Team Northwestern: VisCrit

Web-based critiquing platform

Team

- Long Huynh, Long.Huynh001@umb.edu
- Jeffrey Deng, Jeffrey.Deng001@umb.edu
- Thamany Valbrune, Thamany.Valbrune001@umb.edu
- Christos Kakouros, Christos.Kakouros001@umb.edu

Client

- Steven Franconeri from Northwestern University

Final Documentation - 5/19/2023

Repository Link
https://github.com/longy2k/VisCrit

1 Introduction

1.1 Purpose

VisCrit is a web based application that aims to provide a platform in which students can critique other peers' work. The instructor creates a rubric that is passed to the students through VisCrit. This web application allows students to upload a dashboard, either in PDF or image format provided by the instructor. The students are able to annotate and rate different parts of the dashboard. After critiquing is done students are allowed to save the critiques that were made, and instructors also receive a copy of the critiques that were made.

1.2 Existing Works

- The website Hypothesis enables the ability to annotate the web, with anyone, anywhere.

1.3 Scope

VisCrit is a web based application that provides a critiquing platform for both students and instructors to use. This tool aims to provide an easy to use intuitive interface to critique uploaded PDFs/slides. Since this is a web based critiquing platform no further software has to be downloaded for the site to be used, all that is needed is an internet connection.

1.4 Limitations

- A user must follow the layout for the excel file for the rubric so it can be compatible with VisCrit
- Only one image at a time can be uploaded if a user chooses to upload an image.
- Depending on the size of the image, the quality could look off. Images that fit the 800(W)x500(H) or similar size, and are high resolution look the best.

1.5 Definitions

- **CSS** - Used to edit the styling components of the website
- **HTML** - Markup language that was used to make the page layout
- **ReactJS**- Used to create a responsive and interactive user interface.
- **JSON** - Used for storing necessary data for the rubric and PDF
- **ExpressJS -** Used for back-end framework to provide necessary functionality for handling user requests, managing data and communicating with the front-end.

2 Requirements

- Allow an instructor to upload an excel file or manually upload visuals with the rubric
- Generate a User-View page using information from the data-base or fill in manually.
- Creating an XLSX file from the students inputs and allowing it to be downloaded.

