CS3219 Software Engineering Principles and Patterns

ChairVise Project

Team 11

Code Repository URL:
https://github.com/CS3219-SEM1/chairvise-project-2018-team-11

Student Name	Amrut Prabhu	Muhammed Nur Kamal Bin Mohammed Ariff	Lim Zheng Kai	Li Wanghuan
Matriculation Number	A0162655B	A0154925Y	A0156045J	A0147994J

Introduction

This report is about ChairVise, a web app to enable conference program chairs to visualise and share insights from the conference submissions data. We were given an initial implementation of this application. 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.

Requirements

The following table describes the Software Requirements Specification (SRS) of the ChairVize application. It gives the following information about each requirement:

- NFR/FR: whether the requirement is a Non Functional Requirement (NFR) of the application or a Functional Requirement (FR)
- Description: a clear but concise description of what the requirement entails
- Priority: low, medium or high priority
- Phase of implementation: Early (weeks 1-3), Mid (weeks 4-6), End (weeks 7-8)

S.No	NFR/ FR	Description	Priority	Phase of Implementation
0	**Requirements of Existing version**			
0.1	FR	Upload Box for Users to upload CSVs		
0.2	FR	Show visualisations for the single CSV file with author data		
0.3	FR	Show visualisations for the single CSV file with submission data		
0.4	FR	Show visualisations for the single CSV file with review data		
0.5	FR	Export Visualisations to PDF		
2.3	NFR	The charts components should be reusable		
1	**Make existing visualisations more meaningful for the user.**			
1.1	FR	Add labels for the x and y axes for charts	Medium	Early
2	**Add more visualisations to provide more value to the user**			
2.1	FR	World Heat Map for submissions per country for author data file	Medium	Early

2.2	FR	Visualisations for data from the watchlist file	Medium	Mid
3	**Improve the user interface to make the system more useful and appealing**			
3.1	FR	Navigation Bar for users to navigate between pages	Medium	Mid
3.2	FR	Home Page with instructions on how to use ChairVisE	Medium	End
3.3	FR	Show users error message when unsupported CSVs are uploaded	Low	End
3.4	FR	Organise Visualisations into cards for increased comprehensibility	Low	Early
3.5	FR	Include usage instructions so users know the expected behaviour of the system	Low	Early
3.6	NFR	The app should not crash if unsupported file(s) are uploaded	High	Mid
3.7	NFR	The app UX should make the app easily usable and accessible	Medium	End
3.8	NFR	The app UI should be pleasing to the eye	Medium	End
4	**Creating visualisations by joining multiple csv files**			
4.1	FR	Show insights from data across the files- review data, submission data	High	Early
4.2	FR	Show insights from data across the files- author data, submission data	High	Mid
4.3	FR	Show insights from data across the files- author data, review data	High	Mid
5	**Session Management**			
5.1	FR	Sign up page for user to register using email and password	High	Early
5.2	FR	User can login using their registered credentials	High	Early

5.3	FR	Logged in users can log out of the app using a logout button	High	Early
5.4	FR	Non-logged users can only access SignUp and Login Page	High	Mid
5.5	FR	Users can log in using Google Account	Medium	End
5.6	FR	Logged in state persists even when user restarts app	High	Early
5.7	FR	Users can save a visualisation page session to the cloud	High	Mid
5.8	FR	Users can view, load and delete sessions that are saved to the cloud	High	End
5.9	FR	Show User Avatar to indicate currently logged in User. Google Profile photo shown for Google Account	Low	End
5.10	NFR	The app should have persistent storage to save states across sessions	High	
6	**Mapping Schema(s)**			
6.1	FR	Users can map the columns of a single uploaded file to a specific pre-defined file type	High	Mid
6.2	FR	Users can map the columns of multiple uploaded files to specific file types	High	End

Design

Existing Version

The existing version of the ChairVise application uses Vue.js and charts.js for the front end web pages and visualisations. It is powered by a backend Django server that generates insights from the uploaded files. These two high level components communicate via axios requests and responses. Figure 1 shows a high level overview of the different components of the application.

