E-poster board

FYP 17008

Final Report

Luo Kairen
Supervisor: Prof. Lau Francis
The University of Hong Kong
Department of Computer Science

April 15, 2018

Abstract

E-poster board is a web based content management system (CMS) designed for managing a large amount of digital posters with configurable policies. In this report, a new e-poster board system that utilizes powerful programming technique such as MVC, ORM and RESTful API will be implemented. The goals of this e-poster board system are to provide a flexible and user-friendly way of presenting interactive poster content to audience, and to increase information retrieval efficiency in order to enhance social engagement between poster readers and poster providers.

Keyword e-poster touchscreen CMS web

Acknowledgement

I would like to express my special thanks of gratitude to my final year project supervisor Prof. Lau Franics who gave me the invaluable opportunity to do this FYP project on the topic e-poster, which helped me learn software system development and as well as my Technical English class teacher Mr. Patrick Leung who helped me improve my academic English skills.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Sharing information using a large display panel within an institution has become a common practice in both private and public space.[1] Delivering event information and notices through a digital media is far better than paper posters in terms of management and speed. Also, public digital displays save the effort for poster readers to actively search for information from social media or emails and therefore they help readers focus on closely related information.

However, most existing displays are designed for distributing content in a broadcast manner and only few of these displays really have interactive capacity such as sorting contents according to their categories and discovering further information based on keywords. This one-way sharing channel hinders the distribution of interactive multi-media content and stop readers from actively

exploring events that meet their personal preference. Due to the constraints of display size and lack of interactive features, readers may find that information on the non-interactive e-poster board too generic and irrelevant. The lack of control over the type of information and its breadth and depth make reading traditional poster become an exhausting and inefficient job, and this prior investment of time and energy prevents reader from getting involved into social events they may be interested in.

In the last few years, thanks to rapid growth in smartphone market, demand for touch screen panel with high resolution and high precision has greatly increased. Price of touch screen has been driven down by this trend in the meantime.[2] It is now practical to use large affordable display panel with touch capacity to replace traditional poster board and display with no touch capacity to provide better information service for organization that need an attractive way to promote events or activities.

1.2 Motivations

First, physical posters are hard to be organized and managed. Unlike digital posters which space and printing cost are not a major concern, physical posters have the disadvantage of inefficiency in term of poster board space usage and difficult arrangement for posters with various size and different topic. As shown in figure 1.1a, it is quite common to see when the number of posters become large, it is unavoidable to place posters over one another. Without

(a) Posters placed on top of another poster

(b) Method used to update a physical poster in campus

Figure 1.1: Pictures to illustrate problems on physical posters

an automatic solution to organize posters according to their content and submission time. It makes searching for posters a time consuming task for viewer to find latest and relevant information. Second, information on physical poster tend to be inconsistent with the official source. Once a posters are printed, updating them is labor intensive and it usually cannot be done in a short amount of time. The lack of resource to do full scale update as well as the time gap between updating cause out-dated information on physical posters. An example of mega sale poster is shown in figure 1.1b. The inconsistent problem can be solved if posters are stored in digital format and synchronized with a center database.

1.3 Objectives

The aim of this project is to build an electronic poster board that can promote information sharing and enhance poster viewing experience. It is hoped that through the use of the new e-poster board, the social tie between poster readers and poster content providers will be greatly strengthened and readers could feel that it is more intuitive to be informed of new events and get involved in the community.

1.4 Scope

This project serves as a content management system(CMS) behind any arbitrary e-poster board hardware with display screen and touch capacity. This system will provide a content management system for content provider to create and submit interactive poster, an web based poster viewing interface that follow rules and policies designed by system administrators and a data analysis platform for providing information retrieval tasks.

1.5 Outline of the Report

The outline of this report is as follows. First, some leading digital poster solutions will be analyzed to explain why they are not suitable for our usage scenario. Then an implementation of e-poster system that avoid drawbacks in previous works together with nu-

merous additional enhancements like real time board update, a compatible search engine that can index posters and a recommender system will be introduced and then we will evaluation its performance. After that, we would demonstrate some difficulties and limitations encountered during the system development phase and their mitigation solutions.

Chapter 2

Related Works

There are several existing e-poster systems on the market. A brief analysis will be conducted on squareVIEW, iPosterSessions and ePosterBoards.

2.1 squareVIEW Digital Posters by Digital Media Systems

SquareVIEW is digital poster hosting solution provided by Digital Media Systems for different scenario such as "displaying products & adverts, menus, employee communications and welcome messages".
The reason I pick squareVIEW as the first example is that there are many digital posters solutions that are based on "Android Freestanding Digital Posters" embedded system, a 30-50 inch standing touchscreen monitor, that has been widely used in Hong Kong. In order to avoid duplication, I selected that most representative

provider to discuss.
See Figure 2.1. They are android freestanding digital posters for displaying poster in picture, video and powerpoint format. When poster providers need to update poster content, they can transfer files from a USB stick to the digital poster device, therefore squareVIEW Digital Posters are usually used for files that does not need to be updated over the internet or internet connection is not accessible. [3]

Figure 2.1: squareVIEW Digital Posters

2.2 iPosterSessions by aMuze!

The iPosterSessions is commercial Software as a Service e-poster system provided by aMuze!. From the description of aMuze!, iPosterSessions is built for scientists and scholars to present their research posters electronically and interactively.[4] It allow users to host poster in form of webpage. The system support multimedia playback such as image, video and audio. See Figure 2.2. As it is intentionally designed for aMuze!. Each e-poster screen will

only display one poster and there is no navigation functionality to switch to another poster. Besides that, the layout of poster that can be used on iPosterSessions is limited to a set of predefined template. Users need to build their poster with iPosterSessions' template and users cannot reuse posters that are designed using other software since they are incompatible with the system.

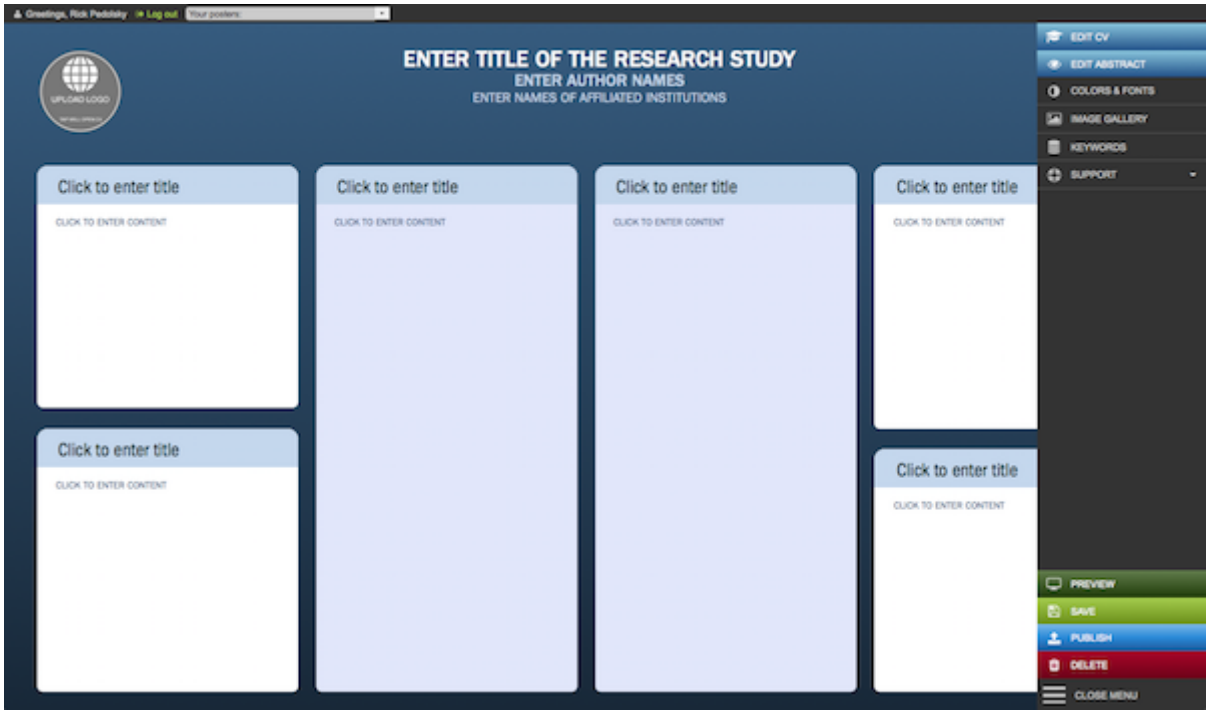

Figure 2.2: iPosterSessions

2.3 ePosterBoards by ePosterBoards LLC

ePosterBoards LLC is an event technology service company based in greater Boston area. Their e-poster solutions ePosterBoards is an all-in-one system for rental which hardware and software work as a whole. Their solution features responsive design available on all devices, analytics for tracking ePosters viewed and user-friendly interface.[5] See Figure 2.3. Each e-poster screen can only show one poster at a time. For e-poster format, ePosterBoards only accepts pdf files that are created based on predefined template, and their system does not support video or interactive elements.