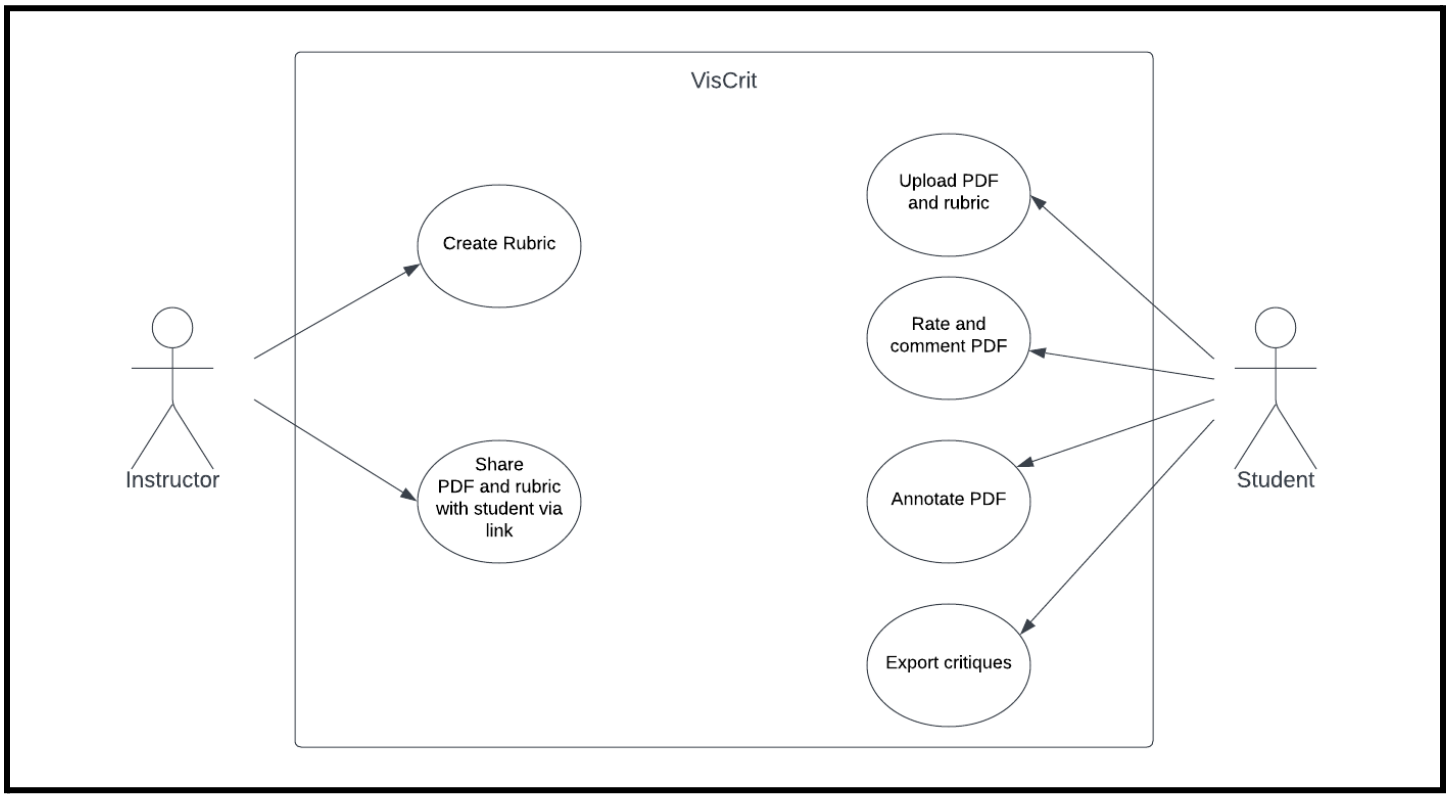
3 Specifications

- VisCrit is a web-based software that requires no extra software to be downloaded to be used. On the main page of the website, there is an upload button where a user can upload a slide in PDF or image format to critique along with an excel sheet (considered the rubric).
- The user is able to rate something based on the categories and subcategories in the rubric, there is a scale from one to five (1 being good 5 being bad).
- A user must also add a comment for the reasoning of the rating.
- A user can also mark up the slide with the highlight box and add text. When a user is done with the critique, the user can export their critiques so both the instructor and the student can have a copy.

4 Design

4.1 Use Cases

The audience for this web-based project are students and instructors specifically at Northwestern University who want to easily critique PDFs. The instructor will be able to provide the rubric and dashboard with students which they upload to the site to begin their critiques. Students are able to rate the PDF on a scale of 1 to 5, annotate areas of the pdf, add comments on reasoning for the rating, and finally when all critiques are complete they can export their file into a CSV file so the student and instructor can have copies of the critiques.



4.2 Architecture

CritiqueBox.js - Generates the critique box to allow users to comment.

Data_Extract.js - Takes in JSON data and parses it to create and return a list of Hierarchy objects.

DocumentReader.js - Loads image or PDF from server onto UserPage. Creates a canvas on the file to allow for highlighting and stores it as an array. The array can then be stored in the Item object upon saving.

Dropdown_Gen.js - Contains the Hierarchy class. Hierarchy contains a list of Item objects which would be the comment items. It also contains a list of sub Hierarchies stored in a map.

ExportItem.js - a class for ExportItems only, expects an Item class object and the index of that item that is being modified in the critique box. This class is used for exporting only.

Item.js - class of Item where all modifications of a specific Item is stored. LocationRt stores an array with the first element being the page number followed by the array of rectangles. Comment stores the String comment, each index represents the like value.

ItemContext.js - Stores all variables/ hooks to be used in all the other ReactJS files. Essentially allows for global variables.

