Online Teacher Training ToolTechnical Design Document

# 1. Tech Stack

The online Teacher Training Tool project is basically a video website with different special rating rubrics toward content of videos. The whole website can be divided into the following parts:

* 1. Backend Part

We chose java as the website’s server-side language. More specifically, we use Spring Boot as backend framework. We will use MySQL as database which will store users’ information, users’ evaluations toward video content, videos’ address and videos’ metadata. Also, the website will include a search engine that can help users find the desired content. Search engine can be implemented by open-source tools like Solr or Elasticsearch or just one another MySQL table.

* 1. Media Server

We need a media server to store and provide access to video files. Although the backend server and the media server can be the same server, the media server part has jump out of the Spring Boot frame, and is worth to mention what kind of technology it needs specifically. To build a media server, we need Nginx to provide RESTful end-point for every video. Website users should be able to upload their own videos to the website. We might also need FFmpeg to transfer users’ uploaded files in order to keep all videos in the same format (mp4 file).

* 1. Front end Part

The frontend part is easy, it usually contains HTML, JavaScript, and CSS. In order to provide high quality user experience, we might need to use HTML5 video player like Plyr. Also If need, we will use React.js to design user interface.

* 1. Tool list

Spring Boot version: 2.1.3. This is the server-side program used for handle all user requests

Java version: 1.8.

Maven version: 3.6.1. It is used for manage Spring Boot program.

MySQL. It provides database for the whole system.

HTML5. It is used for design user interface.

Nginx. Help media server to provide HTTP link for every video.

Solr(might use). It can be used as a search engine.

# 2. Accounts and Infrastructure

## 2.1 Development

We will develop the application on our own laptop. Because it contains media server, after building the media server, we might need to test it by deploying it on AWS. All of our team members has AWS account, and we all have student credit on our accounts. The machine name for developers is : Yongde Cui’s MacBook Pro, Yi Jin’s MacBook Pro, Ying Gong’s MacBook Pro.

## 2.2 Production

After the development is completed, the application might run on department’s server. By now, we do not know the exact production environment.

# 3.Data Sources, Models, Timing

## 3.1 Data Sources

Some lecture videos will be uploaded before the system is put into production. So MySQL needs to be imported some video information at the beginning. Administer do not need to configure initial data. Users’ evaluations and video will accumulate as users start to use the website, and they are both user-entered data. User evaluation will be stored in MySQL database, video will be stored in the media server.

## 3.2 Data Models and Structure

3.2.1 Data visualization

In order to understand the data model better, we should visualize some interfaces and rating rubrics.

The first user interface is user login page. And we also need search page, and evaluation page. Because we have two different evaluation models, we need two kinds of evaluation pages—analysis teaching page and four-part rubric page.