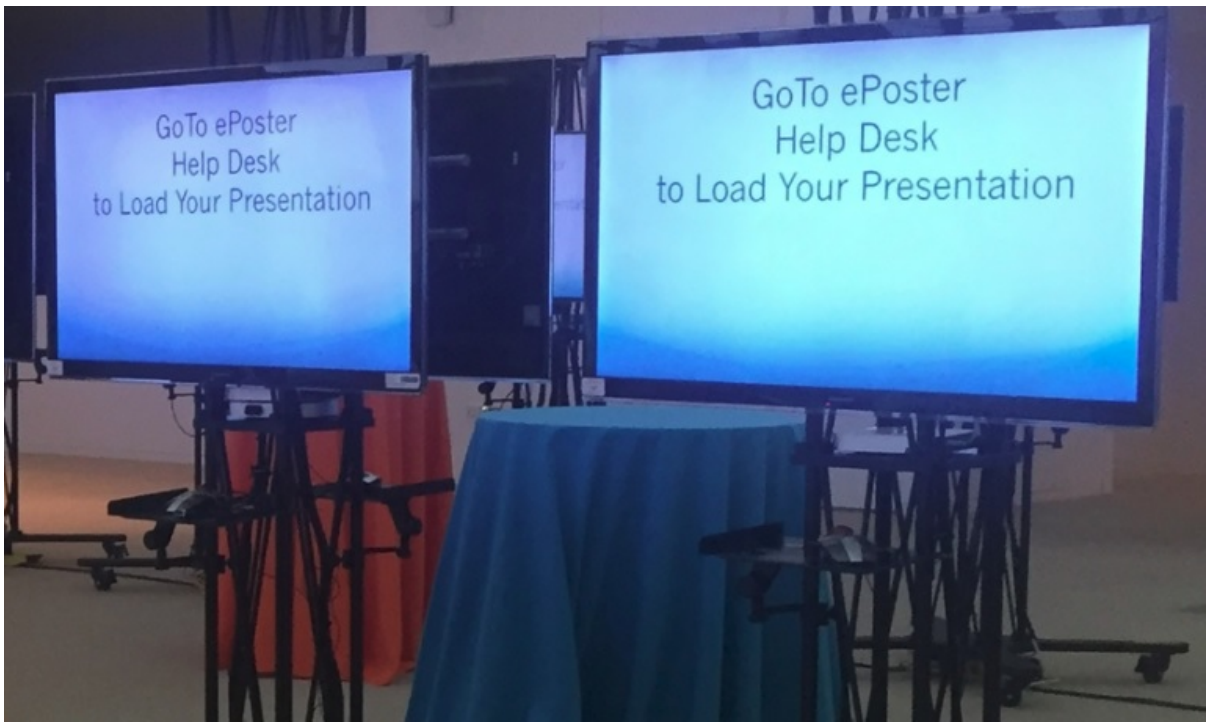

Figure 2.3: ePosterBoards

Functionality	squareVIEW	iPosterSessions	ePosterBoards
Interactive Support	X	✓	✓
Management over Internet	X	✓	✓
Multiple posters on one device	✓	X	X
Support for standard format eg, Portable Document Format(pdf)	✓	X	✓
Posters Searching	X	✓	X
Posters Recommendation	X	X	X
Self-host	✓	X	X

Table 2.1: Comparison of different e-poster solution on the market

2.4 Limitations of existing products

There are several drawbacks for e-poster products on the market based on observation from section 2.1,2.2 and 2.3. Details are list in Table 2.1

Old e-poster solutions suffer from hardware constraints. E-poster systems such as squareVIEW Digital Posters is built on hardware that does not have touch screen panel and internet connectivity, such that their poster contents cannot interact with

user by tapping and swiping. And not being internet connected means that content management cannot be performed remotely and quickly.

In term of privacy concern, the fact that most commercial solutions don't provide self-host functionality which may stop people from using the poster system. Organization may not want to share their internal data with an external party especially for government and Health care entities, and also deploying software solutions without self-host functionality may violate regulation compliance policy.

For ease of adaption, some existing e-poster systems such as ePosterBoards only focus on the certain situations like conference and large events, such that each e-Poster is displayed on one terminal as space is not the primary concern. Support for multiple posters on single screen is usually missing on these types of commercial application. Due to the constraints of space and building infrastructure , not being able to display multiple posters one screen makes e-poster adaption very hard for low-budget users.

Also, we expect e-poster system to be more intelligent. Instead of only performing the print-and-read duty of conventional paper, readers may hope digitalization of textual and image content could provide better services such as poster searching and recommendation services.

In short, after research, we found that no existing product on the market can fulfill all requirements we need for delivering the best potential that a digital version of poster system can give us.

Chapter 3

Methodology

This chapter serves as an overview of different system designs used in the system. First part is the system's architecture that includes an abstraction of software implementation and communicate routes between system parts. Second, the system is separated into components according to their functionality. This both helps system implementers understand the role of each component and it also helps system implementers communicate using same set of naming convention. Lastly, we will introduce some possible techniques to improve poster retrieval efficiency.

3.1 System Architecture

There are several considerations before starting building the system. Since the purpose of this project is to demonstrate potential of a modern e-poster system, design of the system focuses on usability, maintainability and extensibility. In the first iter-

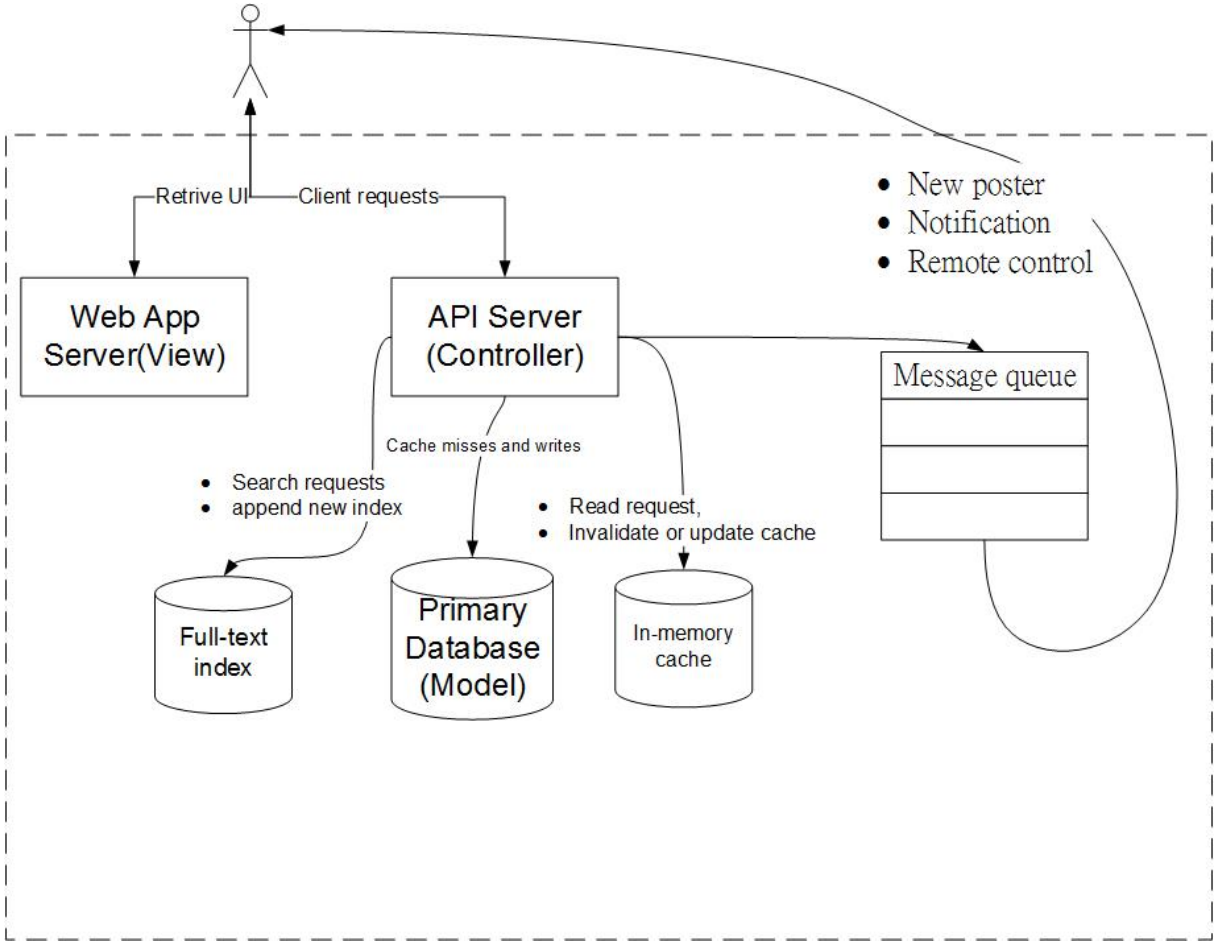

Figure 3.1: Architecture of different infrastructure parts

ation, maximum usability is the foremost thing to consider. A system working prototype with monolithic structure will be built, and based on users' feedback, the system could quickly evolve to address possible issues and additional requirements. In the next stage, as shown in Figure 3.1, front-end user interface and back-end system will be separated into two parts and communication between them are replaced by API interface instead of direct function call. This helps separate concerns between drawing a user-friendly user interface and building a robust back-end system. Decoupling of unrelated parts help make the system to be more maintainable and extensible.

3.2 System Components

All major parts of the system are listed in Figure 3.2. The general system component design would serve as a guideline for separating the system into different tier which can be implemented and tested independently. They are namely Presentation Tier, Application Tier, Data Analysis Tier, Storage Tier, Admin Console. For Admin Console, this system component would potentially interact with all four tiers, so it is shown on the side of the diagram.

Presentation Tier is responsible for rendering interface display and translate interface interaction into system requests that can be processed by Application Tier. Application Tier is to do the heavy lifting job for cooperating resource from system infrastructure and

third party service to provide service. Data Analysis Tier acts as a middleware. Unlike conventional website architecture, the reason we introduce Data Analysis Tier is to improve data processing efficiency and to analyze requests from Application Tier to Storage Tier, and generate feedback data back to Application Tier, Many services such as recommender system and search engine depend on utilities functions in this Tier such as data cleaning and transformation. Storage Tier is an encapsulation of ORM and other utilities that will interact with the database infrastructure.