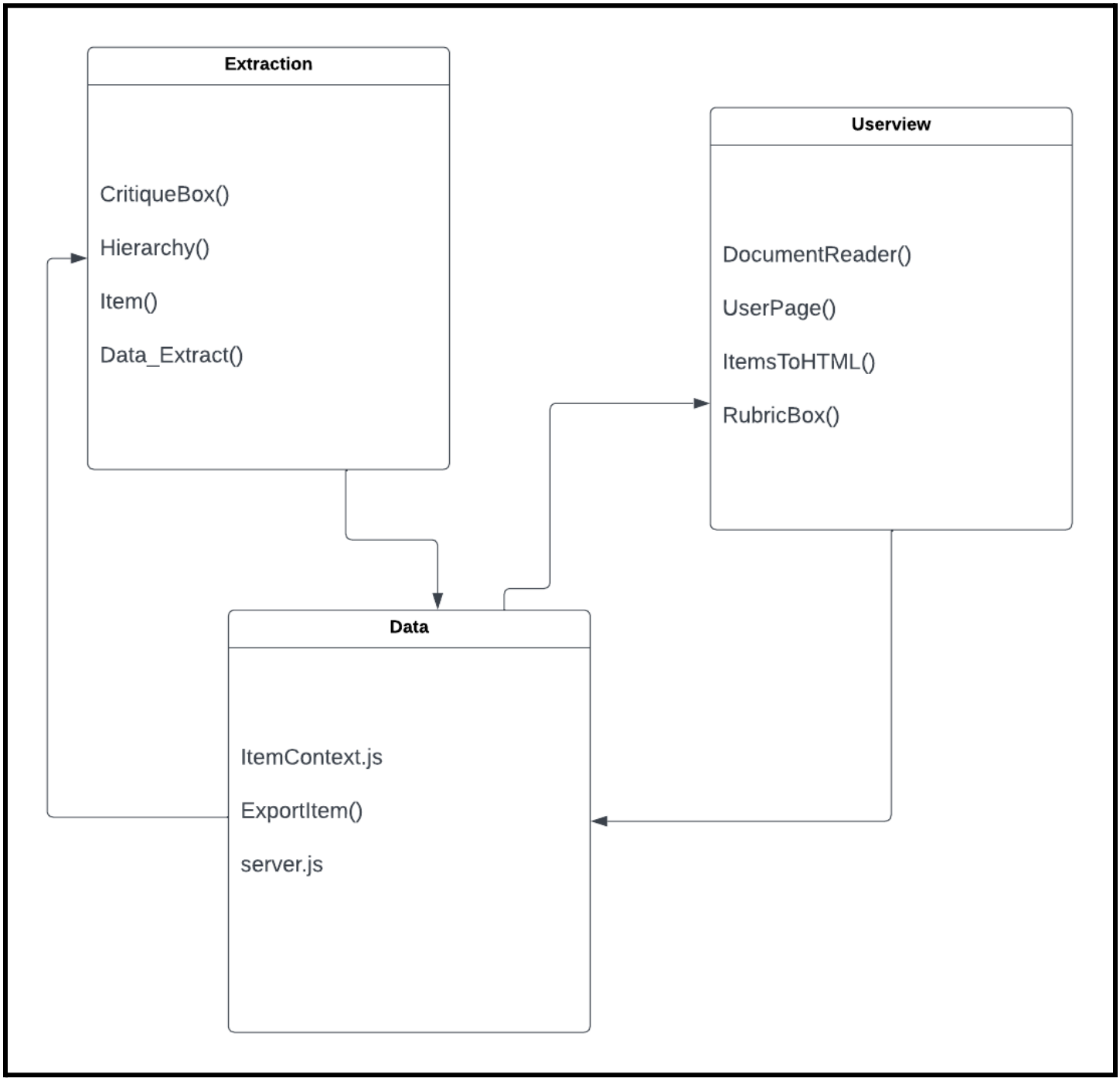
ItemsToHTML.js - Renders each individual item onto the page. Used in Dropdown_Gen.js to render the Hierarchy to page.

RubricBox.js - Renders each Hierarchy item from the Hierarchy list which was generated by Data_Extract.js

