method through mapping of the requests to controller functions.

Data - contains `AuthInfo` which is a helper object to be sent to the client in JSON, similar usage to data transfer in common (no persistent storage). It is used to verify if the user is logged in and to obtain user info. It also defines the `loginUrl` and `logoutUrl`.

Webpage - handles static file requests by redirecting certain URLs to the frontend server

10.5.3. General Structure

All of the controllers are extended from the `BaseRestController`. There are 2 general structures for the Controller components.

The first typical relationship between each of the Controller components and the Logic components that they are associated with is as shown in the diagram below (where “**ABC**” refers to a particular entity)

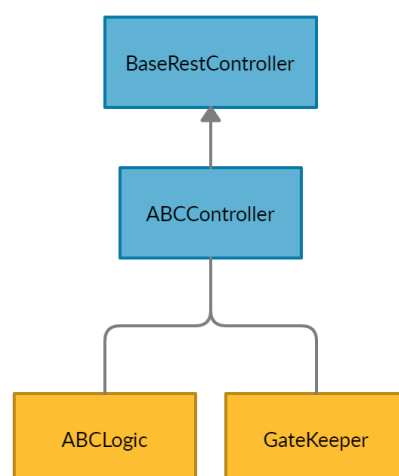


Figure 35: Class diagram for a typical UI Controller component

*Note: This structure is used by **ConferenceController**, **PresentationController**, **RecordController**, **VersionController***

The second typical relationship between each of the Controller components and the Logic components that they are associated with is as shown in the diagram below (where “**ABC**” refers to a particular entity)

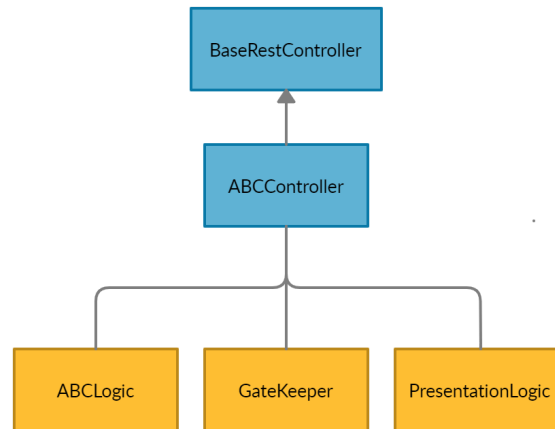


Figure 36: Class diagram for a typical UI Controller component

*Note: This structure is used by **PresentationAccessControlController**, **PresentationSectionController**, **AnalysisController***

Exceptions:

- DBMetaDataController is the only controller that does not need to utilise GateKeeper.
- AuthInfoController is only associated with GateKeeper and has no other Logic components.

10.5.4. Details

AnalysisController	Handles analysis requests sent by the frontend and issues SQL aggregation query to the backend.
AuthInfoController	Checks the current authentication status of the user. For example, it will return a login URL if the user is not logged in and a logout URL if the user is logged in.
ConferenceController	Responsible for the CRUD actions on conference related information.
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	Responsible for the CRUD actions on Presentation related information.
PresentationAccessControlController	Responsible for the CRUD actions on PresentationAccess control related information.
PresentationSectionController	Responsible for the CRUD actions on PresentationSection related information.
RecordController	Accepts the CSV data imported by users and stores them in the database as well as delete unwanted data
VersionController	Persists version entries in the backend and allows the frontend to obtain version entries.
WebPageController	Serves the static production files built by Vue.

10.6 Implementation of features

10.6.1. Personalized Dashboard (FR1.2, FR1.3, FR1.4, FR1.5, FR1.6)

The following sequence diagram shows the interactions between the Logic, UI and Storage components that are required to display the information on the dashboard within the homepage of the application. “ABC” here refers to the type of record (Author, Submission or Review)

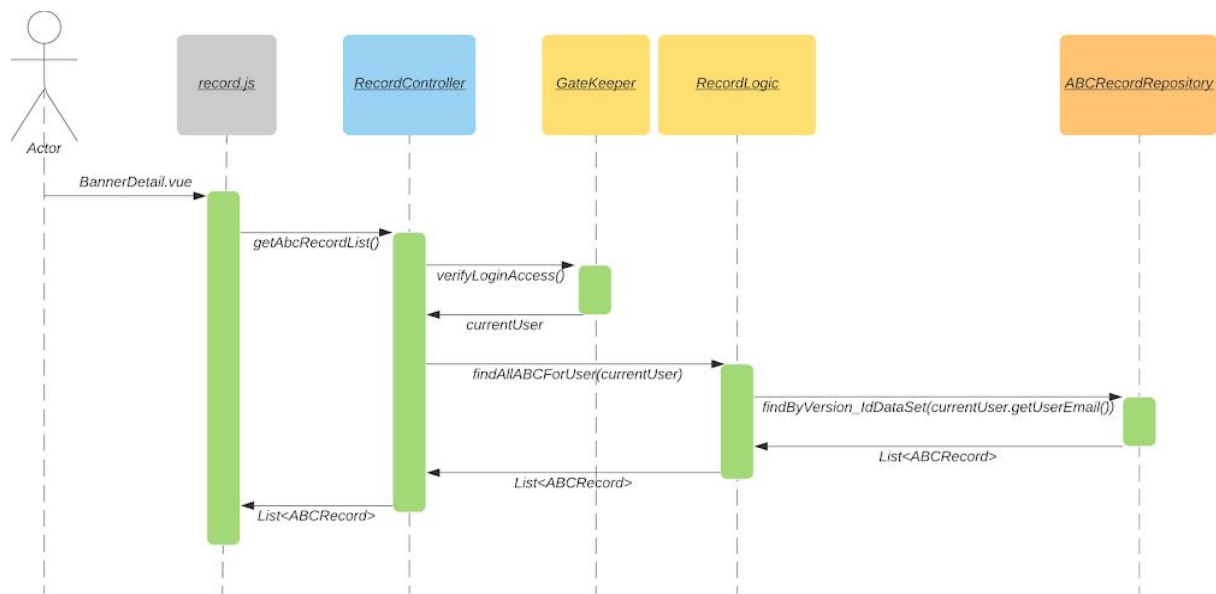


Figure 37: Sequence diagram for when the user is viewing the personalized dashboard

Once List<ABCRecord> is returned, the size of the list is used to show how many records of that type ChairVise currently has.