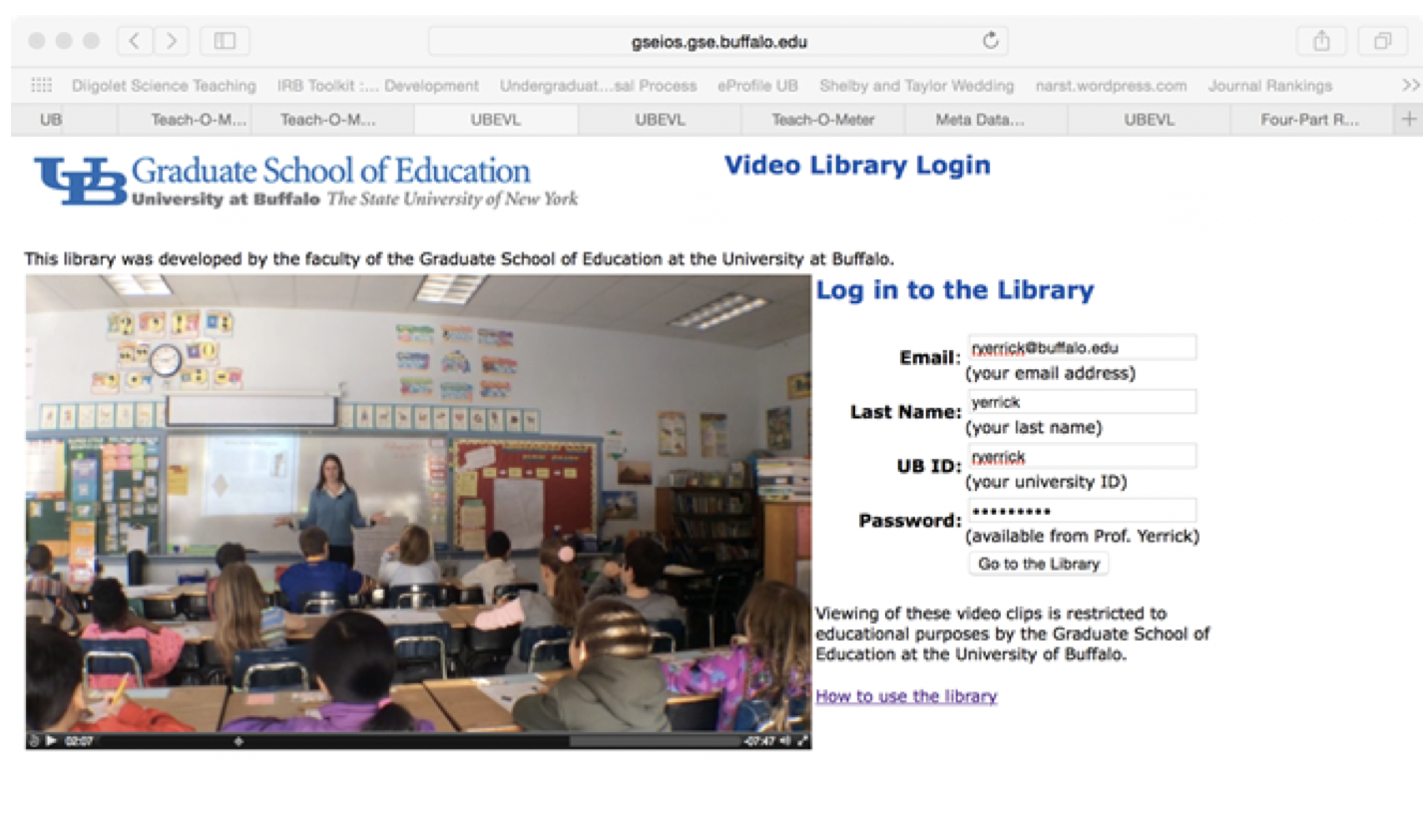


Figure1. Login page

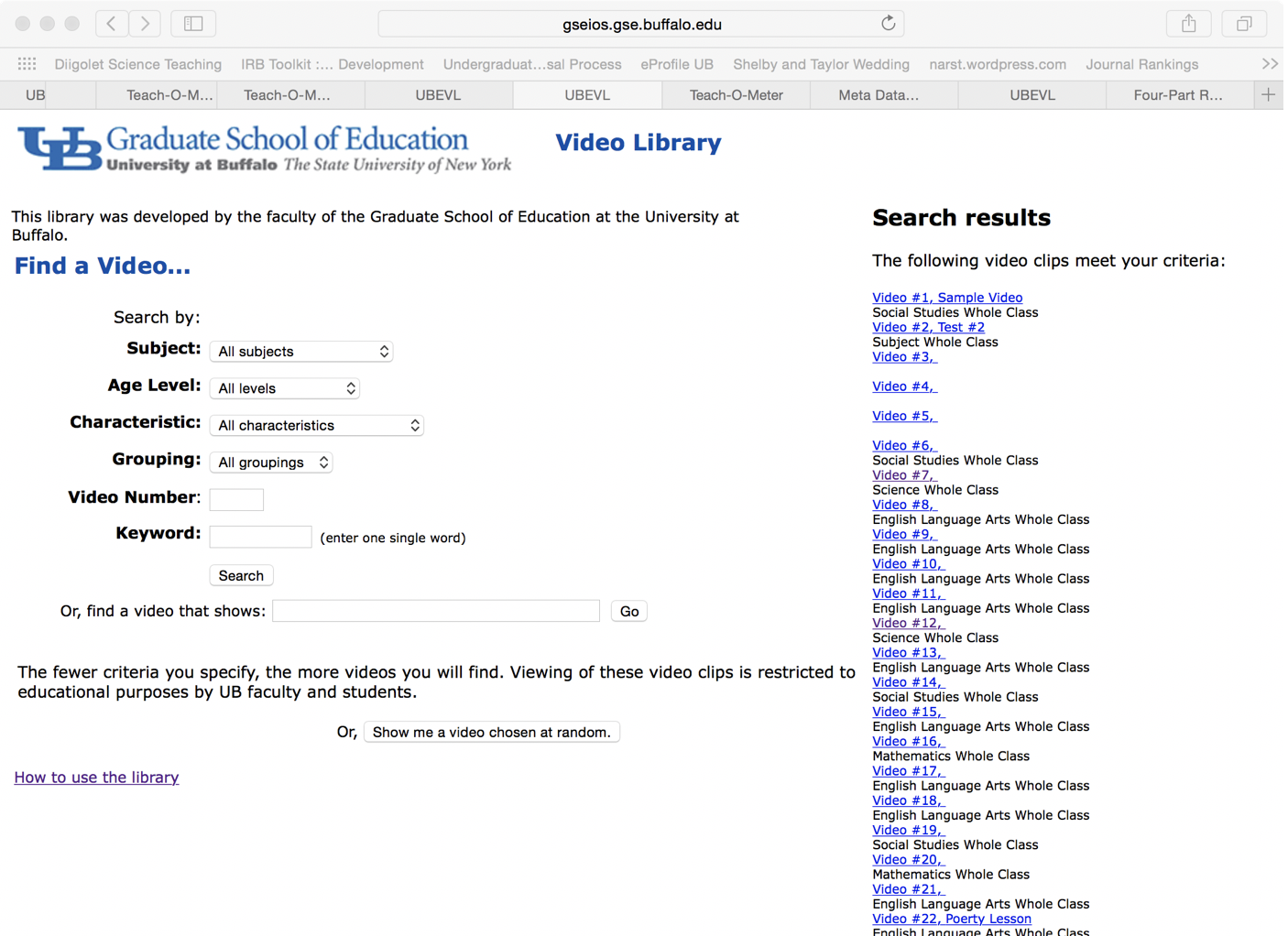


Figure2. Search page

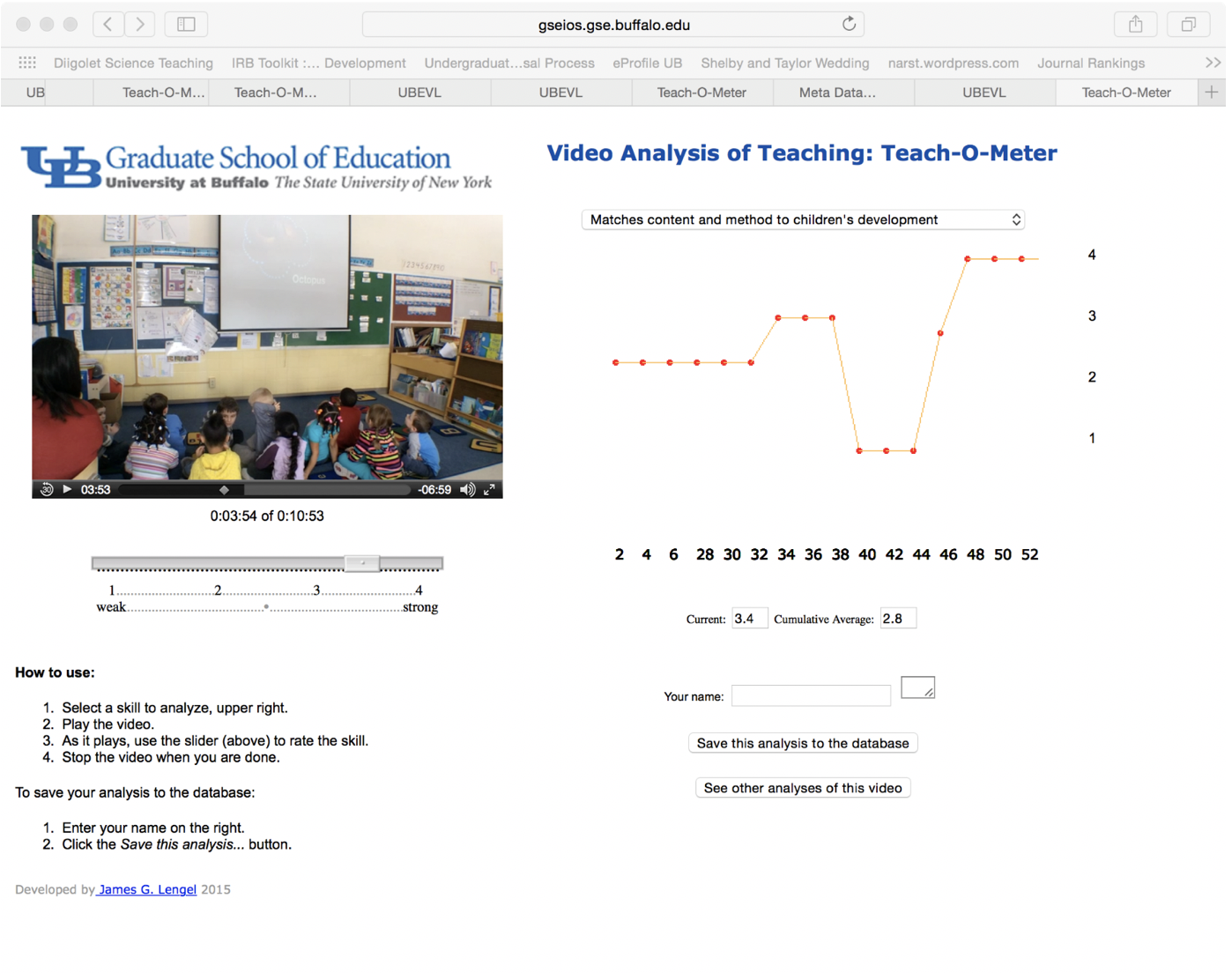


Figure3. Evaluation page: analysis of teaching model

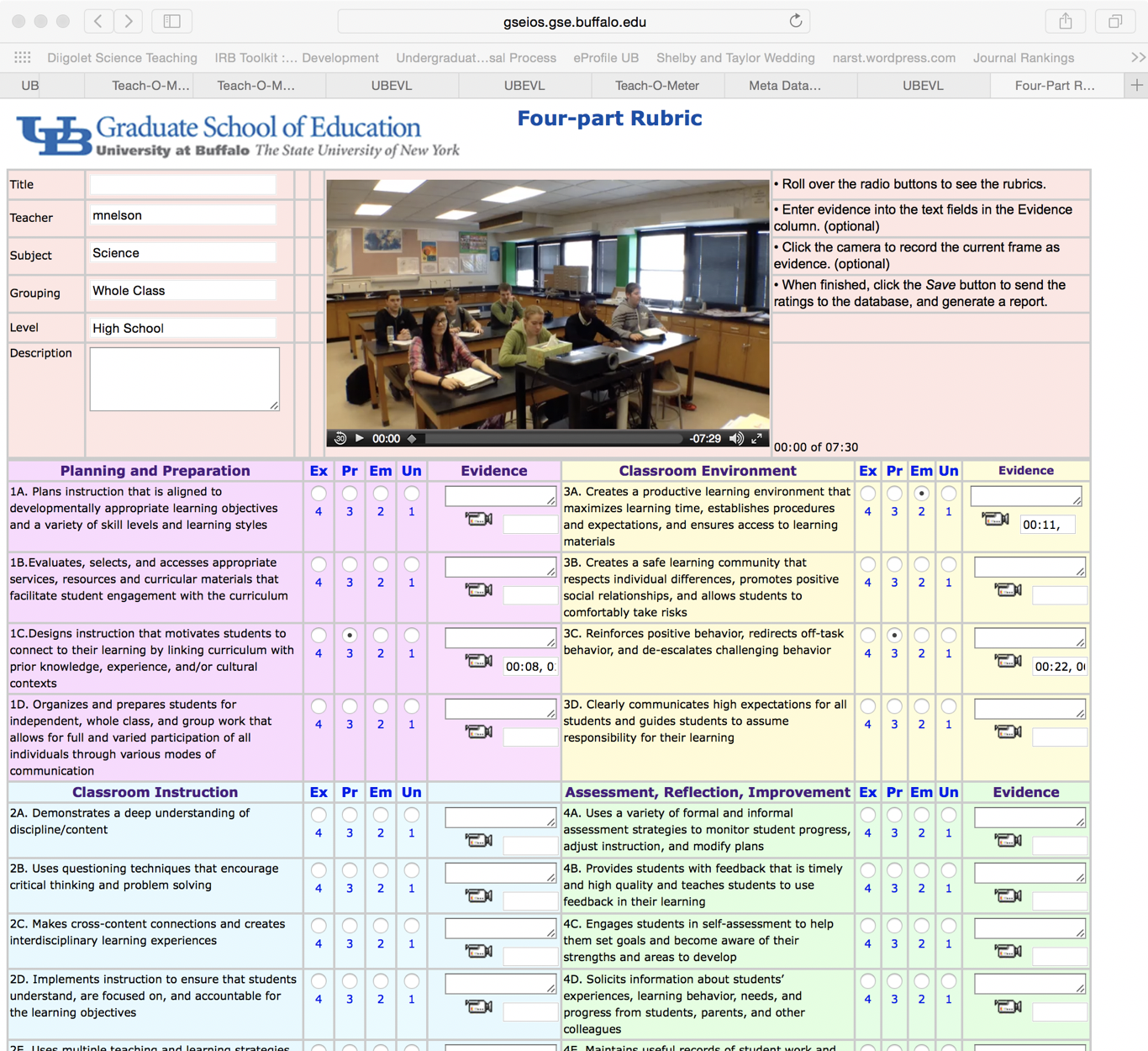


Figure4. Evaluation page: four-part rubric model

It is worth to mention that in the analysis of teaching model, the evaluation can be