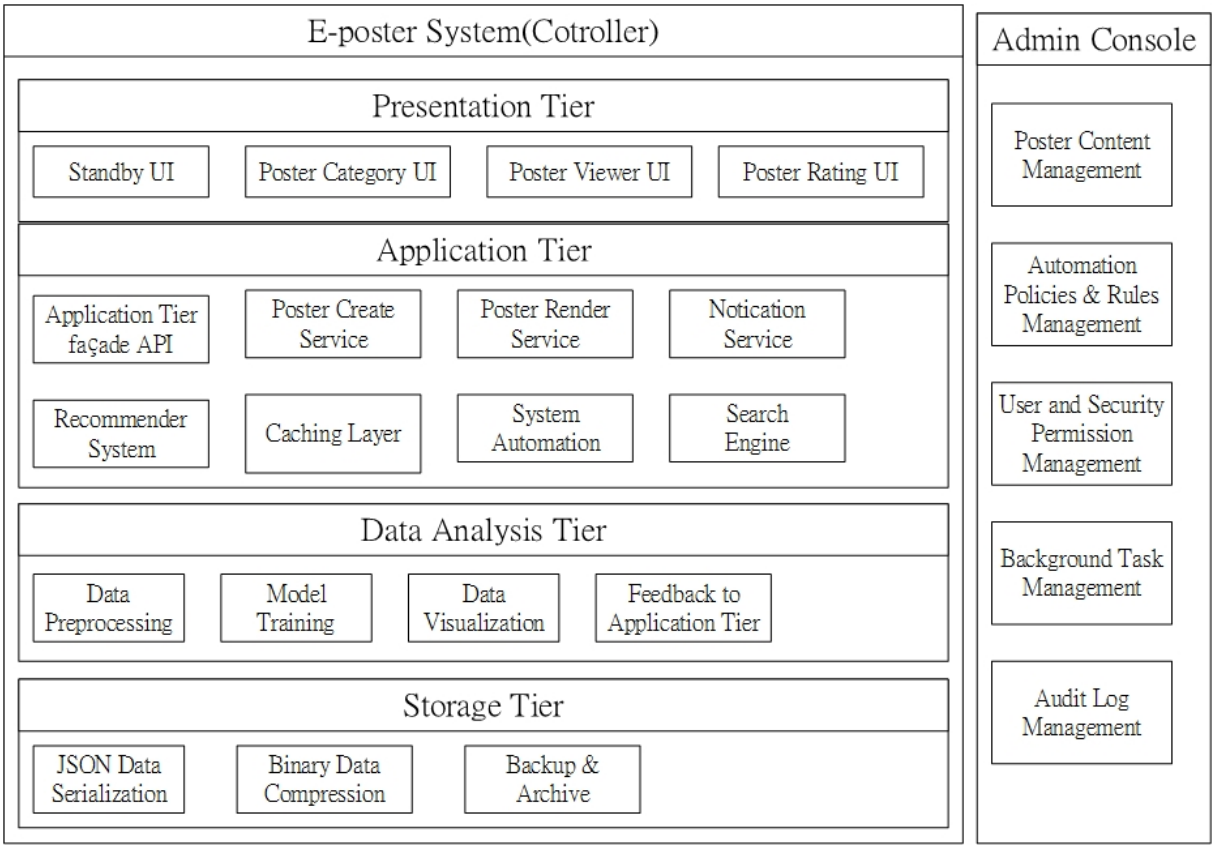

Figure 3.2: An overview diagram to illustrate system components and their hierarchy.

3.3 Efficient Way to Access Poster

When we got lots of posters in our system, a crucial problem arise. How can we efficiently deliver important and relevant information to our users without overloading users with posters that they are not interested in. As [6] points out, the problem is not about the volume of information. We need to have a smarter filter mechanism to tell us what we need to know. In some cases, if we already understand what you want to know, we can directly put our keyword in the search engine. However in most of the times we have little understanding the best thing available, so we need to have recommender system.

The recommender system that we are going to build is based on algorithms that meets following criteria. First, it has to balance between performance and recommendation quality. Since the number of posters will keep growing, the system must be scalable for future usage case. Second, prediction results must be reproducible to enable comparison with user's real choices using root-mean-square error(RMSE). This is essential since it provides an optimization direction for improving recommendation quality both automatically or manually. Finally, a potential parameter in the model may include reader's location, current time, posters' vote count, posters' submission time, event venue and event schedule, and the recommendation model must be able to increase the number of parameters and place weighting on its parameter, because some information is more important, and it is hard for a learning algorithm to get the sense of importance without humans' help.

3.4 Development Environment Setup

The system will be developed and tested under following environment and unless otherwise specified, all system performance evaluation and optimization will be based on this setup.

- Hardware: virtual machine(VM) that has one core of Intel i7 CPU and runs on Openstack Icehouse distribution(HKU FYP server)
- Operation System: Ubuntu 14.04 LTS Server
- Backend Platform: Spring Boot 2.0.0 M7
- Frontend Platform: React v16.0.0
- In-memeory cache: Redis 4.0 stable

Chapter 4

Implementation details

This section serves as an example of a purposed implementation for the e-poster system design. This e-poster board system comprises a user interface that suits terminals with different screen size and a backend server running in service provider's machine, which provides posters content data. The backend server can also response to terminals' requests for different type of services.

Beside the user interface and backend server which are two main parts of system, several other system communication standard and system parts will be discussed separately. They are the purposed poster format, real time board update protocol, and the recommender system for providing poster suggestion feature.

4.1 Terminal User interface

The terminal user interface is a web based single-page-application together with multiple predictable state containers and the interface's functionalities include retrieving current posters information on the server and starting WebSocket workers to subscribe any subsequent poster modifications or broadcast events.

Information exchange between the terminal and the server or internal communication occurs by looking up corresponding API interface and initializing HTTP request in predefined message format. A consistent way to exchange data is convenient for analyzing network communication and making changes to the state container which reduces the chance of getting into unforeseen system status, which enhances user interface stability.

To optimize user experience on a touch screen, a user interface design principle called material design is putted into practice. According to the description on Material Design's homepage, Material Design provides a framework that provide a unified experience for cross-platforms and devices with different screen sizes. [7] Our system adapted the design principle as well as many standard components from Material Design as you can see in Figure 4.1, 4.2, and 4.3 which shows the system user interface on poster searching, poster viewing and poster recommendation respectively. The aim of adapting material design is that the familiarity of material design may help the first time user navigate the system and it also

make the system responsive to different touchscreen size.

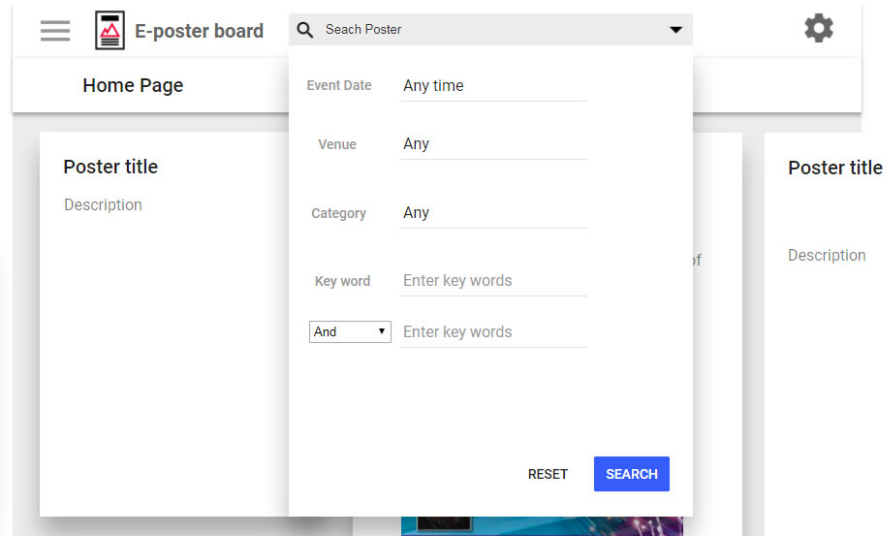

Figure 4.1: User interface for searching posters

4.2 System backend server

4.2.1 Database design and set-up

The system's database is managed by an object-relational mapping tool Hibernate which supports all major database vendors and acts differently to expose a high level programming interface. This saves an institution from worrying about the compatibility issue with low-level database management. The overview of database structure are shown in Figure 4.4. This ER diagram explains the relationship between an e-poster with other database entities. Each entity is stored in a separated table which is serialized and stored in primary database and each record in a table has a unique

Figure 4.2: User interface for viewing posters

Figure 4.3: Standby page for providing poster recommendation

key which is denoted as PK on the diagram.

Poster table is used to store e-poster items. Poster entities store poster content of desired poster format in poster_content attribute together with a rendering snapshot to be shown on the poster showcase layout. Each poster has zero or more event entities which would be essential for providing map navigation and event registration service. Event's venue information is also very important for providing poster recommendation to specific target group. Details about the recommending system will be discussed later.