The sequence diagram required to retrieve the upcoming conference to display on the dashboard is similar. The sequence goes through conference.js first followed by ConferenceController, ConferenceLogic and ConferenceRepository.

10.6.2. Conference Notifications (FR1.1)

The following sequence diagram shows the interactions between the Logic, UI and Storage components that are required to send an email notification to the user.

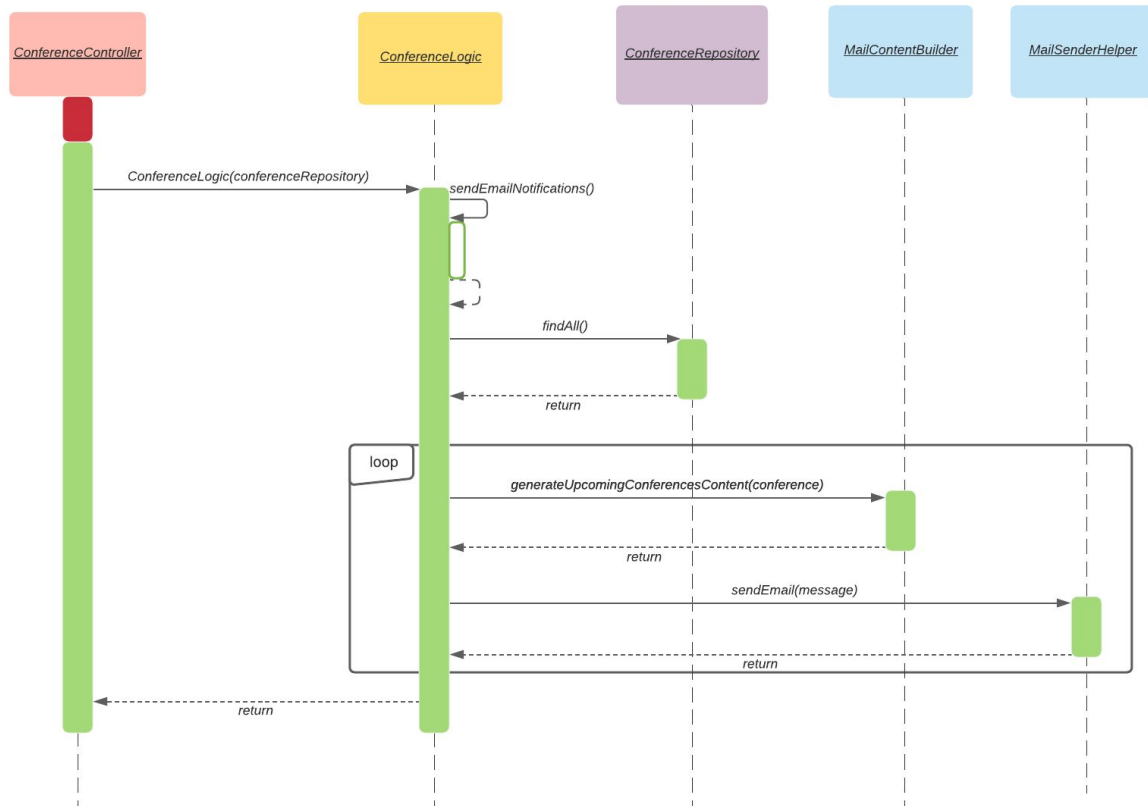


Figure 38: Sequence diagram for sending a conference notification to the user's email

First since the email notification is a scheduled task, ConferenceLogic self-invokes the scheduled method call `sendEmailNotifications()`, after which it requests for all the conferences for the particular user from the repository, and filters those that are within the next 3 days. For the conferences that fall within this category, the ConferenceLogic component calls the MailContentBuilder to `generateUpcomingConferencesContent(...)` and `sendEmail(...)`.

10.6.3. Management of multiple conference data (FR2.1, FR2.2, FR2.3, FR2.4)

The following sequence diagram shows the interactions between the Logic, UI and Storage components that are required to delete conference data within the 'Manage Data' page of the application. "ABC" here refers to the type of record (Author, Submission or Review)