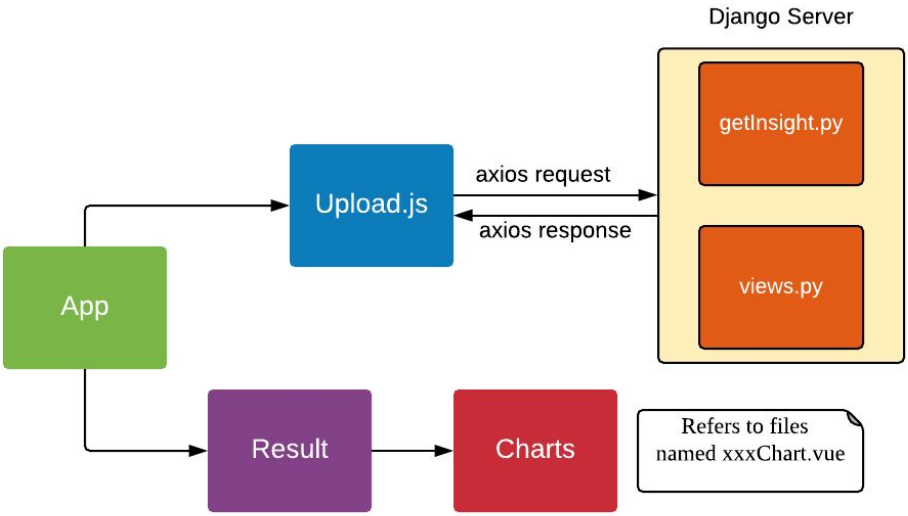


Figure 1. High Level Components of existing implementation

FIgure 2 shows the flow of control of the application. When a file is uploaded to the upload box, it is sent to the Django server for processing. The Django server uses the data from the CSV file to generate insight data which is sent back to the frontend. This data is then routed to the appropriate visualisation component that generates the webpage with the visualisations.