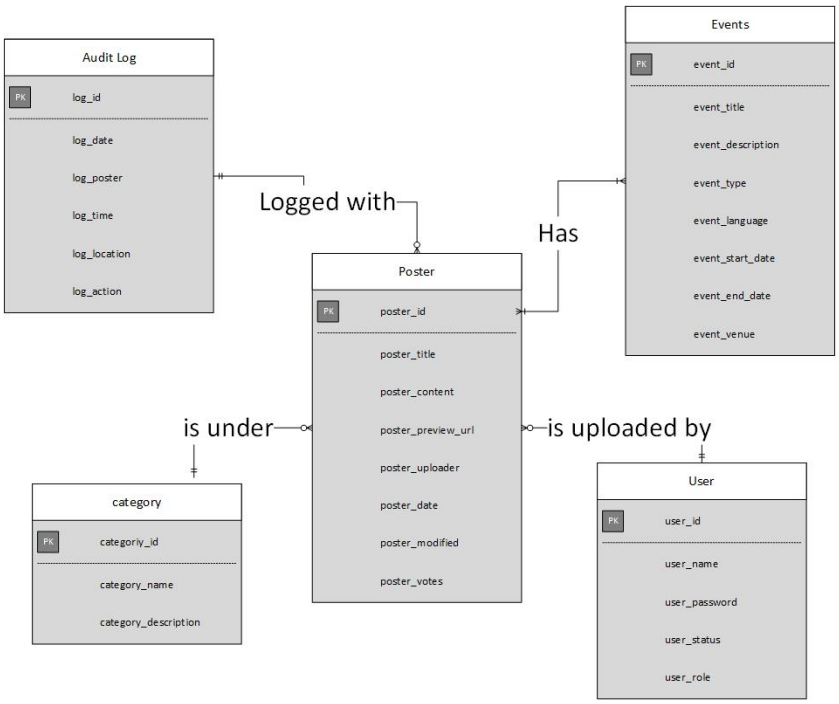

Figure 4.4: An ER diagram of the database setup.

4.3 Purposed Poster Format

There are many potential candidates of poster format, namely plain pdf format, plain html format, pdf format with HTML element layer, pdf converted into image with HTML element layer and pdf converted into Scalable Vector Graphics(SVG) with HTML element layer. Each candidate has been used in industry for many years, especially plain pdf and plain html are simple to implement and have best compatibility with embedded digital devices. Previous works squareVIEW Digital Posters and iPosterSessions are both built based on these two poster format.

However among all these formats, the last option pdf(svg) with html elements is chosen because pdf(svg) doesn't have major defects like other formats in common e-poster usage scenario. See Table 4.1. For plain pdf format, it has a major disadvantage which is poor interaction with the web platform like not being able to play audio and video since Adobe pdf plugin is built on a browser plugin architecture NPAPI which has been phased out due to security reasons.[8] [9] [10] For plain HTML poster format, this format is best suited for expressiveness and performance in web platform. The only drawback is that creating a HTML file that work correctly and look like a pdf poster file a never an easy task. A slight better version of plain pdf would be augmenting pdf with html elements. This brings the power of interaction into pdf file, but it also introduces the cost of incompatibility with mixing pdf and HTML contents.

So in order to combine that easiness of typesetting using pdf and the interaction capability of web elements, a compromised solution would be converting pdf into a web elements that can still perverse pdf's content. A primitive implementation would be converting the pdf file into a binary image format which would is one of the most supported element of the web standard. However using an image format to store the poster content is undesirable as it is slow to transfer over the Internets and it is costly for storage.

Among all other image file formats, vector graphics format svg stores image information with polygons and node attributes which is an acceptable replacement of binary image file in presenting a digital poster where majority of the poster content are about text and geometry graphics. Also, svg format can be converted into HTML canvas element which make svg format fairly easy to be adapted on web platform.

The working principle behind this svg-html hybrid approach is fairly simple. First, traditional poster file like an pdf file that may contains original contents like text and image is analyzed. Metadata like position, size, color and font used by a piece of text can then be recovered by referencing Portable Document Format (PDF) 1.7 specification provided by Adobe. Based on these metadata and multimedia resource extracted from the source file, they can be restructured and rebuilt with svg file format. Then additional html elements that provided interactive functions like event

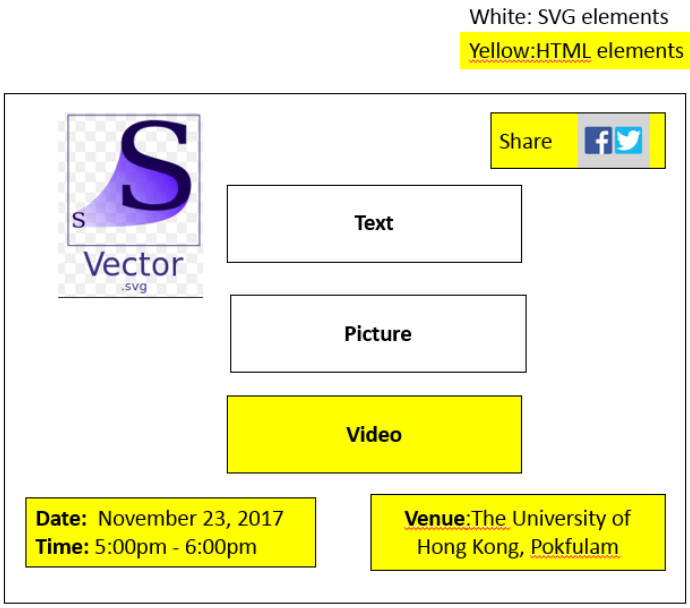

Figure 4.5: Placing HTML elements on SVG file

reminder and map navigation are inserted on the top of svg file, and css stylesheet is used to make sure these html elements appear in correct position as shown in Figure 4.5.

4.4 Real Time Board Update

As stated in the motivation, one of the key benefit of digital poster is the consistency of information between poster provider and viewer in real time. But unfortunately, the entire web platform is based on a protocol called HTTP(Hypertext Transfer Protocol) which does not allow server to actively send message to its client

Table 4.1: Comparison of different e-poster format choices

	Plain pdf	Plain html	pdf with html element.	pdf(image) with html element.	pdf(svg) with html element.
searching	normal, pdf's text content can be extracted and indexed, but the quality may not be satisfactory.	best	normal, same as plain pdf	normal, same as plain pdf	normal, same as plain pdf
interaction	worst	best	good	good	good
performance	bad	best	worst,	bad - good, image's loading performance largely depend on network speed	good
storage	good	best	good	worst	good
compatibility with embedded platform	best	best	worst, behavior of drawing additional HTML element on top of a browser's built-in pdf viewer is unpredictable.	good	good
easy to create and modify	good, may need help from external pdf editor	worst, require lots of technical skill in creating html page.	normal	good, same as plain pdf	good, same as plain pdf

without client making a request first.

There are several common used methods to achieve data synchronization under this constrain. They are short polling, long polling and websocket. Websocket is chosen in this case as it is much more resource-efficient compared to other ways[11]. A metaphor on the working principle behind websocket is shown in Figure 4.6. When the client successfully retrieved and executed the page that need to be updated in real time, an instruction will be sent to browser's websocket opens a persistent tcp/ip connection with the server, which is like subscribing to an email list. After that, both server and client can communicate via this full-duplex asynchronous communication channel without blocking the main thread. On the other hand, short polling and long polling are like picking up a phone with a third party which short and long refer to the length of duration between each polling request. To approximate the effect of real time, the client side need to keep talking to the other end in order to confirm availability as well as to check for the presence of new information.

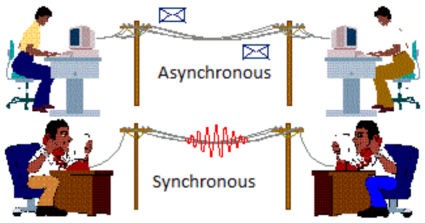

Figure 4.6: Asynchronous vs Synchronous Model

4.4.1 Real time communication API design

To reach real time data synchronization that is capable of performing board state delta update, a message format with special payload that can be understood by both server and client is designed.
There are three type of payload that can be sent by the server, they are

- hello. This message is used to agree on API scheme.
- ping A message from server to check availability.
- pong A message from client to confirm availability.
- error. Return error message on illegal or unexpected operations.
- action. Standard message type for all other operations.

The specification of the Real time communication API is listed as below.

Message format:

```
{
        "type": "MESSAGE_TYPE",
        "ATTRIBUTE1":"VALUE1",
        "ATTRIBUTE2":"VALUE2",
        ...
}
```

Possible values for attribute event:

Event	Description
Data sync related	
poster_added	A new poster was added,
poster_change	A poster's content had been changed
poster_down	A poster was taken down
System administration related	
open_poster	Force switch current screen to participle poster
reconnect	Try to reconnect websocket with a new server address
token_revoked	The JWT token is forcibly revoked to logout current user
Debug related	
show_message	Display a message on the screen
show_terminal_status	Display terminal status on the screen
refresh	Refresh the terminal

4.4.2 Board Data Synchronization Steps

In the first step, client send a HTTP request including Upgrade header to indicate this is a websocket connection, this is known as WebSocket handshake.Once the server that supports websocket protocol receives the request, it sends back an acknowledge response with HTTP status 101. Now an asynchronous communication channel has been established, server send a hello message to the client.

```
{
        "type": "hello",
        "version":"1.0",
}
```

If client cannot receive the hello message, a retry process will be started. From the hello message, client know the communication channel is working. Also, it confirms that the API version has been implemented in the client script. If not, the client page will refresh to retrieve the latest

implementation from web server.

In step two, server initialize a ping-pong checking procedure. For each minute, server send out a ping signal to all connected clients, and clients reply by pong to confirm it.

```
{
        "type": "ping"
}
```

,and

```
{
        "type": "pong"
}
```

Step three. Server logs down the current status of latest data set. If there is any modification on the data set, a message with payload type action will be sent to the client together with the corresponding action event.

```
{
        "type": "action",
        "timestamp":1516459270,
        "event": "poster_added",
        "payload":{
                "id":"1"
        }
}
```

For exception cases. There are many possible reason that real time update will fail. Some of the possible reason include timeout due to heavy workload, design error in implementation. Client can extract those error message to inform its user the exact problem.

```
{
        "type": "error",
```

```
        "timestamp":1516459270,
        "error":{
                "code":0,
                "message":  "Timeout",
        }
}
```


4.5 Poster Full-Text Search Engine

After converting a poster into the standard format as described above, text content will be extracted from the source file to be converted into an inverted index record. A well-structured inverted index table is essential for many text-related functionalities provided by the system to run an efficient manner, eg full text searching and poster content similarity analysis.