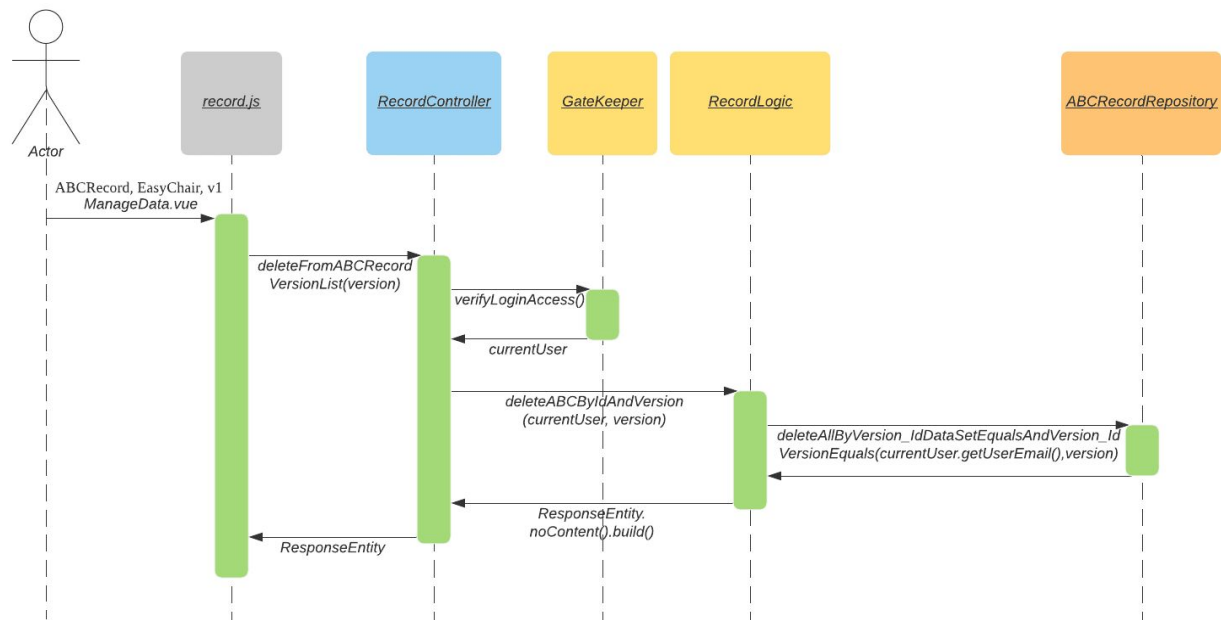


Figure 39: Sequence diagram for when the user is using the Manage Data page

10.6.4. Flexibility of data schemes (FR3.1, FR3.2, FR3.3, FR3.4)

The following sequence diagram shows the interactions between the Logic, UI and Storage components that are required to map the conference data within the Import Data page of the application.

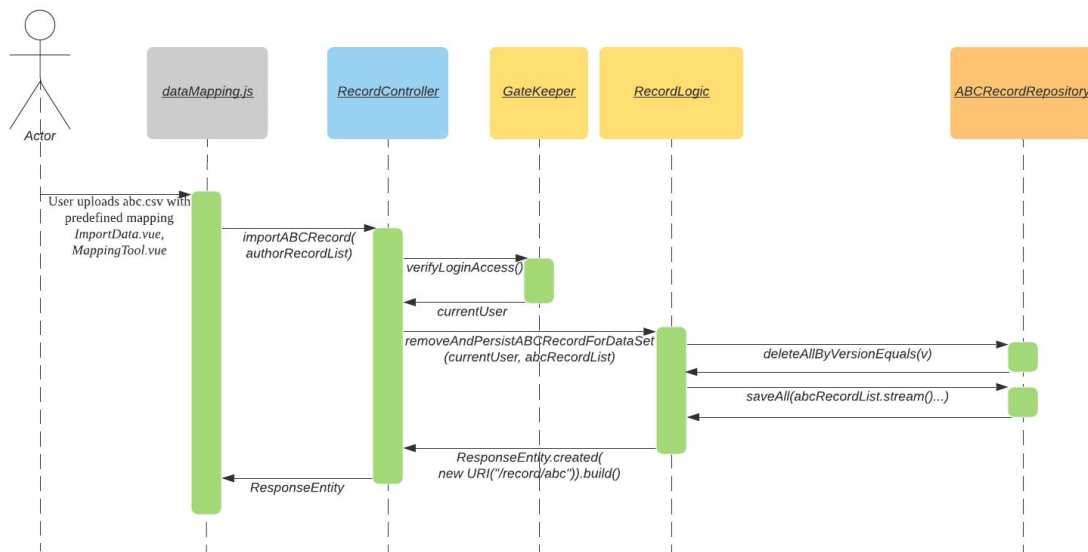


Figure 40: Sequence diagram for when the user is using the Manage Data page

11. Frontend Design

This section serves to give a description of the major components in the frontend architecture of ChairVise 4.0. Subsequent sections provide more information on the inner workings of individual components.

11.1. Architecture

The frontend of ChairVise has been implemented using the Vue.JS framework and Vuex is an official plugin for Vue.js which offers a centralised datastore for use within your application.

VueX adopts an adapted FLUX architecture which features a unidirectional data flow leading to simpler application design and reasoning. The following diagram visualizes the relationship between the constituents:

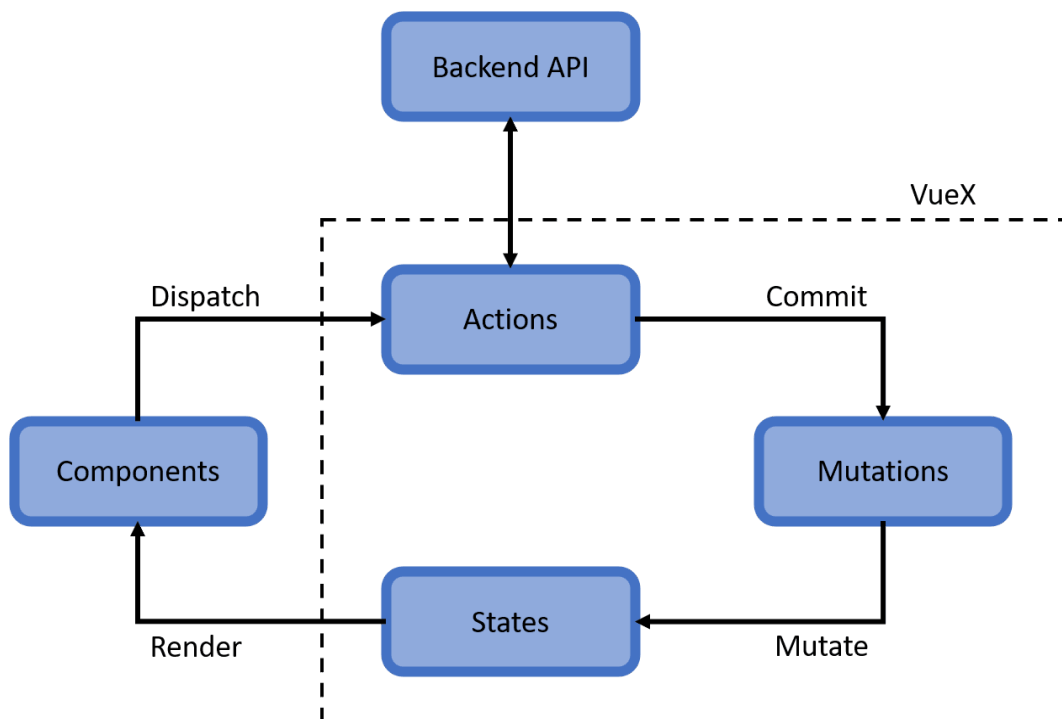


Figure 41: Backend Architecture of ChairVise

11.1.1. Components

Reusable Vue instances that act as custom elements. For organisation purposes, components can also be nested. This allows for the Separation of concerns because each component is distinct from one another, fulfilling different responsibilities.

11.1.2. Actions

Executes an arbitrary amount of asynchronous operations but does not update the state directly.

11.1.3. Mutations

Updates the states directly. Since mutations are the only constituents that can do this, it makes state management predictable. Each mutation has the state as the first parameter and an optional payload as second.

11.1.4. States

This is the data stored in your application that is received by components. When the state changes, the component will update itself.

It uses a single source tree with each single object containing all application level state. It also serves as the "single source of truth" by having a centralised store in an application. This makes it straightforward to locate a specific piece of state, and allows us to easily take snapshots of the current app state for debugging purposes.

ChairVise has divided the store into smaller modules under *src\web\app\src\store\modules*. Each of these modules contain their own states, mutations, actions and nested modules.

11.1.5. Packages

The frontend packages are as described below:

- Components - Contains reusable UI and displays logic components. These are called by multiple pages in the views. They are organized by their view hierarchy. Further details are available in Section 6.3.
- Views - Displays web pages of the application that act as the root for many components. Each of the views act as a single page application (SPA). Further details are available in Section 6.2.
- Store - Core logic of the application that acts as a single source of truth across all components (refer to States). Used to manage Json data and make transactional API calls, enabling the application to keep up to date with its domain databases. Further details are available in Section 6.3.

Further details for each of the packages are provided in their respective sections below.

11.2. Store

11.2.1. Data

predefinedMappings.js - Mappings used when importing data

predefinedQueries.js - Queries used when generating presentations

11.2.2. Helpers

pdfDownloader.js - Used to download a .pdf file of a presentation

pptxDownloader.js - Used to download a .pptx file of a presentation

processor.js - Process the mapping of data to their respective fields

11.2.3. Modules

accessControl.js	Handles logic related to the access control of a presentation
conference.js	Handles logic related to the conference
dataMapping.js	Handles logic related to the mapping of CSV and built-in data schema

presentation.js	Handles logic related to the presentation
section.js	Handles logic related to the presentation section
userInfo.js	Handles logic related to the authentication