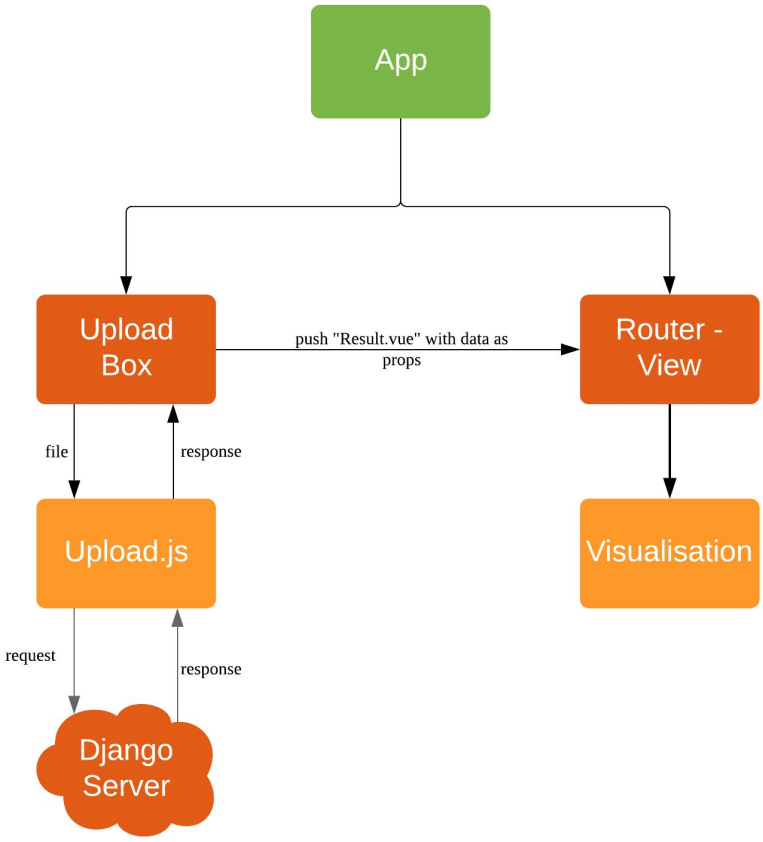


Figure 2. FlowChart of how ChairVise processes CSV files to get visualisations

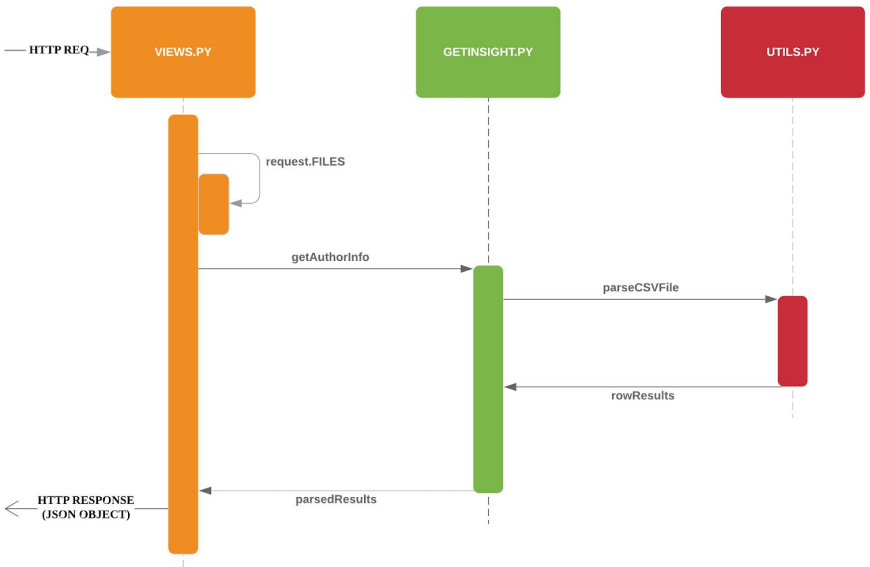


Figure 3. Sequence diagram of the backend of existing implementation

New Version

Development process,

We adopted an iterative and incremental development process by using the Agile framework. In the beginning, we defined the requirements and allocated them to the respective team members and worked on them in phases. Tasks that were given higher priority was completed first and features that would bottleneck the development other features were completed first. For example, we completed the login process in the early stages so we could begin on session management. The length of our sprints lasted generally from 1-2 weeks, depending on the difficulty of the feature. As we developed the product, more ideas were generated and we adjusted accordingly.

First phase of development (Weeks 1-3)

We spent the first week setting up the project and deciding which libraries to use for our project. We decided on using Firebase for user authentication and session management, and VueX to maintain a consistent state. The Login feature was implemented early to facilitate the implementation of session management. Basic UI improvements were made such as organising visualisation into cards. The general architecture of the program was decided.

Second phase of development (Weeks 4-6)

We implemented most of our features during the second phase of development. We used bootstrap to implement the login page and navigation bar. Improvements were made to the UI as it deviates away from the original look. As more files were released, we tried to create more visualisation using the files. Visualisations were also refactored into individual components. Initially, we decide to save user state by saving the entire DOM but decided to switch to saving chartData into firebase instead, thereby adopting the database session state management pattern. VueX was used to store the current logged in user so it will be consistent throughout the program.

Third phase of development (Weeks 7-8)

As we proceed to the final phase of our project, we began formally documenting our development process and providing documentation using the artefacts that were generated during the project.