With the constrain of computing power and storage, we hoped that our inverted index can cover all text with minimal number of word vectors. However, due to the ambiguous nature of natural language, words composed with similar letters pattern can have completely opposite meaning, and words that share similar meaning can turn out to be different in digital representation. System that ignores the ambiguity in natural language can be extremely hard to use. An example would be case sensitivity. Two words 'hku' and 'HKU' is completely different with their ascii code but they both can mean "the University of Hong Kong". A good news is, according to the Zipf's law, the frequency of a word is inversely proportional to its rank in term of popularity in most corpus. Majority of the text is composed with a limited set of words. With some cleaning and special handling of rare words, we can obtain a clean corpus that is concise and efficient to represent in vector form. Therefore, preprocessing of raw text places an important role in natural language processing community.

There are some steps to build our search engine. They includes

1) Text preprocessing
2) Construction of Full Text Index
3) Ranking retrieved result

4.5.1 Text preprocessing

Lots of research have been done on the practice of text preparation. The most popular methods are tokenization, stopwords removal, stemming and n-grams. The processed text will be stored in main database such that these results can be reused in experiment and [12].

Tokenization refer to the process of extracting words from characters sequence in text. The purpose of tokenization is to help identify the meaning behind the string with any length. In English, words are separated by spaces or punctuations, but in many Asian language, words connected together. This require additional text processing technique like word segmentation based on corpus examples. To simplify the preprocessing step, this project will not cover non-English text.

Stopwords removal is to eliminate common words that has no or little semantic meaning such as ”the”, ”to”, ”be”, and ”of”. Because stop words are so prevalent, they serve nearly no purpose in identifying the unique feature of each document, and also discarding stopwords help reduce index size and increase processing performance.

Stemming combines different forms of a word into single representation. This is another common practice to reduce keywords number and consolidate tokens with similar meaning. Stemming can be performed using the dictionary approach and rule based approach. Dictionary approach is more accurate as the word transformation is suggested by linguists or fluent English speakers, but its drawback is the version of dictionary we are using to perform stemming might not be complete for all possible words. Rule based approach is to perform word transformation with algorithm

that derived from observation of common stemming rule such as removing ending ”-ing” and ”-ed”. One of the most popular rule based stemmer is Porter stemmer which has good correctness rate and is easy to implement.

The final step is n-grams. N-grams is picking contiguous n tokens from the cleaned text. The reason we do n-grams is that words may give different meanings they are combined together as phrases. Also since some machine learning algorithms depends on the source text that preserves word order to get better insight with text's meaning. Undoubtedly, full text lost some representation power when they are converted into unigrams. For example, two sentences ”It is good, but I do not like it.” and ”It is not good, but I do like it.” have completely different meaning, but we cannot predict their meaning if we are feeding our machine learning model with unigram dataset.

This information loss problem can be partially solved by increasing the number n in n-grams, but when n is large we will get noise like token ”good but I” which probably has little help to classify the text. Currently we pick n as 2 and we will store both unigram and bi-gram result together with raw text. This is a trade-off between performance and efficiency, and value n should be reviewed when the dataset grows larger.

4.5.2 Full Text Index

There are mainly two types of index we can use to accelerate our NLP, namely inverted index and signature file. Inverted index maintain a hash table of keyword to document-id lists. Each inverted lists maintains the ids of document that contains particular keywords. On the other hand, signature file is to convert keywords into a bitmap, which 1 in binary position i indicated the presence of that keyword i, and vice versa.
Past research suggested that in typical cases inverted indexes perform better than signature files and inverted index can replace signature file in term of functionality. [13] We decided to pick inverted index as our text index-

ing.

Query Processing

In refturtle1995query, query processing strategies can be classified as Term-at-a-time(TAAT) and Document-at-a-time(DAAT). TAAT processes terms one at a time for all documents and get back the document scores by summing up partial scores in all results. DAAT is to get the document score for each document for all query term which next document will be retrieved at the end of previous one. If we have a large corpus where IO is the bottleneck, we want to use DAAT to increase throughput by introducing IO parallelism to process tons of documents on different storage devices. TAAT is used when the corpus is small since it is easier to implement and understand, but it is rarely used in modern search engine since TAAT has limited performance improvement in distributed environment which makes it hard to scale.

Algorithm 1 Document-at-a-time Algorithm

```
1: procedure DocumentAtATime(Q:query,I:index,k:topK)
2:     L ← Array()                          ▷ Store relevant inverted lists
3:     Rank ← PriorityQueue(k)                                  ▷ Top k list
4:     for all terms q ∈ Q do
5:         L.add(GetInvertedList(q, I))
6:     for all documents d ∈ I do
7:         score_d ← score_d + Score(Q, d, L))
8:         Rank.add(score_d, d)
       return top k results from Rank
```

A simplified DAAT algorithm is written as below. See algorithm 1. When we evaluate using DAAT, we loops through each document d, and

calculate score_d in each pass. At the end, we will get back a priority queue of top k (score, document) pairs.

4.5.3 Search Results Ranking

The DAAT algorithm we used to retrieve documents from inverted index is working in boolean retrieval model which means there are only two possible outcomes for query evaluation. They are either true or false, relevant or irrelevant. This may not be very helpful for the users. In order to improve the search results, we are going sort documents with ranking score.

Two common approach are provided as score function for ranking the searching result. One approach is to compare them with the recommendation score discussed in section 4.6. Another approach will be comparing document by keyword relevance or so-called similarity. The exact way to do it will be decided by the users and these two ranking options will be shown as "popular" and "relevant" on the search interface.

We will discuss how we rank results based on recommendation score later. For the relevance part, it is important to point out that there is no subjective definition for relevance. However, based on the TREC retrieval experiments, a ranking score call BM25 has shown promising results [14] and it is also the most popular ranking algorithm. We are going to implement our search engine with BM25 ranking algorithms.

Calculate Relevancy with Okapi BM25

BM25 is a bag-of-words similarity ranking function used in search engine. BM25 is defined as following:

$$\text{Score}(Q,d) = \sum_{q \in Q} \text{IDF}(q) \cdot \frac{tf(q,d) \cdot (k_1+1)}{tf(q,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|L_d|}{L_a vg}\right)} \tag{4.1}$$

where $tf(q,d)$ is term frequency for query q in document d and IDF is inverse document frequency. There are many variance of IDF, and the

most common one is denoted as $IDF(q) = \log \frac{N}{n_q} = -\log \frac{n_q}{N}$ where N is the total number of documents in the system and n_q is the number of documents that contains query q. k_1 and b are tuning parameter where k_1 usually is between 1.2 and 2 and b usually is 0.75. [15]. L_d is length of document d, and L_avg is average length of all documents

4.6 Recommender System for Poster Suggestion

Since we are bootstrapping a new poster platform without any prior knowledge about our users, recommender system performance would be evaluated using Movielens dataset from research group GroupLens. The assumption made is that a good recommendation algorithm can be extended to system with different items and user groups which then can a good fit for our poster system as well.

Our recommendation algorithm is based on collaborative filtering with matrix factorization technique which are both well studied methods that have shown promising results in many large scale recommender systems.

The recommender System problem states like this. Given N users and M items, the problem we are going to solve in collective filtering is filling the missing value in an incomplete matrix of ratings $\mathbf{R} \in \mathbb{R}^{\mathrm{NxM}}$

4.6.1 Conventional Approach

Cosine similarity

In recommender system, usually used Pearson correlation coefficient to measure similarity between documents. However in 2013, a research group grouplens found out that cosine similarity work better over mean-centered data instead. Because cosine similarity is easier to implements and under-

stand, in later chapter, when I refer to similarity, I mean cosine similarity and its definition is

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{4.2}$$

where A and B refers to vector representation of posters.

Content-based filtering

Content-based filtering is the most primitive and intuitive form of recommendation algorithm. The idea behind is if you like item A, I will recommend you another item B which is really similar to item A. This algorithm can be easier extended to its top-k version which recommended items are sorted by similarity scores.

Content-based filtering can be very flexible, since content of a poster includes poster name, extracted text, tagging, etc. When we use content-based filtering, we will build user profiles on items that they already rated. Based on these profiles, we recommend items that has feature which is most similar to their profiles.

Collaborative filtering

Collaborative filtering(CF) is a form of collective intelligence. Instead of making recommendation based on a single source, such as expert's opinion or a quality score based on item's content, we make predictions based on user preference with the assumption that the unintentional decision made by the crown is good or even better than experts' judgment.

There are two common approaches to perform collaborative filtering, they are neighborhood models and latent factor models. For the neighborhood-based approach, we have item-item collaborative filtering

and user-user collaborative filtering.

The formula for user-user CF is

$$P_{aj} = \hat{r_a} + \frac{\sum_{i \in NS_a} \text{sim(a,i)} * (r_{ij} - \hat{r_i})}{\sum_{i \in NS_a} |\text{sim(a,i)}|} \tag{4.3}$$

where Sim(a,i) is cosine similarity between user a and user i's rating profiles. NS_a is Nearest neighbor set of user a, and j is the poster item to be predicted.