11.3. Views and Components

11.3.1. Analyze.vue

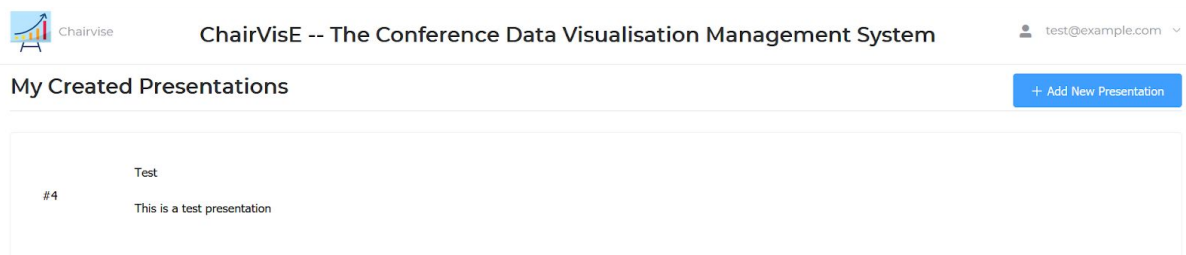


Figure 42: Screenshot of *Analyze.vue* as shown on the '/analyze' path

emptystates/EmptyPresentation.vue

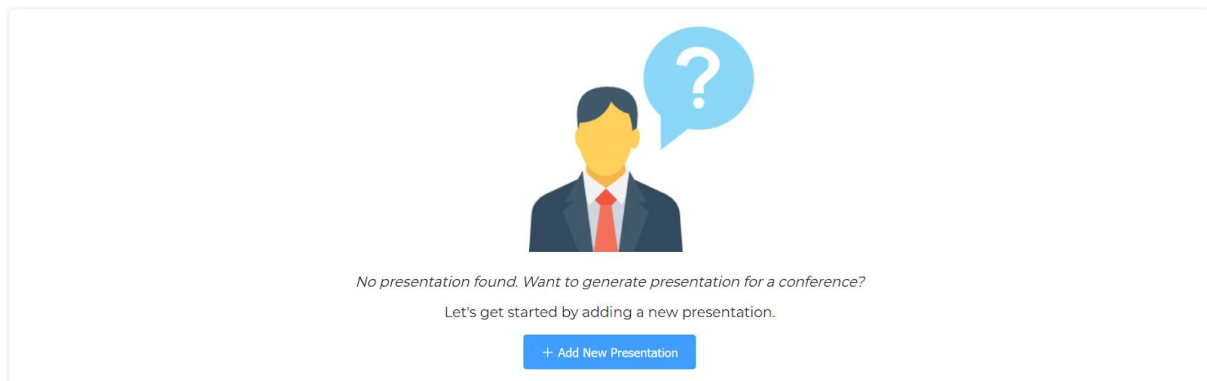


Figure 43: Screenshot of *EmptyPresentation.vue* as shown on the '/analyze' path

11.3.2. ConferenceSection.vue

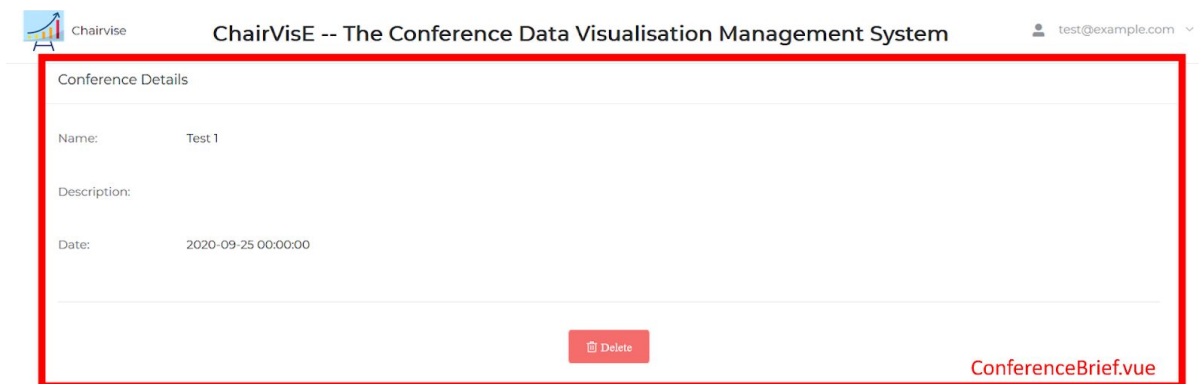


Figure 44: Screenshot of ConferenceSection.vue as shown on the '/conference/:id' path

11.3.3. Home.vue (FR1.3, FR1.4, FR1.5, FR1.6)



Figure 45: Screenshot of Home.vue as shown on the '/home' path

11.3.4. ImportData.vue (FR3.1, FR3.2, FR3.3, FR3.4)

The screenshot shows the 'Upload Data' form in the ChairVisE system. The form is titled 'Upload Data' and contains three main sections: 'Record Information', 'Mapping Information', and 'Version Information'. In the 'Record Information' section, there are two groups of tabs. The first group, 'Conference Type', has three tabs: 'EasyChair' (selected), 'SoftConf', and 'Others'. The second group, 'Table Type', has three tabs: 'Author Record' (selected), 'Review Record', and 'Submission Record'. The 'Mapping Information' section contains a note about expected CSV column headers: "submission #", "first name", "last name", "email", "country", "organization", "Web page", "person #", "corresponding?". The 'Version Information' section has a 'Version' label and an 'Input Version' text field.


Figure 46: Screenshot of ImportData.vue as shown on the '/importData' path

MappingTool.vue

The screenshot shows the 'MappingTool.vue' form in the ChairVisE system. The form is titled 'Mapping' and contains two main sections: 'Database fields' and 'Imported data fields'. The 'Database fields' section lists nine fields with their corresponding database names: 'Submission Id' (submission #), 'First Name' (first name), 'Last Name' (last name), 'Email' (email), 'Country' (country), 'Organisation' (organization), 'Web Page' (Web page), 'Person Id' (person #), and 'Is Corresponding' (corresponding?). Each field has a 'String' label and a delete button (X). The 'Imported data fields' section is currently empty. At the bottom of the form, there are two buttons: 'Upload' (green) and 'Back' (grey).