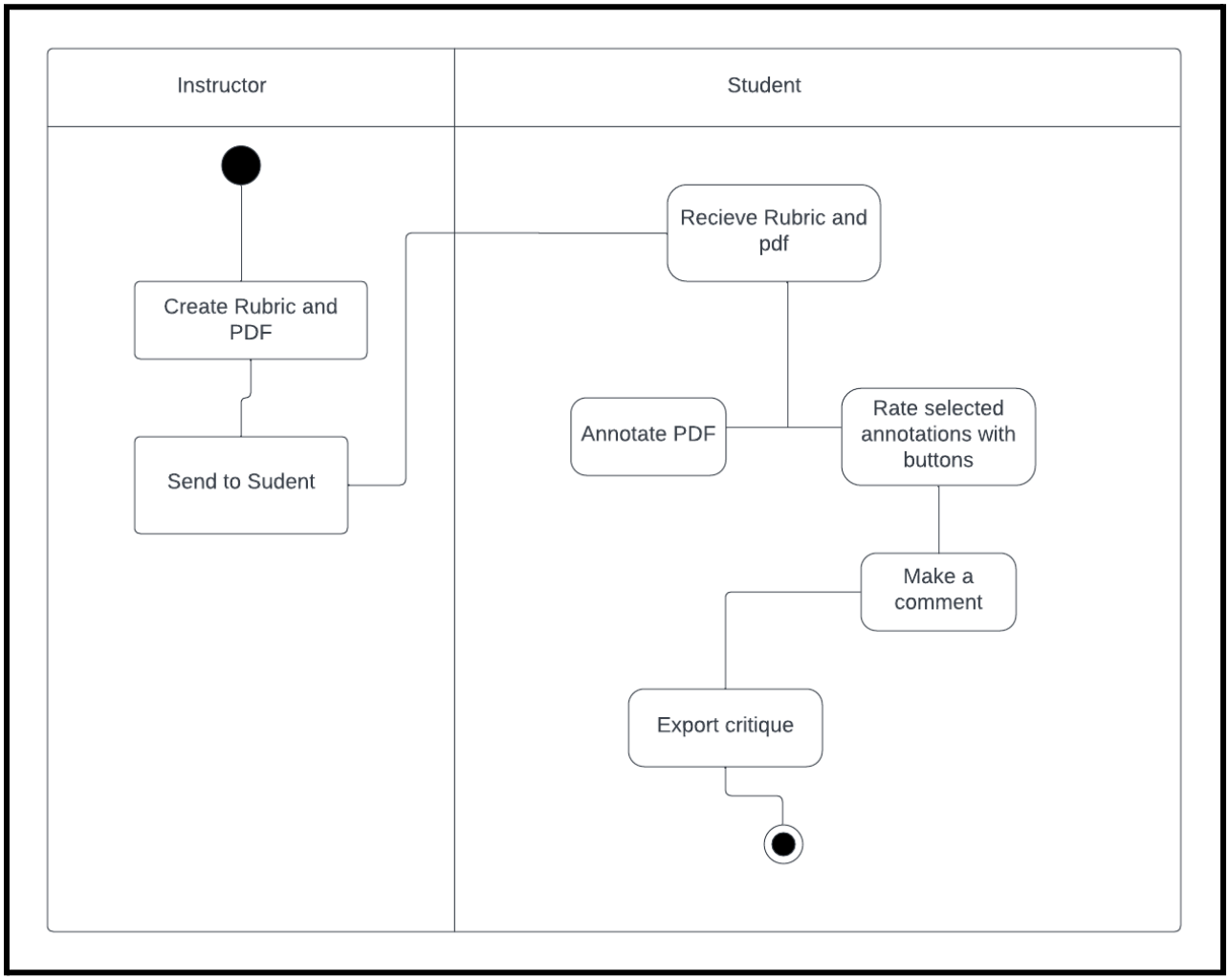
UserPage.js - Loads JSON data from server for rubric. Utilize Data_Extract.js to turn JSON data to list of Hierarchies and store it. Generates global variables with useContext for other files to use. Renders DocumentReader and RubricBox if there is data in the server, otherwise defaults to an upload page to store data onto the server.

Guide.js - Loads a guide onto the landing page.

Server.js - Allows for storing and accessing files from the server.


Extraction
CritiqueBox()
Hierarchy()
Item()
Data_Extract()
Userview
DocumentReader()
UserPage()
ItemsToHTML()
RubricBox()
Data
ItemContext.js
ExportItem()
server.js

4.3 Workflows



4.4 Technologies and Implementation Details

For our front-end, we used ReactJS and Javascript, our back-end was created by using ExpressJS, which was used for storing the uploaded files. For styling, CSS was used greatly to create a clean interface. For our PDF annotations we used a package called PDF.js that allowed us to render a PDF, then afterwards applied a canvas over it to give it its annotation functionality. We also made an excel workbook extractor for the uploaded excel sheet (the rubric), it displays the contents of the rubric.

4.5 User Interface

The user interface starts with a page that allows the user to upload both their PDF and their XLSX file that contains their rubric. Also on the main page, there is a tutorial that explains how to use the application. After users upload their files, they're redirected to a page that displays the PDF/image and along with the rubric split. This page is where annotations, comments, and ratings can be made, there are several categories and subcategories that can be collapsed and re-opened for the rubric. Pages of the PDF can be changed by pressing the left and right arrow button. Once all

comments and annotations are completed users are able to select the export button to download their responses in a CSV file.