The formula for item-item CF is

$$P_{aj} = \hat{r_j} + \frac{\sum_{i \in NS_j} \text{sim(i,j)} * (r_{ij} - \hat{r_i})}{\sum_{i \in NS_j} |\text{sim(i,j)}|} \tag{4.4}$$

where Sim(i,j) is cosine similarity between item i and item j's content. NS_j is Nearest neighbor set of item j, and j is the poster item to be predicted.

Latent Factor Models by Matrix Factorization

Collaborative filtering has two problems. First, it has poor performance if the item-user matrix is sparse which is also called the cold start problem. In many recommender system, most user would only rate a small fraction of the items. For example, I only rated a book «Hands-On Machine Learning with Scikit-Learn and TensorFlow» and another user rated «Deep Learning». We did not have at least one common rating in the item-user rating matrix. So mathematically, our similarity score is zero. Although it is clear that we share some interests. Collaborative filtering will not be able to find out unless we also provide book category information.

A technique called matrix factorization is introduced to recommender system trying to infer underlying latent factor model. A basic representation of the model is denoted as $\hat{r}_{\text{ui}} \approx {q_{\text{i}}}^{\intercal} p_{\text{u}}$ such that user-item interaction

is modeled as inner products of q_i, the degree item i process different feature factors, and p_u, the degree user u like those features.

Conventional SVD cannot be used to factorize an incomplete matrix $\hat{r}_{ui}$. One way to address this problem is by filling the user-item interaction matrix with default value such as average or median rating score. However this approach distorts data considerably and make prediction inaccurate. A better way to handle this problem is to use stochastic gradient descent to minimize regularized squared error and learn the optimal factor vectors p_u and q_i. Our goal is

$$\min_{\text{q,p}} \sum_{\text{u,i}\in\kappa} (r_{ui} - q_i^\intercal p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{4.5}$$

where κ is user item pair and r_{ui} is training set. [16]

4.6.2 Autoencoders for collaborative filtering

Beside conventional recommender systems algorithms such as neighborhood models or matrix factorization , we will also try a novel CF method that is based on neural network.

In 2015, [17] propose a collaborative filtering method called AutoRec that is based on autoencoders. We will train this model and test its performance in chapter 5.

The idea behind AutoRec is trying project $r^{(u)}$(user-based) or $r^{(i)}$(item-based) into a low-dimension hidden latent and trying to reconstruct $r^{(u)}$ or $r^{(i)}$. The loss function is denoted as

$$\min_{\theta} \sum_{r\in S} \|r - h(r;\theta)\|_2^2 \tag{4.6}$$

where $h(r;\theta) = f(\boldsymbol{W} \cdot g(\boldsymbol{V}\boldsymbol{r} + \boldsymbol{\mu} + \boldsymbol{b}))$ and f and g are activation function, $W \in \mathbb{R}^{d\times k}$, $V \in \mathbb{R}^{k\times d}$.

To prevent overfitting, it uses a regularise the learned parameter where regularisation strength $\lambda > 0$, the loss function then becomes equation 4.7.

$$\min_{\theta} \sum_{i=1}^{n} \left\| r^{(i)} - h(r^{(i)}; \theta) \right\|_O^2 + \frac{\lambda}{2} \cdot (\|\boldsymbol{W}\|_F^2 + \|\boldsymbol{V}\|_F^2) \tag{4.7}$$

The author also demonstrates the architecture of item-based AutoRec model. See figure 4.7

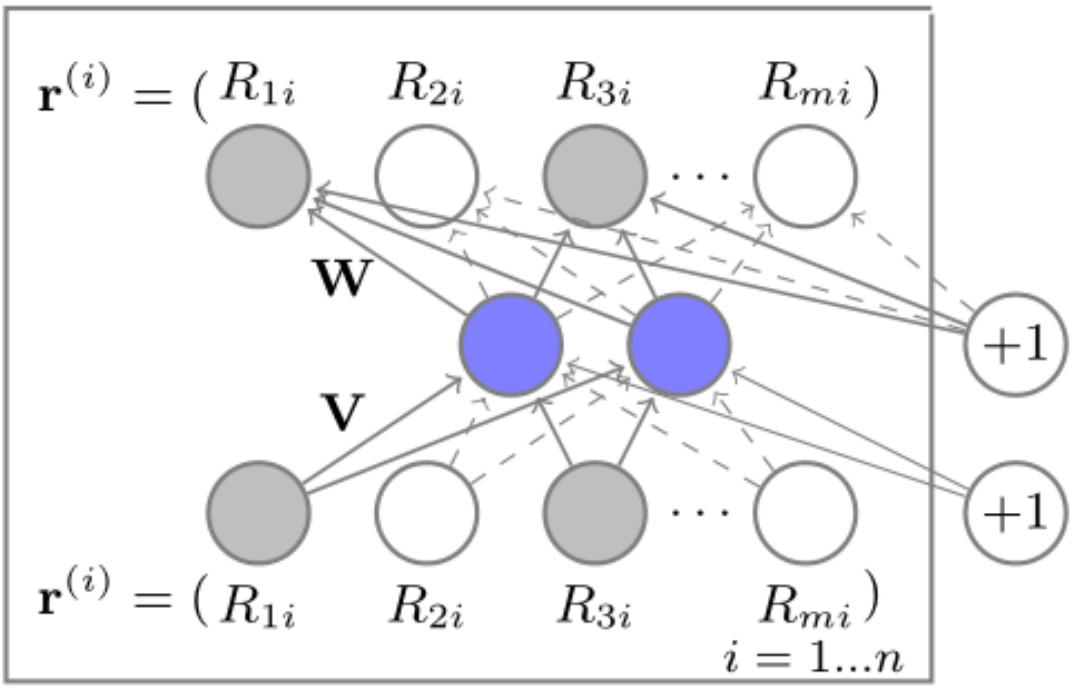

Figure 4.7: Item-based AutoRec model. $\boldsymbol{W}$ and $\boldsymbol{V}$ are fully connected.

The predicted rating is then calculated by

$$\hat{R}_{ui} = (h(r^i; \hat{\theta}))_u \tag{4.8}$$

4.7 Data Pipeline

The pipeline we used to provide data service is shown in figure 4.8. When users interact with the poster board system, interaction data is stored in the main database on online server. These data is then preprocessed into desired format and downloaded to local PC. Now we can analyses our dataset and try different machine learning algorithms on developer's machine. We can train those machine learning models manually or use a scheduler to do the training automatically and see the model performance report. If we think these model is good enough to launch to the production server. We synchronize the pre-trained model to the server. However the exact model format need to match with the working environment.
After the first batch of trained model is deployed, online system can hot load these models and compute query result using CPU in real time. User can call the corresponding API service and get back results in REST style.

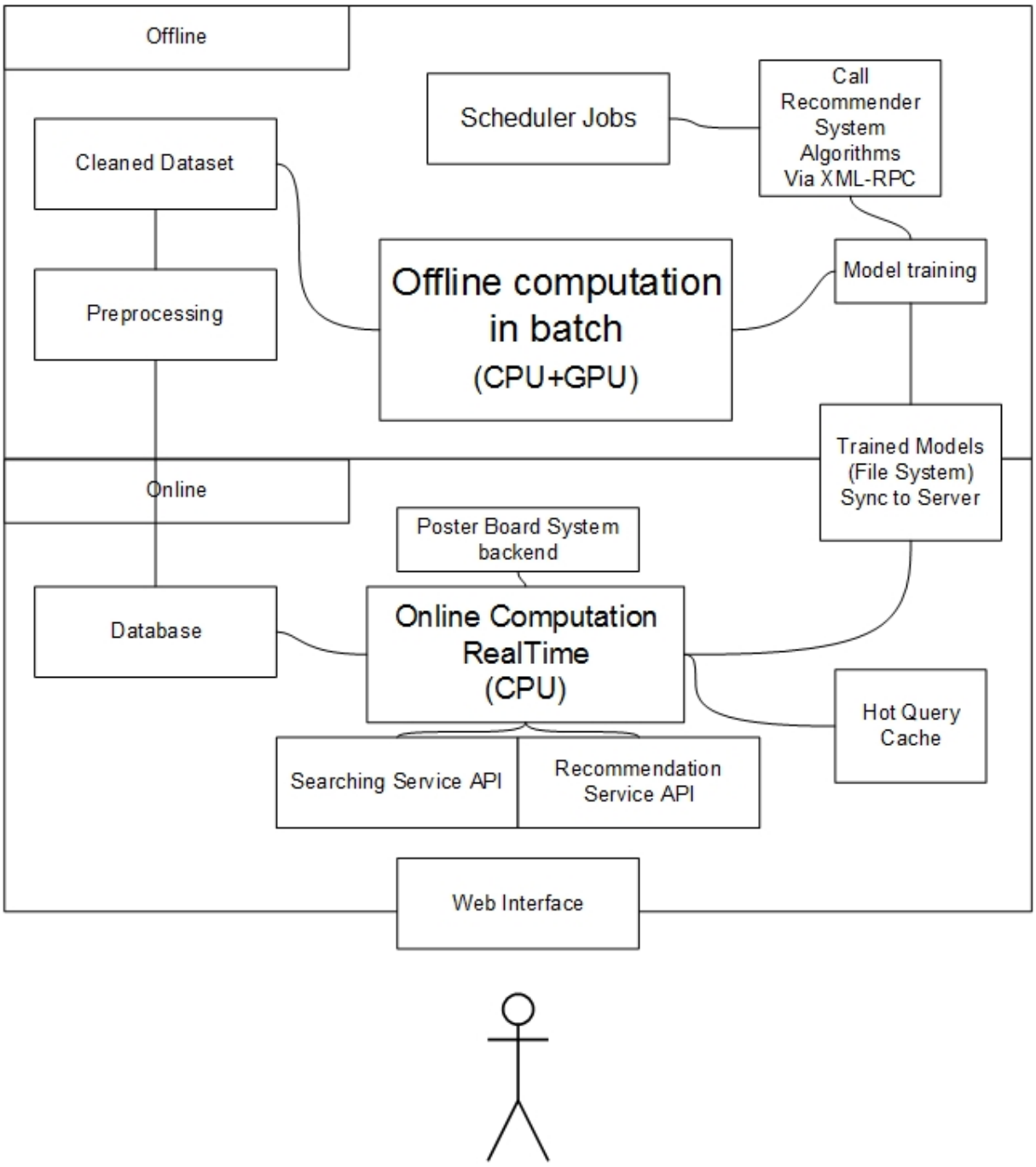

Figure 4.8: Data pipeline for recommender system and search engine module in poster board system

Chapter 5

Experiment Results

Our experimental evaluation has three parts. First, we will create a set of functional tests on the CMS system and test them with manual judgment. Second we evaluate our search engine with datasets provided by the University of Glasgow to see their mean average precision on real world cases. Lastly, we evaluate our recommender system using Movielens and Netflix dataset to test its recommendation quality.