Figure 47: Screenshot of MappingTool.vue as shown on the '/importData' path

11.3.5. ManageData.vue (FR2.1, FR2.2, FR2.3, FR2.4)


ChairVise

ChairVisE -- The Conference Data Visualisation Management System

test@example.com

Select the criteria below to specify dataset to display

AuthorRecord

1

EasyChair

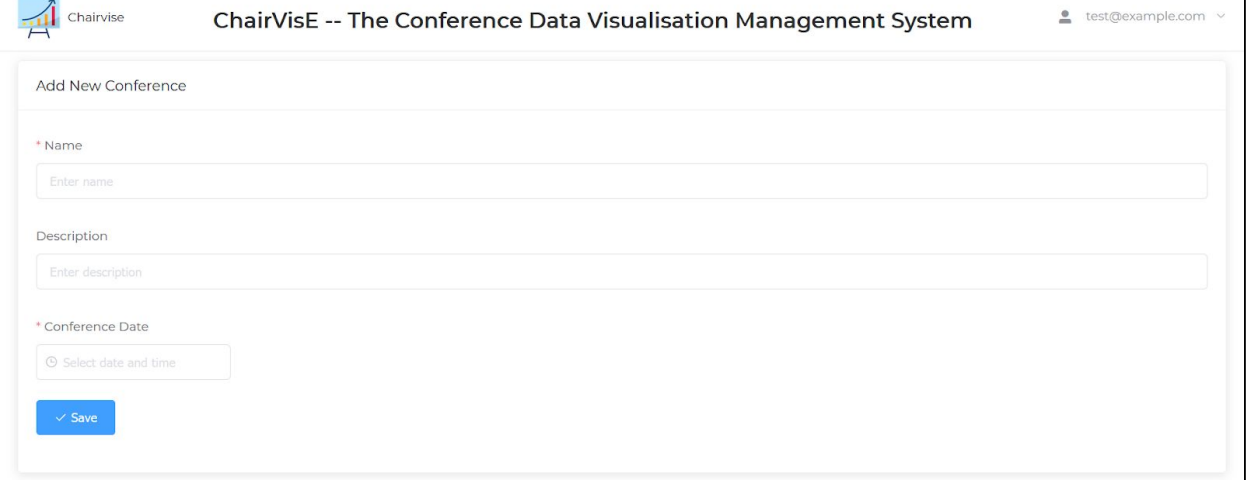
Retrieve Data

Delete Data

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

Figure 48: Screenshot of ManageData.vue as shown on the '/manageData' path

11.3.6. NewConference.vue (FR1.1, FR1.2)



The screenshot shows a web application titled "ChairVisE -- The Conference Data Visualisation Management System". The user is logged in as "test@example.com". The page is titled "Add New Conference". It contains three input fields: "Name" (with a red asterisk), "Description", and "Conference Date" (with a red asterisk). The "Name" field has a placeholder "Enter name". The "Description" field has a placeholder "Enter description". The "Conference Date" field has a placeholder "Select date and time". There is a blue "Save" button with a checkmark icon.

Figure 49: Screenshot of NewConference.vue as shown on the '/conference/add' path

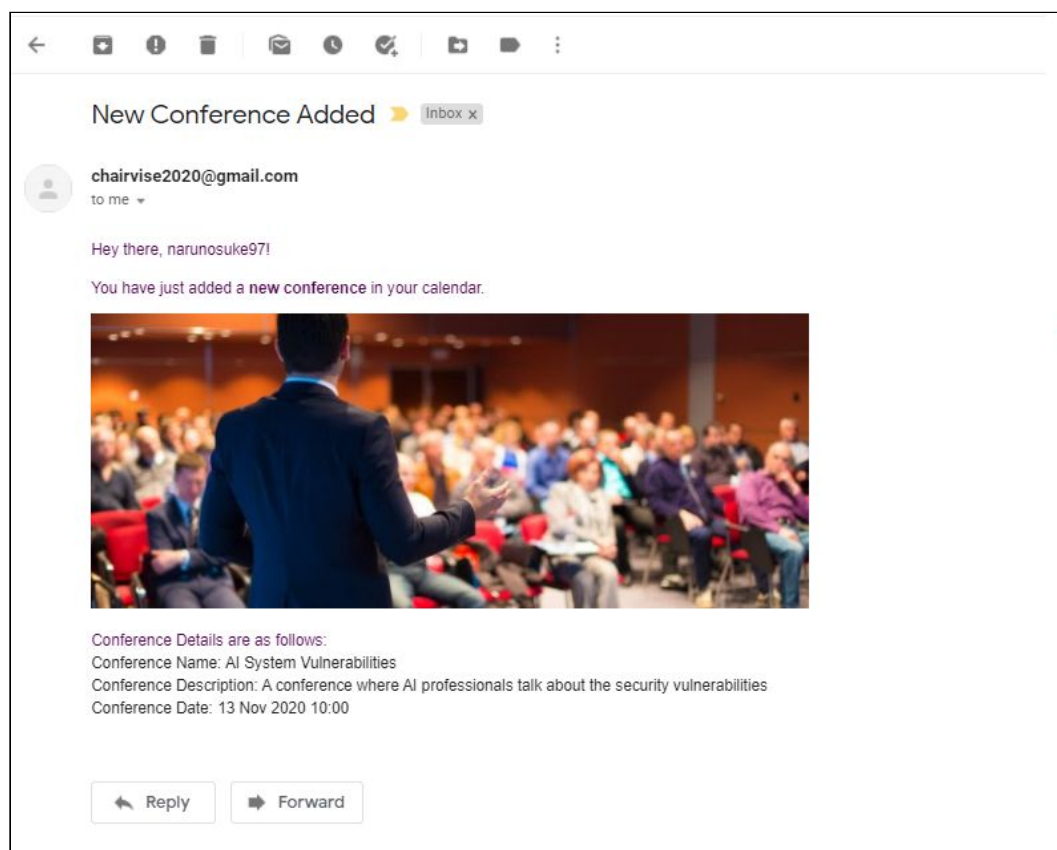


Figure 50: Screenshot of email notification when new conference is added

11.3.7. NewPresentation.vue

ChairvisE

ChairVisE -- The Conference Data Visualisation Management System

test@example.com

Create New Presentation

Name

Enter name

Description

Enter description

Save

Figure 51: Screenshot of NewPresentation.vue as shown on the '/conference/add' path

11.3.8. PresentationSection.vue

ChairvisE

ChairVisE -- The Conference Data Visualisation Management System

test@example.com

Presentation Details

PresentationBrief.vue

Name: Test

Access Control: Created by test@example.com

Description: This is a test presentation

PDF

Powerpoint

Share

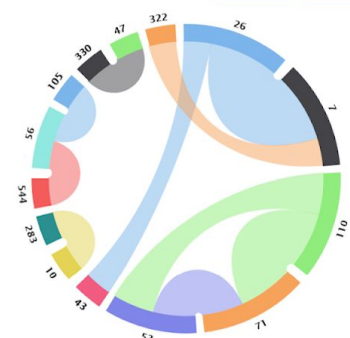
Edit

Delete

Inter-author collaboration

Edit

Delete



Highcharts.com

AbstractSectionDetail.vue

Select version

1

Add section

Please select a section to add

+ Add New Section

SectionListPanel.vue

Figure 52: Screenshot of PresentationSection.vue as shown on the '/conference/add' path

AccessControlPanel.vue

Share with other users: ×

Shareable Link

Any one with the link

http://localhost:4040/analyze/5

Cannot Access ▼

Specific Access Control

Email	Access Level
No Access Control for this Presentation!	