Figure 1: This is the main page of our site where the user is prompted to upload their PDF and rubric

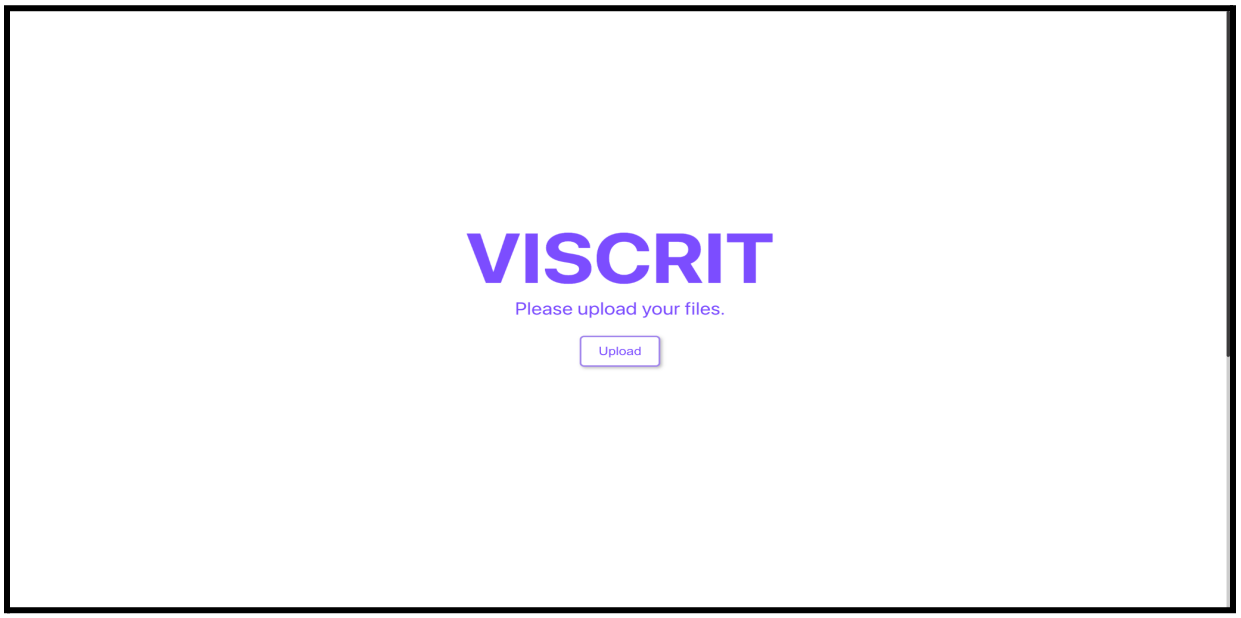


Figure 2: Student selects desired XLSX file and PDF

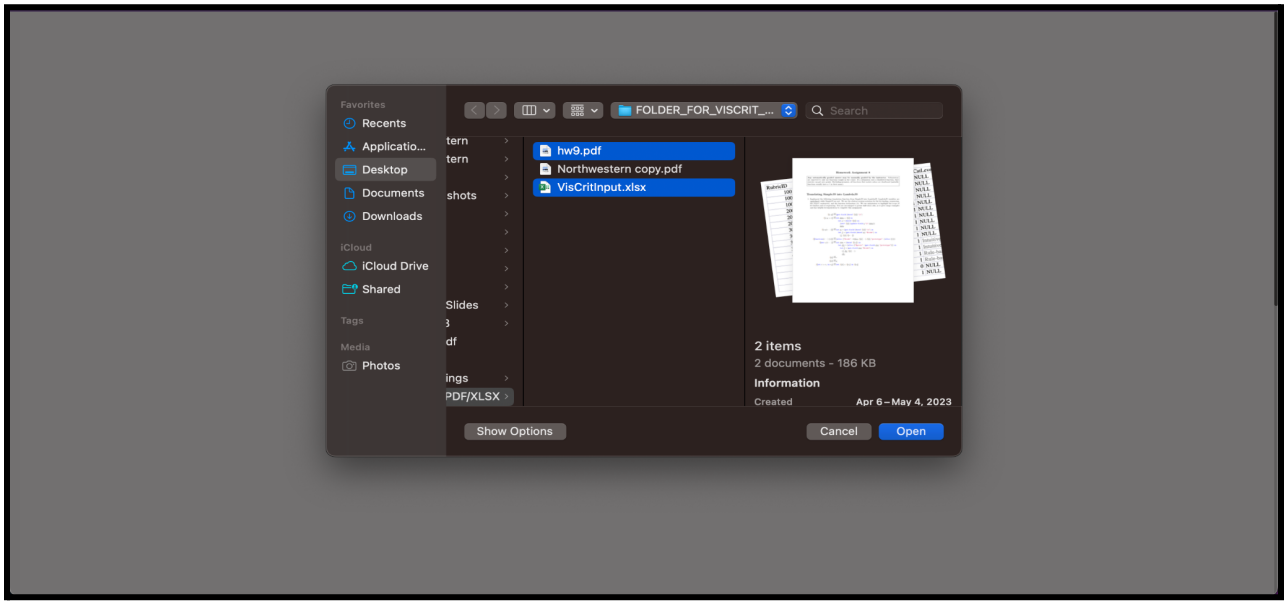


Figure 3: CritiquerID Selection

Figure 4: Rating and location button where a user can select the location that an annotation goes and rate it.

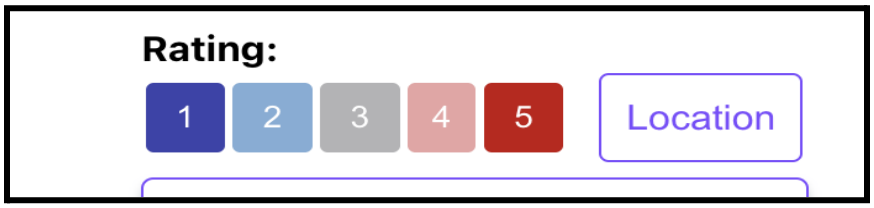


Figure 5: Main critiquing page where users can choose the correct critiquerID and begin their annotations