regarding to many attributes like “communication skill”, or “get feedback from students”. The score for one attribute can change as the video plays forward. Thus we might need a long string to document the scores change as time goes by.

3.2.2 SQL table design

Now, let’s think about the concrete data model. All the user should login before accessing the lecture videos, thus we should first build a user table to store users’ information. This table is mainly used for user authentication. For that purpose, the fields of this table should be:

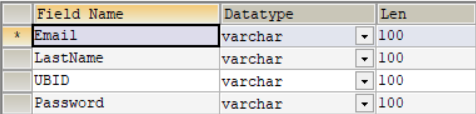


Figure5. User table

The second interface is search interface. In this page, user enter keywords to search for related content. So we need a video table for storing every video’s meta data and access address. For that purpose, the fields of this table should be:

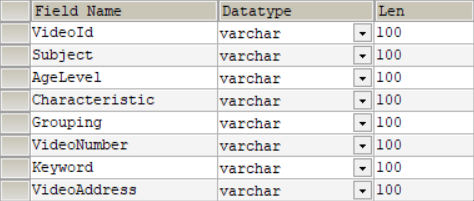


Figure6. Video table

Next, we need two tables to store information for the two kinds of evaluation model. We can build one rating analysis table for the first evaluating model. And a rubric table for four-rubric evaluating model. In each table, the UserId means the user who make the evaluation to the VideoID.