Add New Access Control

* Email address

Email of the user to share

* Permissions

Permission the user will have ▼

Add

Figure 53: Screenshot of AccessControlPanel.vue as shown on the '/conference/add' path

emptystates/EmptySection.vue

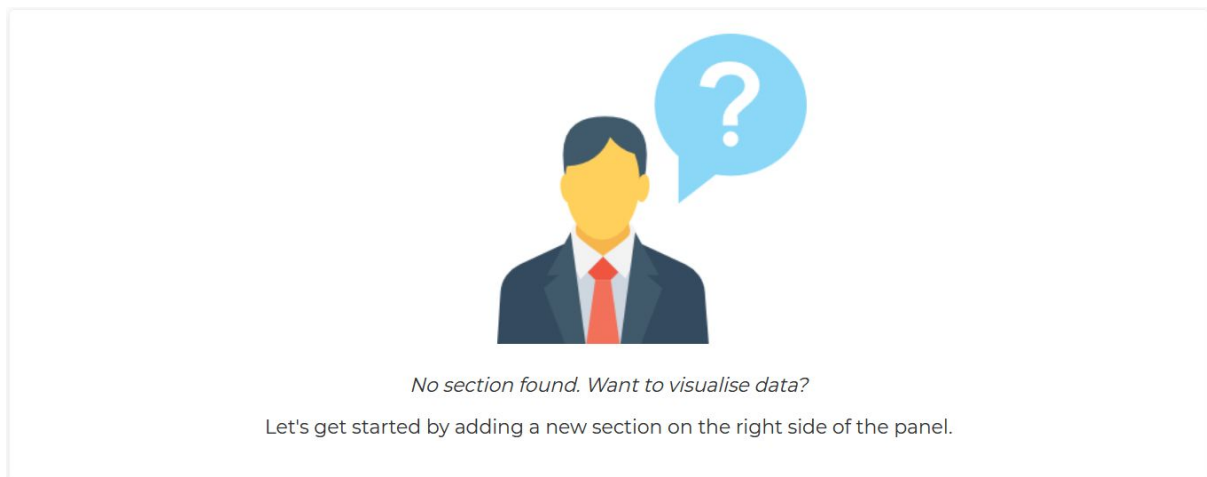


Figure 54: Screenshot of EmptySection.vue as shown on the '/conference/add' path

11.3.9. UserGuide.vue

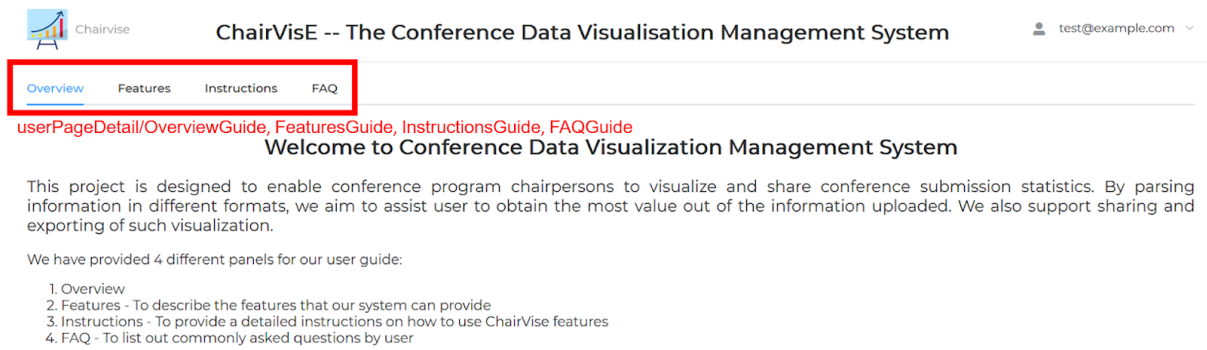


Figure 55: Screenshot of UserGuide.vue as shown on the '/conference/add' path

11.3.10. ViewConference.vue

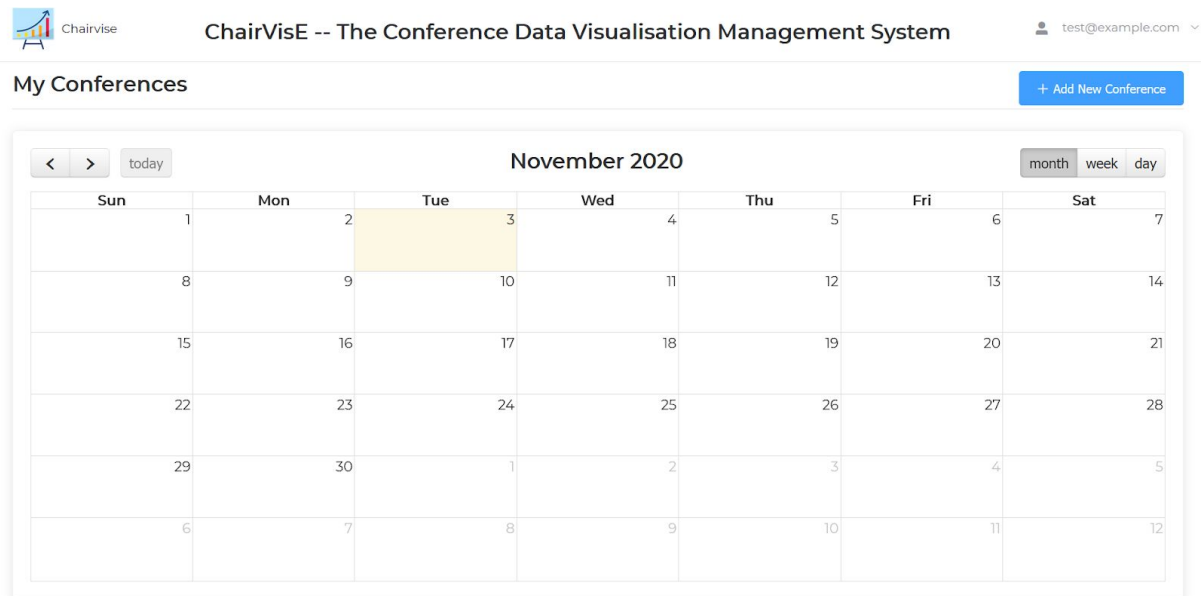


Figure 56: Screenshot of ViewConference.vue as shown on the '/conference/add' path

12. Additional Suggestions

12.1. ChairHub

12.1.1. Networking

ChairHub would introduce social networking features to ChairVise, in order to facilitate the sharing of conference statistics to other users of the system. It can be thought of as a social blog space for conference chairs to share the presentations they prepared over their own conference data.

In particular, ChairHub should allow users to add other users as “friends” within their network or remove current “friends” from their network. The purpose of this is so to give users the option to make their conferences either public, friends only or private.

In order for the above to be achieved, ChairHub should maintain user profiles that contain information about the user. Compulsory fields for ChairHub users should be

- Their name
- Their occupation (Assistant Prof, Prof, Dean etc.)
- Which educational institute they are affiliated with

Similarly, ChairHub users should have the option to make the details of their profile pages either public, friends only or private.

12.1.2. User Authentication

Future developers may wish to consider including authenticating features for ChairVise. For example, new users of ChairVise would have to sign up for ChairVise with an email affiliated to their university. An email should then be sent to the users in order for them to activate their account.

Additionally, the domain of their email address should be validated to ensure that the users are from legitimate educational institutions. A possibility would be to check against this database of email domains: <https://github.com/Hipo/university-domains-list>.

12.1.3. Conference Events

An additional feature that can be considered would be the ability for users to indicate whether or not they will be attending a conference. This would require modifications to the conference entity. An example process could be as follows:

1. The conference organiser creates a new conference event
2. The conference organiser then publicizes the event

3. Other conference chairs indicate whether or not they will be attending the conference
4. An email is sent to the conference organiser to inform him of a pending attendee verification request
5. The conference organiser verifies the attendees
6. Conference attendees now have the newly created conference event within their calendar on ChairVise
7. The conference event is added as a new event within the attendees' Google Calendar

It would also be useful to add an additional privacy layer to presentations by making them visible to all conference attendees for a particular conference. This brings the total number of options for presentation sharing to 4: Public, friends only, attendees of a particular conference only or private.

12.2. User Interface Improvements

12.2.1. Google Calendar Integration