Design Diagrams

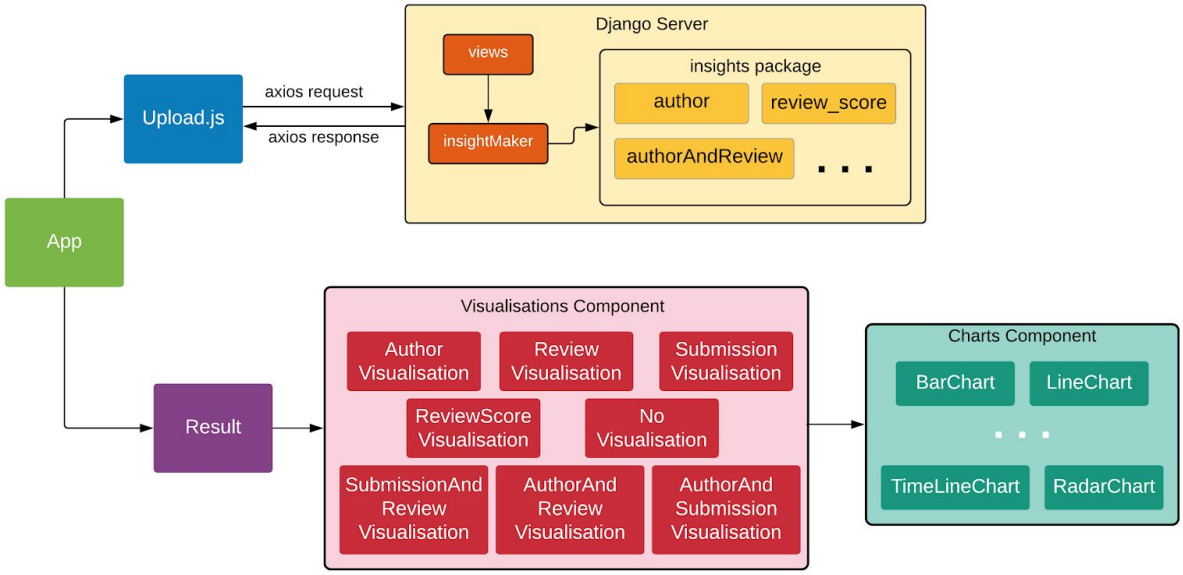


Figure 4. High Level Components of improved version

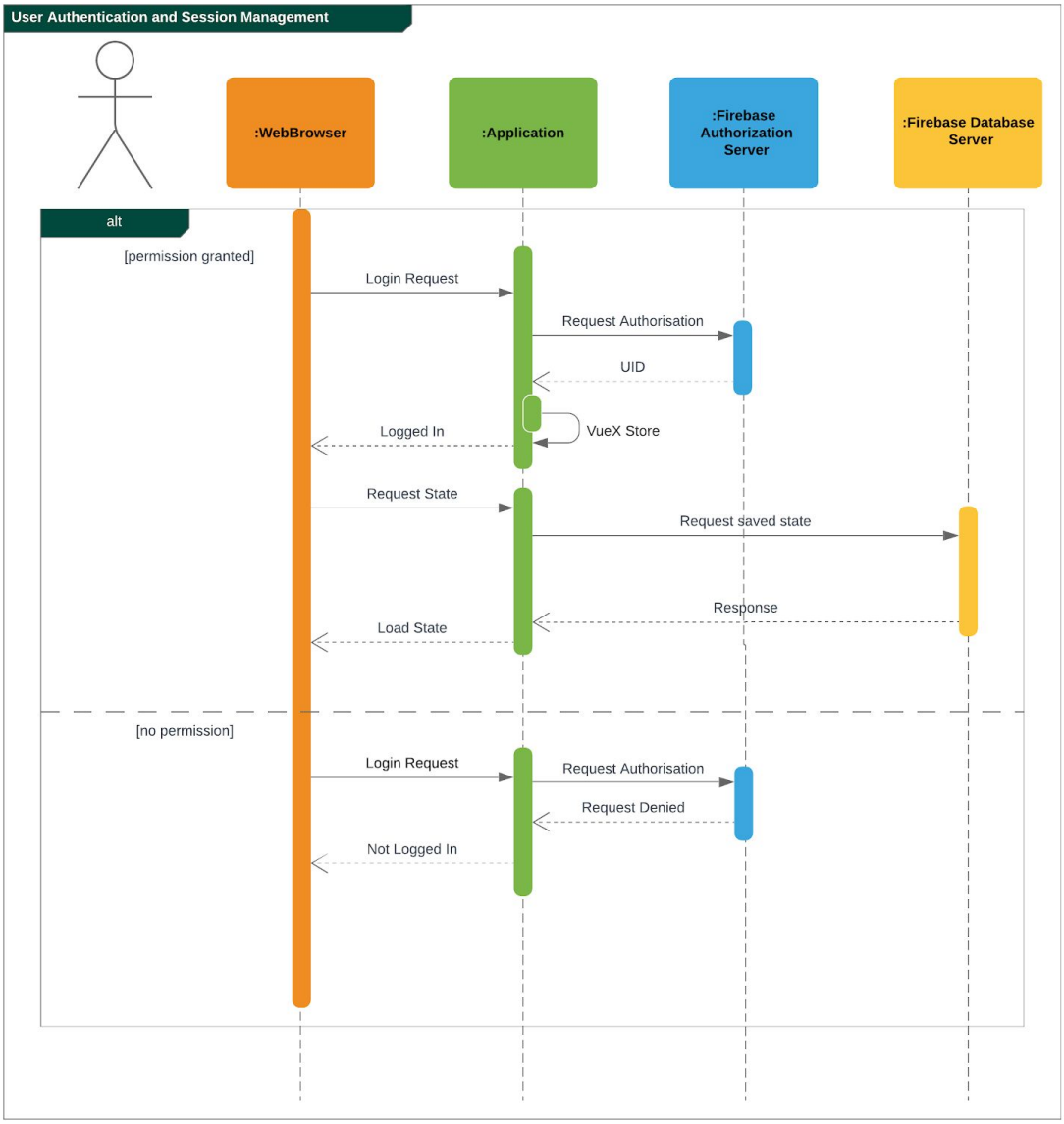


Figure 5. Sequence diagram of the Login Implementation

Design Decisions

- **Modularise Visualisations in the Frontend into Logical Components**
    - Benefits:
        - Reduce complexity when adding and editing visualisations, since now, Result.vue does not contain all the code for visualisation but rather, all possible Visualisation Components (**Separation of Concerns**). To add a new Visualisation Component, just add create a new Visualisation Vue file and import it into Result.Vue. To edit a particular Visualisation, just go it's corresponding Vue file without digging much into Result.vue.
- **Vuex - state management pattern + library**
    - Benefits:
        - Instead of passing around current logged in 'user' object between Components using Vue.js props and events, we implement a single source of truth: Vuex Store. Now, if a component wants to get the logged in user's property (eg. their profile Photo), the Component just have to query the Store. (**Single Source of Truth**)
        - Debugging of user state is easier, since it is stored in one place: the Store instead of in multiple components
- **Firebase Authentication**
    - Benefits:
        - Firebase Authentication provides backend services and easy-to-use SDKs. It supports authentication using passwords, phone numbers, popular federated identity providers like Google. Integration was easy and it saved us the trouble of creating a backend to manage users, thus more time could be spent on other features.(**Abstraction**)
        - Security of user account is guaranteed by Firebase. If we implemented the backend of logging in ourselves, the Security Protocols we will likely implement in only 8 weeks would definitely not be adequate.
        - User's Visualisation session states easily stored in the NoSQL Realtime Database provided by Firebase
- **Most heavy Data Processing done in Backend**
    - Benefits: Dedicated Backend reduces reliance on User's hardware to process data
    - No choice: Neither firebase nor Vue.js can do heavy data matching and processing, we must generate our own backend.
- **Separate Backend Files (and functions) for different CSV File Combinations**