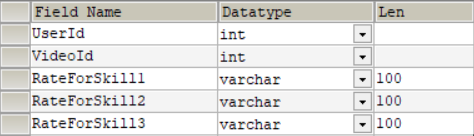


Figure7. Rating analysis table

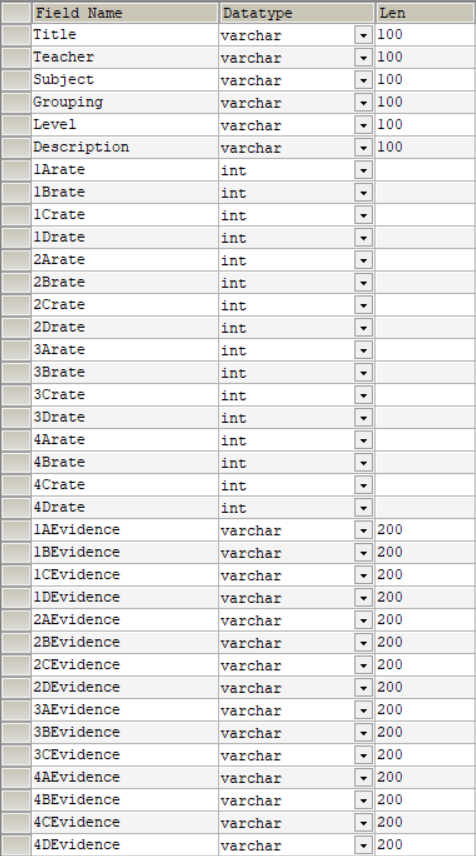


Figure8. Four-rubric table

Every field is a column in the SQL table. Our sponsor has a requirement that the system should allow administer to add new rating attributes regard to the video content. And this is the reason why we need set special method in java to allow administer add new columns to the evaluation tables.

3.2.3 Entity relationship diagram



Figure 9. A brief entity relationship diagram

## 3.3 Timing

The data in the application can be user’s uploaded videos, and user’s evaluation toward the video content. User can include new video to the database by uploading it, and add new evaluation by completing the specific rubric. Once user add a new video or a new evaluation to the database, the data should live forever unless the user chose to delete it. Also the system should not archive deleted content.

# 

# System Architecture Diagram

The system architecture can be shown as below:

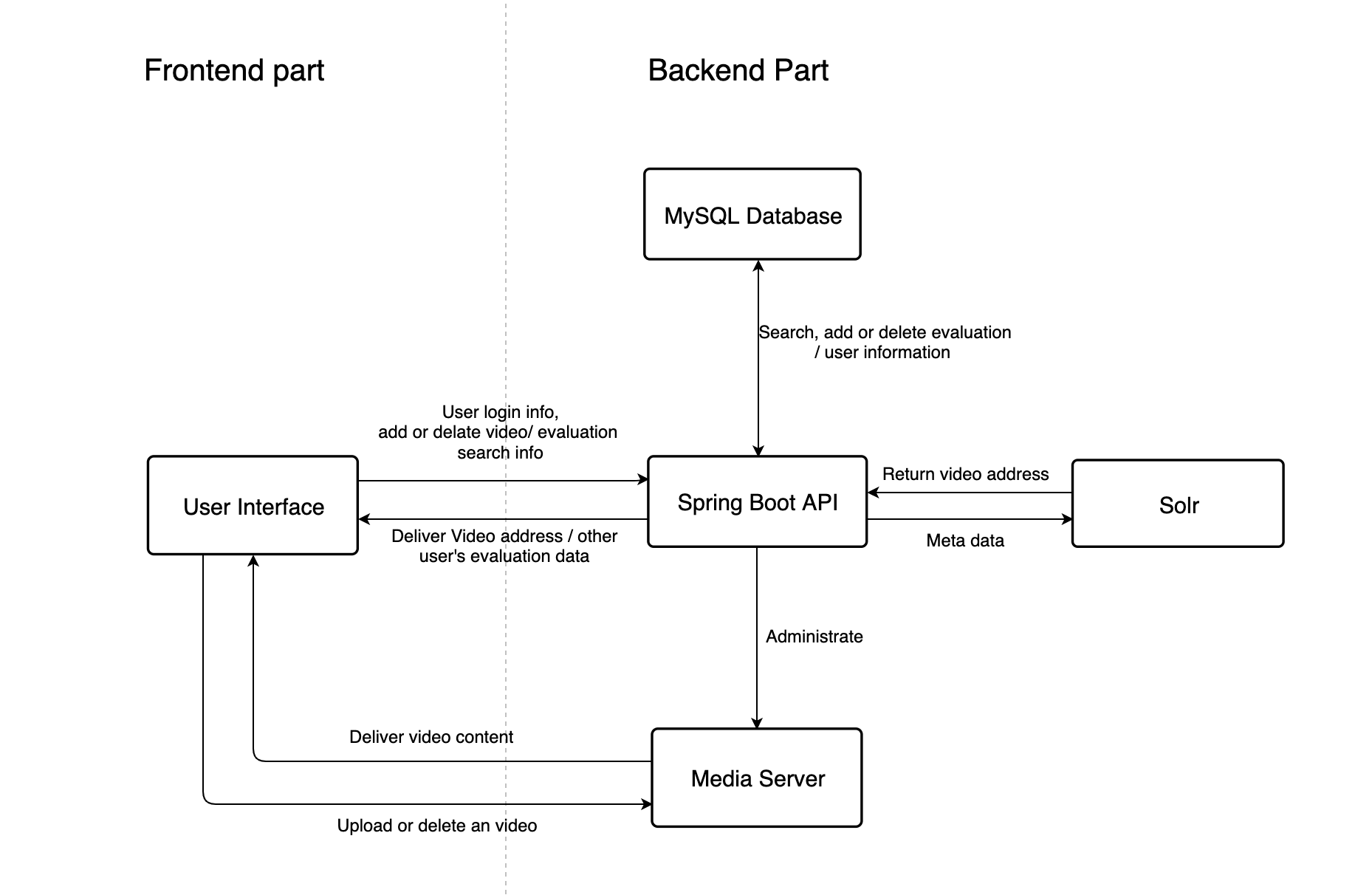


Figure 1. System architecture diagram

User interface includes the following part:

1. Login-page (including HTML, JavaScript and CSS) used for authentication.
2. Uploading-videos page (including HTML, JavaScript and CSS) used for user upload videos to media server.
3. Watching-video page (including HTML, JavaScript and CSS) used for users watching chosen video.
4. Searching pages (Including HTML, JavaScript and CSS): a search engine, used for finding desired content in Media Server, this part should include one function to show searching result from Spring Boot searching handler (after clicking specific searching result, user interface will go to watching-video page).
5. Evaluating pages (Including HTML, JavaScript and CSS) used for users comment and rate for a video.

Spring Boot API part includes the following part:

1. Start-page handler: responds to users’ initial request, give users login interface for authentication
2. Login-checking handler – checks the information users entered on the login page. If it is correct, give users permission to enter the system (give users searching interface). Otherwise, keep users from entering the system. This handler works between the login page and the User table (MySQL).
3. Uploading-video handler: handles uploading requests from users (who are authenticated), put the users’ video into the Media server, add video information into the Video table (MySQL), give user status feedback (successfully uploaded or not). This handler works between uploading-video page, Media server and the Video table (Or Solr index engine).
4. Watching-video handler, handle the requirement from searching result page. Giving users access addresses to their wanted videos.
5. Searching-video hander, handles searching requests from searching video page. Search video in Media server based on users’ desired (keywords entered on the searching interface). Return user the searching result (Watching video interface). This handler works between searching page and Video table (or Solr).
6. Evaluation hander, handles users’ evaluation toward specific videos. This handler works between evaluation pages and two evaluation tables.

Media server including:

1. Providing stream access to all the videos. Users can access all the video stream by HTTP links. This function is used for watching video pages.
2. Store new videos uploaded by users. This function is working with Spring Boot Uploading video hander.

Solr (may) includes:

1. All videos information. Interact with Spring Boot searching-video handler.

MySQL includes:

1. User table, stores users’ information, used for authentication. This table interacts with the Login-checking handler.
2. Video table, stores videos’ information (like metadata or access address). This table will interact with Spring Boot Uploading-video handler and Searching-video hander.
3. Two evaluation tables (Rating analysis table and Four-rubric table). These two tables store users’ evaluations toward all the videos. These two tables interact with the Spring Boot Evaluation hander.

# Deployment Methodology

The System will be put on a server. When we turn the system, we should give sponsor an administer account, and a complete backup source code. The system should originally contain some uploaded videos given by Professor Yerrick, and new video and evaluations will be added by real users after the system is deployed.