Oregon State University
College of Engineering

COMPUTER SCIENCE CAPSTONE

MAY 25, 2020

IMPROVEMENTS TO IMAGE ANNOTATOR AND INTEGRATION WITH CYVERSE

PREPARED FOR

OSU EECS RESEARCH

FUXIN LI _____ _____

Signature *Date*

PREPARED BY

GROUP 22

NICHOLAS KIDDLE _____ _____

Signature *Date*

Abstract

This document is a summation of all work relevant to the 3 term long Capstone project of Intelligent DEep Annotator for Segmentation.

CONTENTS

1 INTRODUCTION

1.1 Understanding the Problem

Image Annotator aims to use computer vision to enable researchers in biology to effectively record and report findings. The current iteration of Image Annotator allows researchers to preform the required actions of annotation at a multi-layer level on plant tissues. However, its user interface implementation does not place its ease of use much above it's current analog counter parts. The user interface needs optimizations to allow for quick and easy user actions. This will make the Image Annotator implementation more relevant over its counterparts.

Currently Image Annotator is only able to run on a dedicated Oregon State University server. This limits the number of users we can have to researchers at Oregon State University. It is the hope that this project will be able to help scientists beyond Oregon State University.

Specific issues with the interface its navigation and usability. The aspects of the interface do not respond well to user actions. Because of this there is a discontinuity between what the user expects to happen and what actually happens. Users must navigate many clicks and mouse gestures to create an annotation label. This can be shortened to be a more fluid experience.

1.2 Client

The client for this project is Dr. Fuxin Li. He is a professor at Oregon State University, this project is part of a research project he works on sponsored by the National Science Foundation. Dr. Li would be available for meetings when needed and would reply to quick questions over email.

1.3 Furthering the Project

For the next steps to take in this project please refer to Appendix 3.

2 REQUIREMENTS

Revisions: Revised February, 18, 2020.

Section	Original	New
Specific Requirements	Keyboard Shortcuts for changing between classes and objects.	Reduce scope to creating input text submission on key stroke.
Specific Requirements	Vague mentions of creating CyVerse compatibility.	Generate more specificity on how to integrate with CyVerse
Specific Requirements	No mention of quality of life features.	Add quality of life features to requirements.

2.1 Introduction

2.1.1 Purpose

This document is intended to coalesce the project's parameter's for completion. It will also serve as a binding contract between the client and this team who has been tasked with the following through with this project till its completion as per the the CS461 class requires. Upon the final deadline this document will also be used by the CS461 professors and graders to measure and determine the grade that the team shall receive for their work.

2.1.2 Scope

The product will be used by biologists for annotation of plant tissues. The product will serve as a tool to aid researchers in their gathering of data. The product must be able to segment objects in an image accurately and be easy and intuitive to use for all researchers. The product must be available on a platform, CyVerse, which will provide resources to researchers allowing for easy startup costs.

2.1.3 Abbreviations

Abbreviation	Description
CS	Computer Science
CS461	Senior Capstone Project
NSF	National Science Foundation
GPU	Graphics Processing Unit

2.1.4 Overview

There are two more sections in this document: Overall Description and Specific requirements. The former of which is an overview of how the product is intended to be used. The latter is a compilation of all the requirements for the complete product.

2.2 Overall Description

2.2.1 Product Perspective

This project will be a continuation of an ongoing research project funded by the NSF grant. The aim is to create an image annotation tool for biologists to use to annotate plant tissues through machine learning algorithms. The project allows for the segmentation/separation of objects within an image. Machine learning will not be explored in this project as it is not within the scope. The project takes form in two ways. The first being the user interface.

This is the side of the product to which the users will be interacting with. The interface is composed of two different screens. The first screen the user encounters on start is a web page that asks for the user to upload an image for annotation. To upload an image the user must click on the upload button and up will come the operating system's file

finder. Here the user simply selects and confirms the image for upload. Upon upload the user is taken to the second screen.

The second screen is a new page; in the center is the Image that they have just uploaded. This part of the page takes up the most screen real estate. This is also where the most user interaction takes place, as it is where the user will draw with the mouse cursor to start image segmentation. Above the Image is the Tools and Utilities section. In this section the user has options to zoom in and out on the image, select which pen to use for annotation as well as its width, a bounding box selector to limit the scope of the image, and the process button to process an annotation of the selected object. This is the area that has the second most interactions. To the left of the Image is the Hierarchy Panel which contains the ability to create an object as well as select an object to annotate. Each object here can be assigned a color and has information about whether it has been processed or not. Finally left of the Image is the History Panel where you can see a log of processed annotations as well as options to undo and redo an annotation. This makes up all of the user interface.