5.1 System Functional Test

In this section, I am going to demonstrate several e-poster board's use cases in Table 5.1 and their expected outcome.

Table 5.1: List of functional test cases

ID	Test Case	Preparation	Expected outcome
1	Read poster with 3d model	Create new Poster database. Move 3d model teapot to website folder ./test/model/teapot.3ds. Upload test poster ./test/poster/3d.poster.	3D model correctly displayed.
2	Read poster with TeX math equation	Create new Poster database. Upload test poster ./test/poster/TeX.poster.	Equation correctly displayed.
3	Read poster with Embedded video	Create new Poster database. Upload test poster ./test/poster/Youtube.poster.	Video correctly displayed.
4	Open top 10 recommender list	Create new Poster database. Upload test poster ./test/poster/{1,2,3,4,5,6,7,8,9,10}.poster. Create new user 1. User 1 give 5 star to poster 1, 4 star to poster 2, 3 star to poster 3 , 2 star to poster 4, 1 star to poster 5.	Poster 1 ranks number 1 Poster 2 ranks number 2 Poster 3 ranks number 3 Poster 4 ranks number 4 Poster 5 ranks number 5
5	Enter ”development” on search engine	Create new Poster database. Upload test poster ./test/poster/{1,2,3,4,5,6,7,8,9,10}.poster.	Search results is Poster 2 and then Poster 1

Figure 5.1: Test Case 1. A posters with famous 3D model utah teapot created with blender is shown in red box.

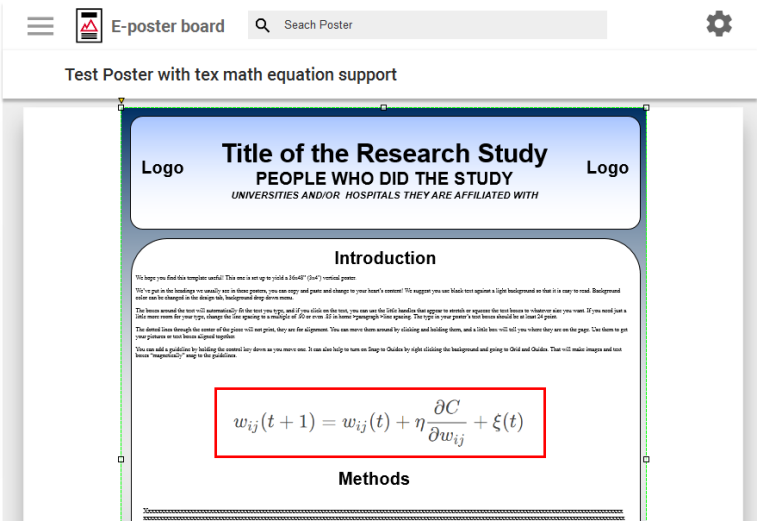

Figure 5.2: Test Case 2. A poster with equation written in TeX math mode shown in red box

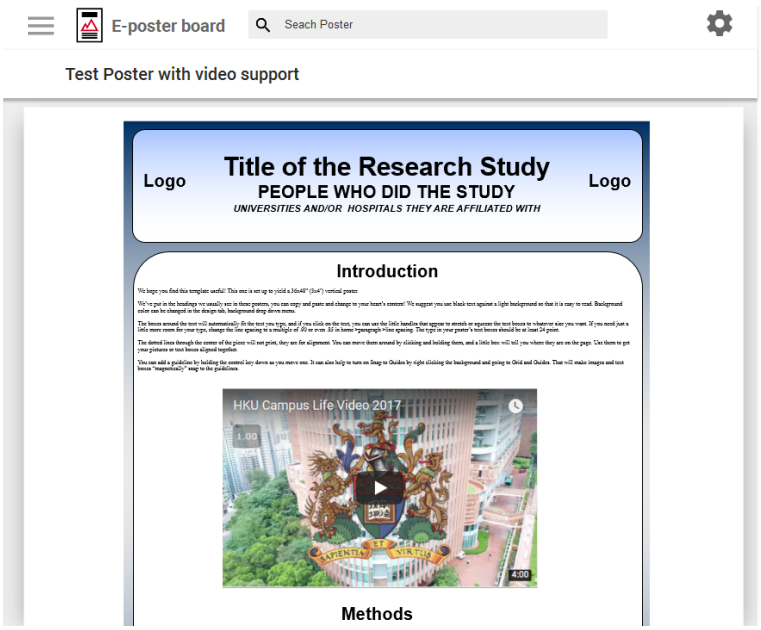

Figure 5.3: Test Case 3. A posters with embedded video from youtube. Video source is `https://www.youtube.com/watch?v=s8kMynR1rKg`

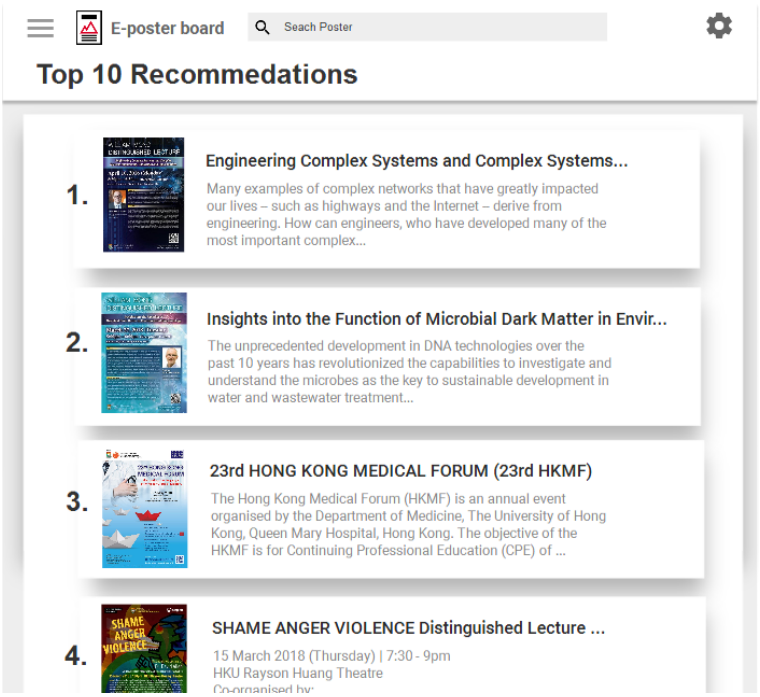

Figure 5.4: Test Case 4. Recommender Top 10 list Interface.

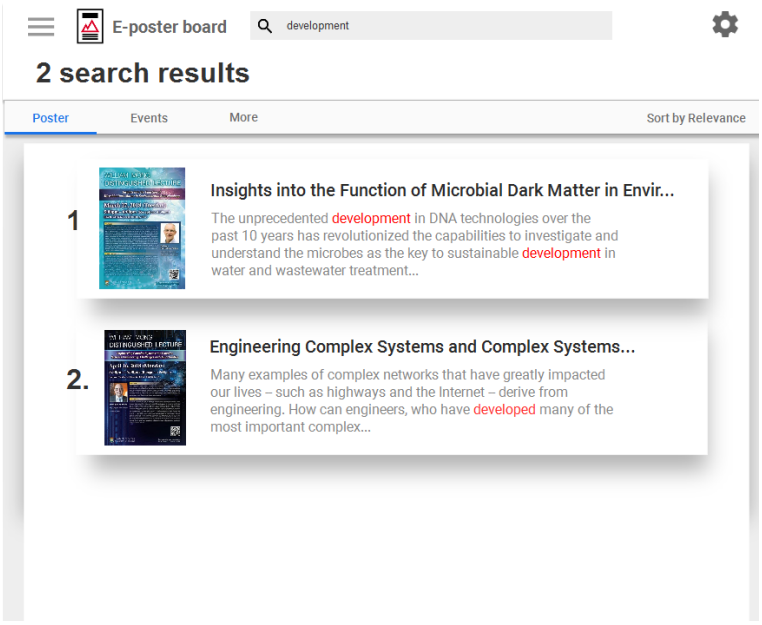

Figure 5.5: Test Case 5. The display interface for Search Engine with keyword "development". Red text indicated text matches with keyword.

5.2 Evaluation on Search Engine

5.2.1 Dataset and Evaluation Setup

We are going to conduct experiments on five TREC datasets disk1&2 TREC1-3, disk4&5 TREC 2004, WT2G TREC8, WT10G TREC9-10, and DOTGOV2 TREC2004-2006. Analysis of datasets without removing stopwords and other preprocessing steps is in Table 5.2.