    - Benefits:
        - No need to use another Backend Database to query for insights
        - Faster than using database queries
        - Size of the data is small enough to store in memory, since we only store it for processing, i.e., temporarily

        - The data from the files do not need to be stored in a database for our implementation of state management
        - Easy to maintain and add new features
    - Drawbacks:
        - Not as scalable as traditional Database querying

Suggestions for improvements and enhancements

Improved User Interface

- Full-Page dropbox
    - To make the drag-and-drop feature more intuitive for users, we could make it such that a user can drop a file on any part of the webpage instead of a specified box.
- Material Design
    - To create a visualisation that unify the user experience across multiple platforms
- Different types of charts
    - Include more charts types other than bar and pie charts
- Interactive charts
    - Current implementation allow user to adjust the number to show
    - Charts could be interactive and allow user to hover, zoom and pin-point over an axis

Support for Mapping Schema from general csv files to standard mapping

- Implementation: We implement mapping schema between frontend mapping and uploading csv file. Despite send csv file to backend jango, we also send a mapping schema as JSON file generated by user.
- Mapping Single Type of File: This can be easily done by user and send a single csv file with its mapping
- Multiple Types of File: Mapping will send in an array of JSON

Support for more file types and file combinations

- More file types
    - Current implementation only supports CSV files. It could be extended to support more file types
- More file combinations
    - Combination of different files to give more insight into the data instead of just visualising
- Interactive charts

Support for resetting password

- Ability to change passwords
    - Current implementation does not allow user to change password
- Ability to retrieve forgotten passwords
    - Current implementation does not allow user to retrieve forgotten password

Support for access-control

- Ability to share a saved session
    - Share a saved session with other users, allowing them to edit and save and collaborate on the same file using a unique url