The second side of the project aims to bring the functionality described above to a wider audience. The image annotation uses computer vision to recognise objects in an image. This operation is done through machine learning which is computationally expensive and requires expensive hardware to make it feasible. If the product is to be used by more researchers the cost of entry must be lower. This is accomplished by integrating with CyVerse. CyVerse offers a platform for serving software and with their resources. Essentially they are doing the heavy computations that the product requires. The current state of the product does not allow for CyVerse integration and will be explored as a new aspect of the project that has been untouched so far.

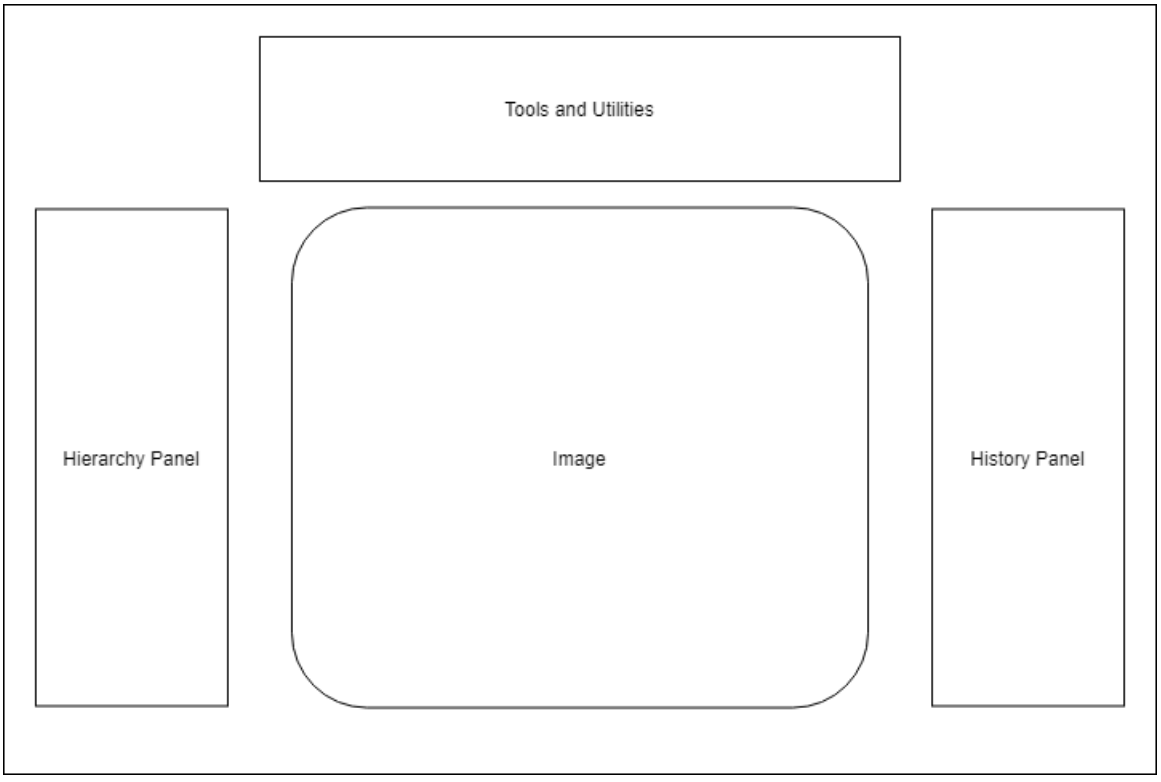


Fig. 1. Graphical depiction of the user interface elements and their relative positions for screen two.

2.2.2 Product Functions

The product will function as a research tool for annotation plant tissues by biologists. A user just needs to upload their own image and then they can begin annotation. The product allows for annotation of many different objects in a single image. Each object contains its own undo and redo history and can be set to be represented in any color.

2.2.3 User Characteristics

The product is designed for researchers and more specifically biologists doing plant research. The product uses some sophisticated technologies like machine learning, but this is all going to be abstracted away from the user. There is an expectation that the user will have a science background, but anyone should be able to annotate any image with this software. The product will not require any sort of base education level so we expect all users to be able to understand and use the software. We can determine that it is possible that anyone could use the software as the machine learning model can recognise any object that has a distinct shading or outline and not just plants.

2.2.4 Constraints

The software requires powerful GPUs to be able to complete the machine learning tasks. This means that the user must have access to their own computational resources or use the CyVerse implementation of the software that is being asked by the client.

2.2.5 Assumptions and Dependencies

The client has deemed that CyVerse be the choice platform to integrate with since it is a hub for science data driven technologies. One caveat as of writing is CyVerse does not currently have a GPU resource publicly available; though there are talks of the organization adding these soon.

2.3 Specific Requirements

The product must be able to recognise the object that the user has chosen to annotate in the image. This means that the user's pen markings must align with the image being processed. If there is a mismatch in the data then the wring object could be annotated.

The product must be quick to use. The most important and most used elements should be close to each other. This must be in an ordering that makes sense with both the number of interactions and the relationship of two interactions. i.e. The pen should be close to the image because they are both used often and are always used together.