Table 5.2: Analysis of Experiment Dataset

	# documents	total terms	unique terms	dataset size	Index construction time with pdf,Word extraction
disk1&2	741,856	307,973,251	950,711	0.7 GB	6 minutes
disk4&5	528,155	251,357,053	923,451	2 GB	12 minutes
WT2G	247,491	249,819,413	1,653,154	2 GB	18 minutes
WT10G	1,692,096	988,159,514	753,651	10 GB	1 hours
DOTGOV2	25,205,179	21,831,927,051	64,672,381	425.4 GB	23 hours

5.2.2 Evaluation Metrics

We are going to use mean average precision(MAP) to evaluate our search engine's performance. In TREC community, Mean Average Precision is considered the standard benchmark metric due to its good stability. Equation 5.2 is the definition of MAP.

$$Precision = \frac{|R_i|}{|P|} \tag{5.1}$$

$$MAP = \frac{1}{|Q|} \sum_{j \in Q} \frac{1}{|R|} \sum_{i=1}^{n} Precision(R_i) \tag{5.2}$$

where R_i is set of relevant documents retrieved, P is set of all retrieved documents, Q is test query set.

5.2.3 Evaluation Results

For each dataset, we perform evaluation with topical relevance using qrels provided NIST for all topics.

Table 5.3: Evaluation Results for our Search Engine by MAP@10 (higher is better).

	our Search Engine	Exact word classifier with id order
disk1&2	0.2226	0.0536
disk4&5	0.3015	0.0453
WT2G	0.3072	0.0323
WT10G	0.2021	0.0254
DOTGOV2	0.3111	0.0212

The experimental results are presented in Table 5.3, and we can see that our search engine performs much better than our baseline exact word classifier with no relevance score, which indicated that our ranking function is more than presenting documents to user at random order. Also, unlike the baseline which its MAP score declines with dataset size, our search engine implementation's MAP score is consistent to be between 0.2 to 0.3 for dataset with different sizes and documents numbers which means our search engine has good stability for large dataset.

5.3 Evaluation on Recommender System

5.3.1 Dataset and Evaluation Setup

In this experiment, we use three datasets Movielens-1M, Movielens-10M and Netflix. For algorithms that require dense user-item interaction matrix, missing entries are filled with global average rating. Cosine similarity is used in collaborative filtering algorithms that require to calculate the similarity between items or between users. We use 5-fold cross validation on the training set and take the average value of 5 test results. The analysis of our dataset is in table 5.4.

Table 5.4: Analysis of Experiment Dataset

Dataset	#Users	#Items	#rating	Sparsity
MovieLens-1M	6040	3706	1000209	95.80%
Movielens-10M	71567	10681	10000054	98.69%
Netflix	470758	4499	24058263	98.86%

5.3.2 Evaluation Metrics

$$\text{RMSE}=\sqrt{\sum_{t=1}^{n}\frac{(\hat{y}_t - y_t)^2}{n}} \tag{5.3}$$

To assess the prediction accuracy of algorithms in our recommender system, we use root-mean-square error which is defined in equation 5.3 where $\hat{y}_t$ is real rating value and y_t is predicted value.

Table 5.5: Comparison of Different Recommender System Algorithms in RMSE

	Movielens-1M	Movielens-10M	Netflix
GlobalAverage	1.117	1.060	1.130
user CF	1.036	0.986	1.092
item CF	0.876	0.851	1.209
SlopeOne	0.901	0.857	1.140
SVD	0.852	0.811	0.905
user AutoRec	0.874	0.867	0.901
item AutoRec	**0.832**	**0.785**	**0.834**

Table 5.6: Build time and running time of different algorithms. Build time includes I/O time and training time for AutoRec. Running Time is the total amount of time to predict all missing rating.

	avg Build Time(s)	avg Running Time(s)
GlobalAverage	0.421	1.492
user CF	0.422	28.209
item CF	5.04	17.552
SlopeOne	2.849	7.468
SVD	9.079	0.639
user AutoRec	3150.154	45.251
item AutoRec	3135.584	45.121

5.3.3 Evaluation Results

In table 5.5 and 5.6, we see that item CF performs really well in both movielens-1M and movielens-10M dataset but it has worst RMSE in netflix dataset. The neural network based item AutoRec achieves the best result in all three datasets. However it took nearly 52 minutes to train the model and 15 second to run. This can be a huge computation burden for the server to provide recommendation in real time and it is hard to scale. As a trade-off between resource demand and prediction performance SVD seems to be the best choice for our system. Since SVD took reasonable amount of time to train, and the time to make prediction is minimal across other 6 algorithms we have.

Chapter 6

Difficulties and Limitations

6.1 Performance issue on complex poster items

Parsing a huge svg vector file in plain text form can be time consuming. The reason why svg is bloat and of large size usually would be technical decision in pdf to svg converting step doesn't meet our needs. The top priority of pdf to svg converter are to parse all detail of pdf contents and translate them in svg standard. In this process, many unnecessary information like unnecessary attributes, invisible elements and machine generated comments make the svg file very bloat and therefore it is slow to read and process. Since the svg won't be modified or analyze extensively by the system, there is no need to preserve these hidden information. It is safe to remove these unnecessary contents in the converting step.

Also, another preventive method would be setting restrictive policies for file of maximum number of elements or largest file size allowed to be uploaded before processing to the parsing step.

6.2 Performance issue on recommendation system

Due to the information-intensive nature of e-poster file, ranking operation is inflexible to perform using the request-and-response method in real time.

Hence, an engineer decision is made to reduce system load by utilizing in-memory caching . After the initial calculating, recommendation results could be stored in in-memory key-value store and later these information can be retrieved from the store to accelerate the whole process. The calculation will only run once an hour to keep ranking information updated, and this simple engineering tactics avoid repeating expensive calculation steps and reduce response time on the terminal end. Although ranking result may not be up to date in real time, it is not a major issue as poster submission and modification are expected to happen in a relatively steady and moderate rate. A good ranking algorithm would not have huge variation in output result when input sources have little change.

Chapter 7

Conclusion

To help community members find out events and notices they interested in, this paper describe a possible implementation of e-poster board system. The system uses a multi-tier architecture and responsibilities of each tier are separated to reduce programming complexity. This report has contributions in three areas. First, a suitable software architecture for e-poster system is designed, and it is practical for this architecture to handle specific needs in providing interactive functionalities. Second, it introduces a new poster interaction format to enhance user experience. Third, this report provides solutions for technical problems such as poor scalability of complex poster file. Third, this report introduces some computer techniques such as recommender system and search engine that can be used to further improve user experience in using e-poster board.

There are several directions for future development and research on improving e-poster board system. One direction would be reducing set-up environment constraint without sacrificing usability. Currently, this poster system require a terminal device with internet and interactive display support. This may seem acceptable in in-door scenario, but it may not be flexible to meet these setup requirement in out-door space where wireless connection and display protection may not be present. A hardware equipment setup with better out-door space support should be found when

internet-of-thing(IoT) devices become more prevalent. Another direction will be developing better toolset for creating svg file with HTML elements. As svg file has good compatibility with PDF format and the web platform, it is not hard to predict that it will be widely used for displaying complicated document file with interaction functionality in near future. Also, a mature workflow both to create and publish web-argumented svg file will be in need to accelerate adaption from paper poster board to online poster platform.

Bibliography

[1] E. F. Churchill, L. Nelson, and L. Denoue, "Multimedia fliers: Information sharing with digital community bulletin boards," in *Communities and technologies*, pp. 97–117, Springer, 2003.

[2] J. Colegrove, "The state of the touch-screen market in 2010," *Information Display*, vol. 26, no. 3, pp. 22–24, 2010.

[3] "Digital posters — electronic signs." `http://www.digitalmediasystems.co.uk/digital-posters/`. Accessed: 2017-09-30.

[4] "Interactive electronic posters for multimedia presentations." `https://ipostersessions.com/`. Accessed: 2017-09-30.

[5] "eposterboards." `http://www.eposterboards.com/conference-setups/`. Accessed: 2017-09-30.

[6] C. Shirky, "It's not information overload. it's filter failure," *Web. September*, 2008.

[7] "material design, introduction to principles." `https://material.io/guidelines/material-design/introduction.html#introduction-principles`. Accessed: 2018-01-12.

[8] "The final countdown for npapi." `https://blog.chromium.org/2014/11/the-final-countdown-for-npapi.html`. Accessed: 2017-10-22.

[9] "Netscape-style plug-ins do not work after upgrading internet explorer." https://support.microsoft.com/en-us/help/303401/netscape-style-plug-ins-do-not-work-after-upgrading-internet-explorer. Accessed: 2017-10-22.

[10] "Version 52.0, first offered to release channel users on march 7, 2017." https://www.mozilla.org/en-US/firefox/52.0/releasenotes/#changed. Accessed: 2017-10-22.

[11] V. Pimentel and B. G. Nickerson, "Communicating and displaying real-time data with websocket," *IEEE Internet Computing*, vol. 16, no. 4, pp. 45–53, 2012.

[12] W. B. Croft, D. Metzler, and T. Strohman, *Search engines: Information retrieval in practice*, vol. 283. Addison-Wesley Reading, 2010.

[13] J. Zobel, A. Moffat, and K. Ramamohanarao, "Inverted files versus signature files for text indexing," *ACM Transactions on Database Systems (TODS)*, vol. 23, no. 4, pp. 453–490, 1998.

[14] E. M. Voorhees, D. K. Harman, *et al.*, *TREC: Experiment and evaluation in information retrieval*, vol. 1. MIT press Cambridge, 2005.

[15] R. R. Larson, "Introduction to information retrieval," 2010.

[16] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, 2009.

[17] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," pp. 111–112, 2015.