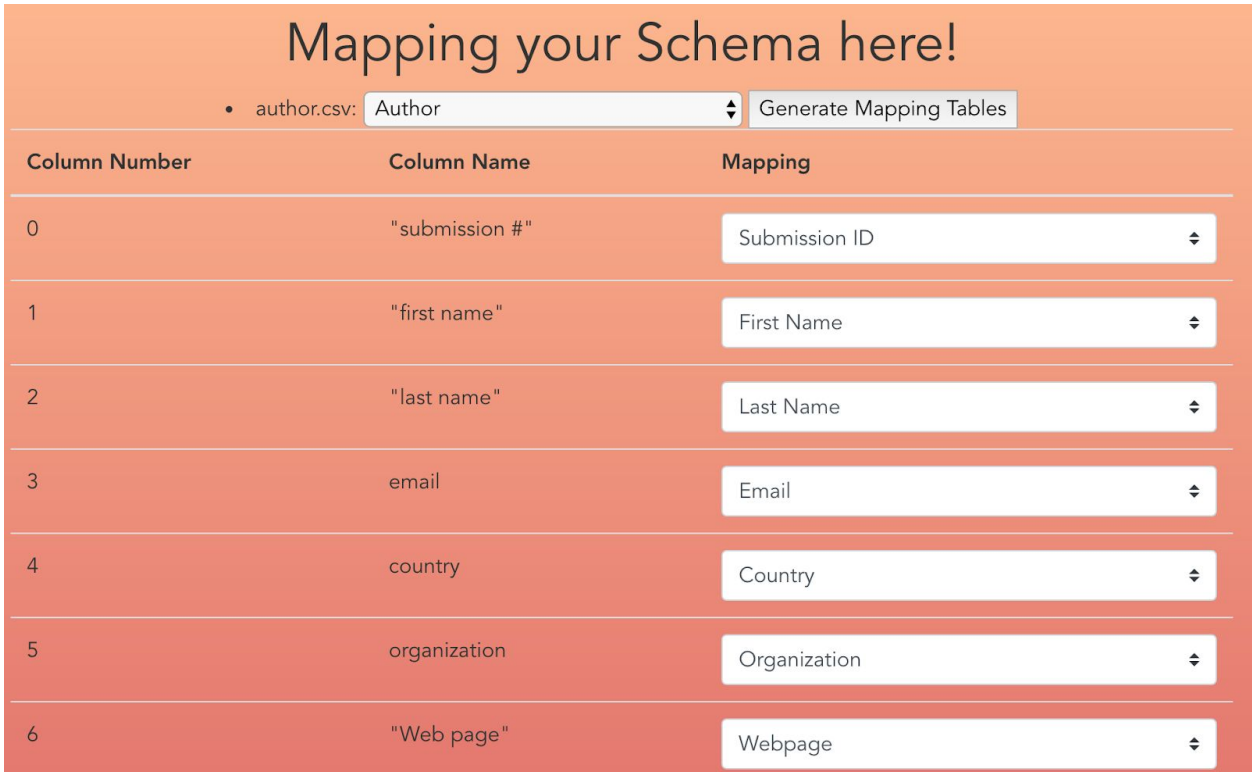
Support for exporting in other formats

- Support for graphic format
    - Current pdf file produced is huge, more than 10mb
    - Support for graphic format such as png, jpeg could reduce the file size