This applies to conference events that are created by the user, as well as conference events the user has indicated that they will take part in (as explained in section 7.1.3).

In addition to the existing notification feature, a Google Calendar event would be created upon the creation of the new conference event or when the user joins a conference for the first time.

This can be done by adding a .ics (Internet Calendaring and Scheduling Core Object Specification) file attachment to the notification sent to the user.

12.2.2. Colour Coding Schemes

Recently, having a dark-mode version of websites has become common to accommodate for users with sensitive eyes (and therefore prefer looking at things that are less bright) and as a way to conserve the battery life of their devices.

On top of having a dark-mode for the website, it would be useful to allow users to customise the colours of their presentation charts by providing them with the option to select from various colour schemes for their presentations, or to create their own colour schemes and save them for future use in different presentations.

12.2.3. Visualisations

Under the presentation section, ChairVise currently lists all possible visualisations for users to choose from. Not only is this list lengthy and cumbersome, it does not rule out

visualisations that cannot be made due to the lack of data uploaded. It would be more user-friendly to perhaps list the visualisations that are creatable at the top of the list, and the remaining at the bottom of the list with details to let users know what files they should upload in order to be able to create these presentations.

In addition, future developers can make visualizations more meaningful and valuable for users by allowing users to annotate within the chart itself and improving the co-authorship mechanism.

12.2.4. Step by step guide

To improve the usability of ChairVise, it would be useful to have an optional tutorial that guides the user through the basic features of the site for new users who are logging in for the first time.

A possible way to do this is by using intro.js (<https://introjs.com>) to create a walkthrough of the features that ChairVise has to offer. This saves new users time needed to figure out how to navigate the site by themselves.

12.3. Application Improvements

12.3.1. Testing

ChairVise currently has no test cases to test the robustness of the frontend and backend components. Future developers may consider creating a set of test cases that will be able to test the source code with the requirements. Examples of the type of testing that future developers could implement include Unit Testing, Integration Testing and GUI Testing.

Useful resources regarding testing are available through the following links:

- <https://spring.io/guides/gs/testing-web/> (backend)
- <https://vue-test-utils.vuejs.org/> (frontend)
- <https://vuejs.org/v2/guide/testing.html> (frontend)

12.3.2. Use another front-end design toolkit to support responsive design.

Currently, ChairVise's frontend is not responsive. Future developers could replace the existing framework with another framework that supports responsive design. Besides ensuring that the website is responsive, the visualisations in the presentation section have to be responsive as well.

Some examples of responsive frontend frameworks are Bootstrap Vue (<https://bootstrap-vue.org/>) and Vue Material (<https://vuematerial.io/>).