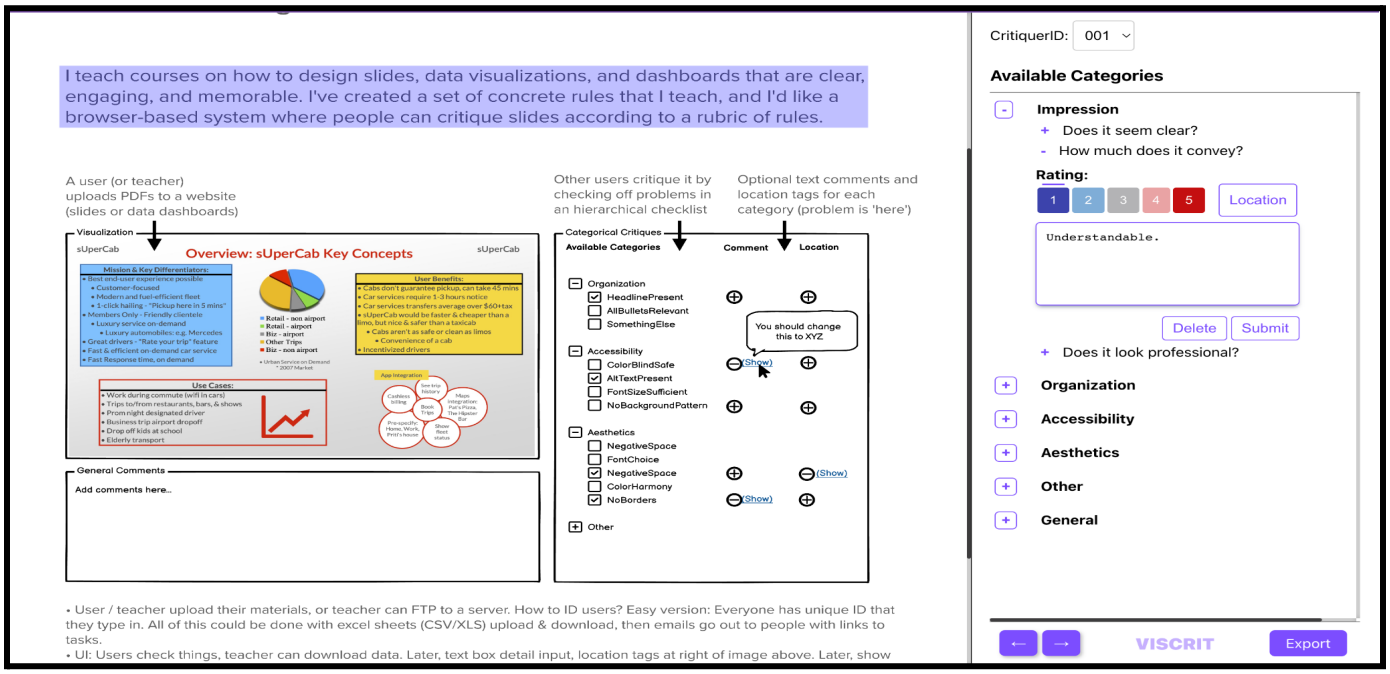
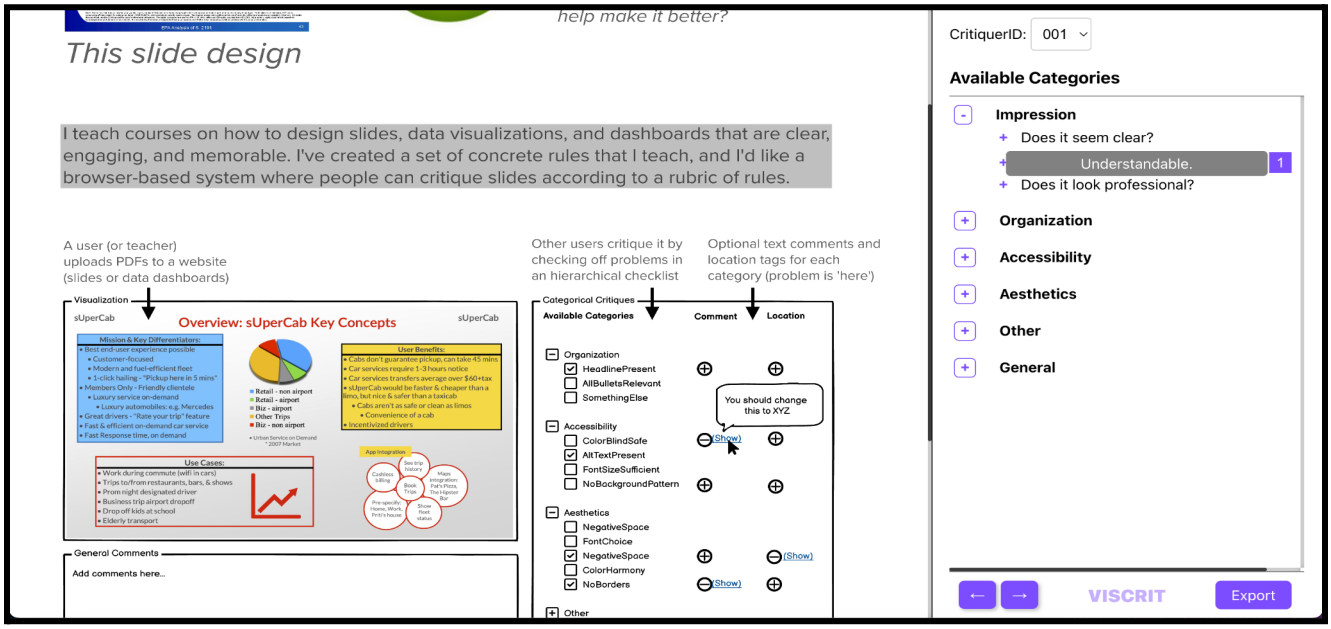


Figure 6: Once a rating has been submitted, a user can hover over the rating to display the annotation made and the comment.