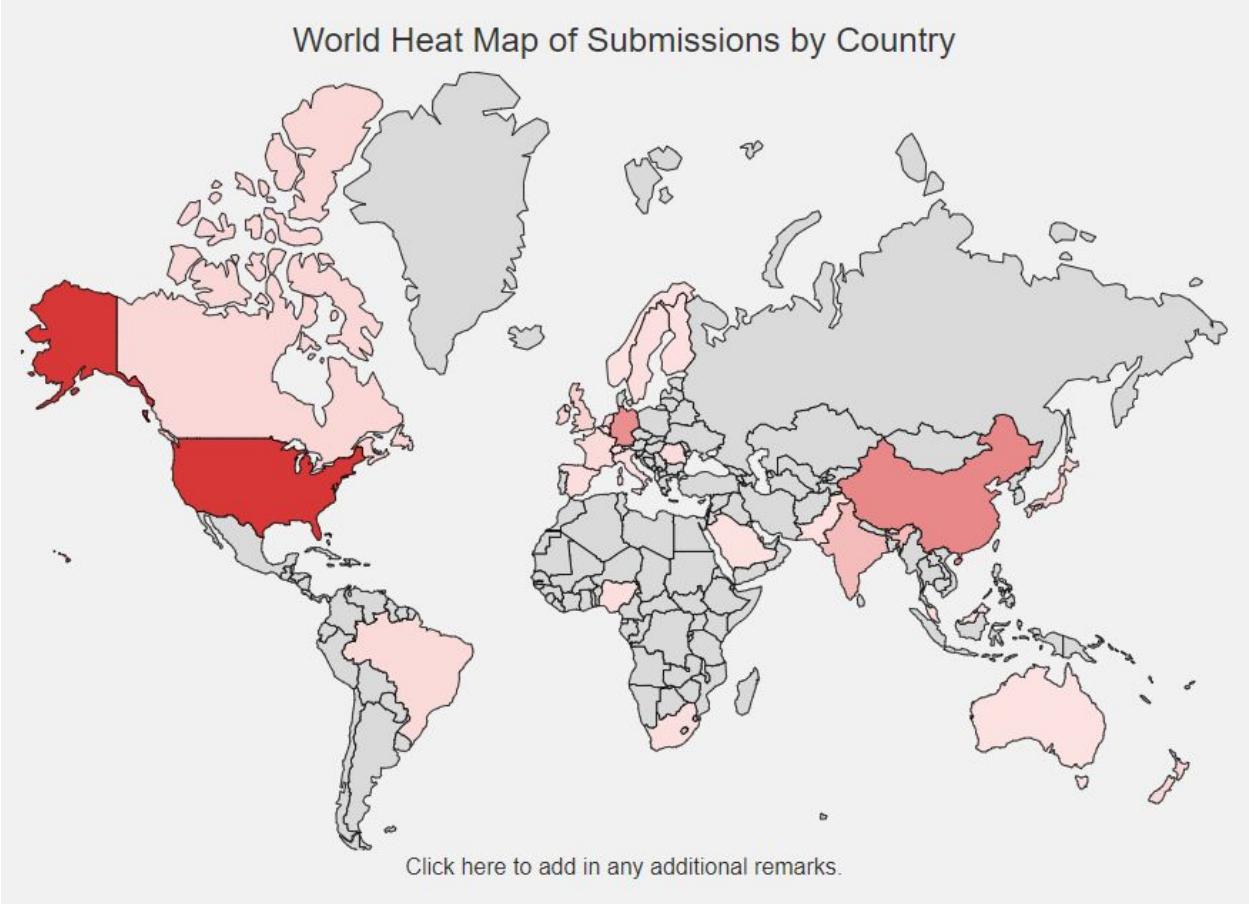
Visualisation Screenshots

This section shows the new visualisations produced by the ChaiVise application and some of the improvements to existing visualisations.

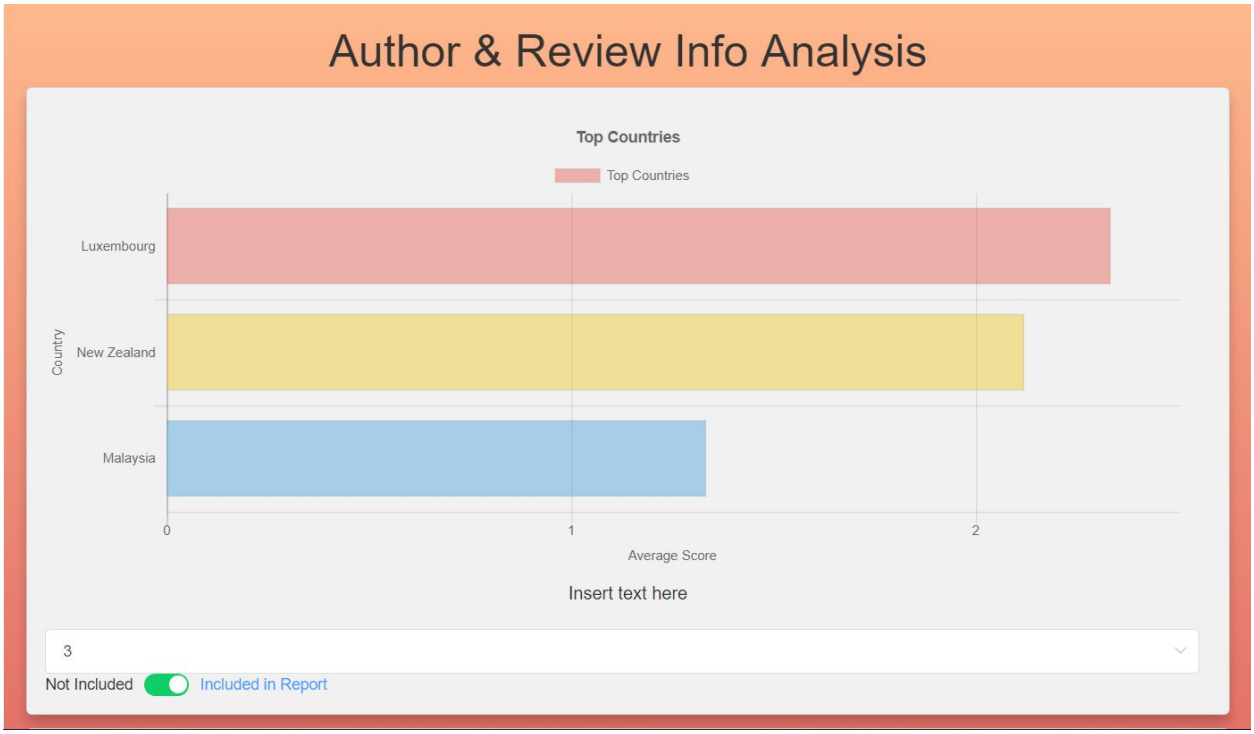
Screenshot 1: Home Page

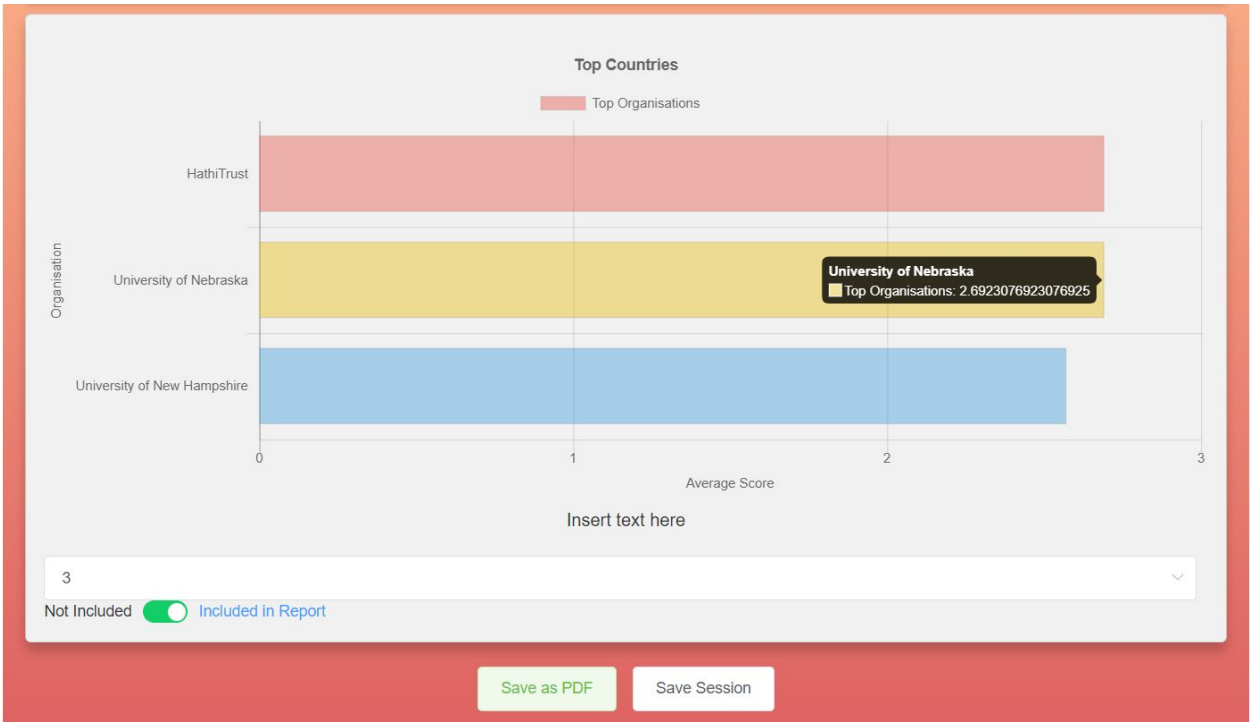


Screenshot 2: Mapping Data Interface



Screenshot 3: Country Distribution Diagram





Screenshot 5 and 6: Result Charts