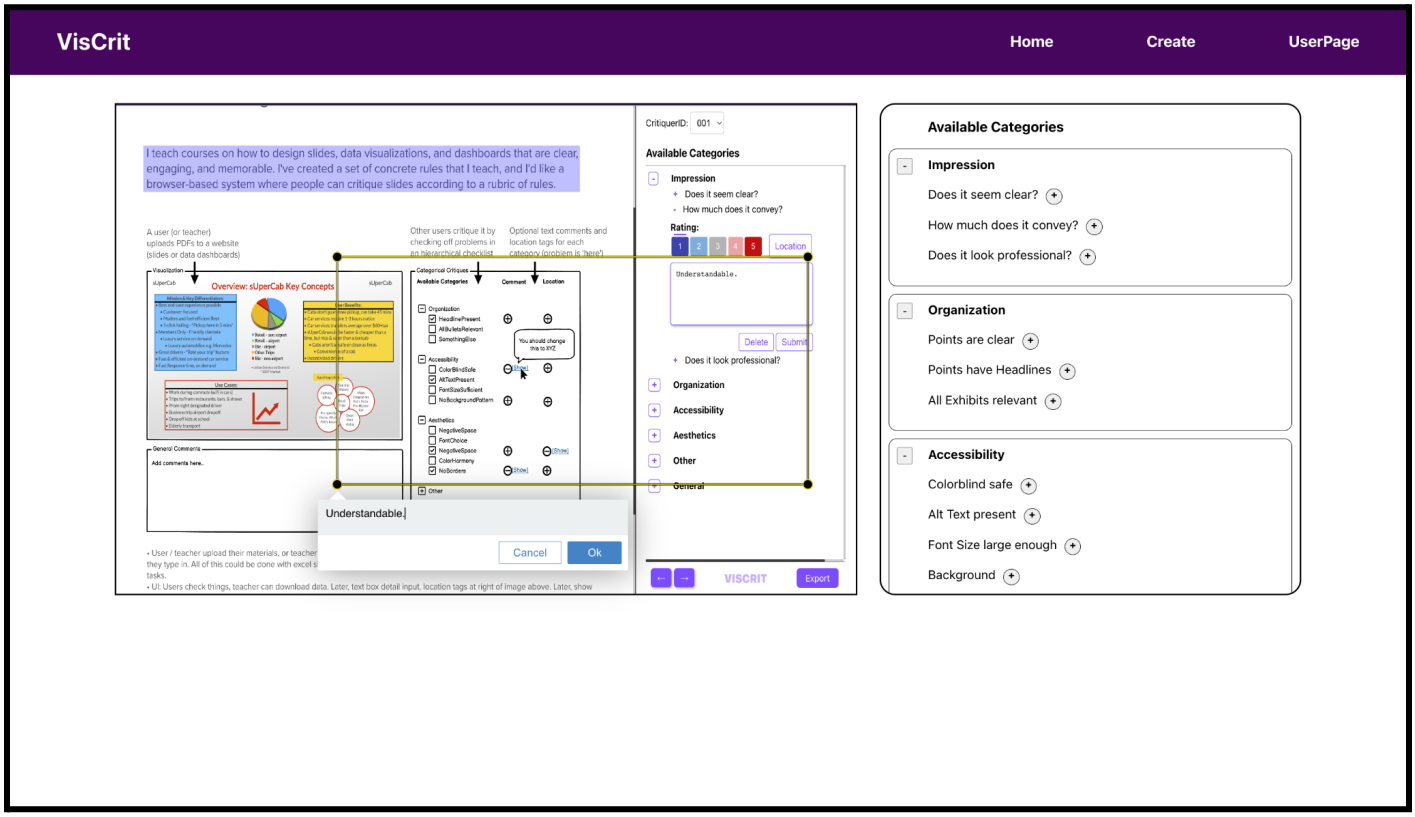


Here is an example rubric and dashboard.

5 Deployment and Testing

- During the course of the semester, we realized many issues with VisCrit during our prototyping phase. For example, when trying to implement PDF rendering we found out that many of the external libraries were not compatible with our current state of development for our annotations.
- To combat this, we used the canvas API in conjunction with pdf.js to overlay the uploaded PDF overtop the canvas for our annotations
- Another issue that we had during the testing phase was getting the location of the annotation to stay in the same coordinate location on the page when scrolling and switching pages.
- This issue was resolved by taking the coordinates of the annotation and storing it into an object array.
- In earlier versions of VisCrit, PDF documents were not supported, only PNG, IMG, and JPG. In this previous version we used marker.js library for image annotation support.

Previous version of VisCrit(Image support only), with annotations



https://github.com/longy2k/VisCrit/tree/54be0e2b5af1776a5566477eb37b91af86385e65

6 Supplemental Documents and Notes

https://github.com/mozilla/pdf.js/

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

https://github.com/parallax/jsPDF