The user should always be aware of the state of the interface. Often times the image processing for an annotation may take up to ten seconds. During this time the user interface is disabled. This means no buttons or elements are interactive. There is currently no user feedback that an image is processing other than a disabled interface. With no feedback provided it is as though the interface has crashed and may result in a user to prematurely restart the interface.

The user should not require any high-end hardware to run the system. Integration with CyVerse will allow the user to have a low cost of entry into our software.

The product must function flawlessly from an interactions standpoint. No interaction should have erratic or unexpected behavior. This means that semantic mappings must be intuitive between all elements of the page. i.e. A button 'undo' should behave like an undo in all other systems and only undo one action at a time.

The product will not be difficult to parse. High contrast color schemes will make all elements on the page legible for those who may be hard of sight or colorblind. This is necessary for appealing to a large group of users.

The product will be able to allow users to place a bounding box in the image to reduce the scope of annotation of the image. This will reduce wait times when processing because only the bounding box is being searched for an object instead of the whole image. The bounding box will always be a subset of the pixels in the whole image.

There will be quality of life improvements such as focusing on often used text-boxes on launch of the site and allowing for the ’enter’ key to submit the form as well as move the focus to the next text-box. This will allow the user to navigate the UI more efficiently than previously was available.

2.3.1 Gantt Chart

3 DESIGN

3.1 Introduction

3.1.1 Purpose

This document is intended to inform and explain the implementation decisions by this group. There will be a break down of each feature that project requires for its completion. Each feature will have their implementations explained and why they are suitable for this project given their scope. This document is meant to serve as a reference for when working on the actual implementation and will be updated to changes as necessary. Finally it will serve as a source of information for later personnel to build on top of.

3.1.2 Scope

The product will be used by biologists for annotation of plant tissues. The product will serve as a tool to aid researchers in their gathering of data. The product must be able to segment objects in an image accurately and be intuitive to use for all researchers. The product must be available on a hosted platform, which will provide resources to researchers allowing for easy startup costs.

3.1.3 Stakeholders

This document is intended to guide stakeholders' expectations of the project's implementation and goals. The hope is that this document serves as a well communicated guide to the methods of through which this project will be conducted.

3.1.4 Abbreviations

Abbreviation	Description
UI	User Interface
JS	JavaScript
XML	Extensible Markup Language

3.2 Designs

3.2.1 Interaction Viewpoint

3.2.1.1 Viewpoint Description: This section will take a look at the way in which a design user will interact with the system to be implemented. A design user should be able to distinguish the intended features to be added onto the current system. This will serve as both high documentation of the implementation and a guide to using the described features.

3.2.1.2 Design Concerns: The stakeholder has made a request for a number of features to be added to the current system these features are intended to address usability concerns in the user interface.

3.2.1.3 Design Elements: Bounding Box: The bounding box system on the user interface is currently broken and requires fixing. The bounding box is a selector tool used to constrain the image being annotated to a smaller cropped are. The purpose of this tool is to speed up image processing times. When a bounding box is applied it creates a subset of pixels where the super-set is the entire image. Now when the image is sent to the back-end of the system, where the machine learning algorithms are applied there is a smaller amount of pixels and thus computations that need to be made.

In order for the bounding box to be considered working it must meet these constraints: selected region started by mouse-down event and ended by mouse-up event is correctly represented in back-end data structure, visually show this selection on image with a colored rectangle outline, verify that panning the image and zooming in/out on the image does not result in shift of rendered rectangle (relative to image not view-port), and verify that ”clear rectangle” button removes the rectangle from the view-port and back-end data structure, inspect for any other misbehavior’s of the bounding box when using other aspects of the UI.

The implementation for this will be conducted in the JavaScript language. There will be need for use of debugging tools provided by Chrome browser’s built-in tool set. With the use of these tools it was concluded that the bounding box was sending the data to the back-end correctly and receiving the new data back correctly, but there are misalignment’s in the visual representation on the UI from the image mask that is returned.

There is a spot in the code where X, Y coordinate pairs are converted to an index for a one-dimensional array that represents all the pixels in the image. The pixel coordinates that are given are to be color shifted to represent the object mask on the image. To do this the proper conversion must happen from the X, Y coordinates to the one-dimensional array index. The issue seems to be in this conversion, but only when the bounding box is in use. The best hypothesis for this issue is to consider that the shift in the coordinates is happening because of a change in the sizing of the image; this could be by the bounding box height and width. One way to test this would be to print out the coordinates at the back-end and then at the front-end prior to the index calculation. Then print out the current bounding box width and height. Now we can measure an approximate change measure visually in the UI (displacement of X, Y mask relative to where the object is in the image). We can then see if this displacement matches our height and width of the bounding box. If this is proves to be a correct assessment of the issue then we can proceed by accounting for this shift if there is a bounding box present.

Quality of Life Improvements: To increase usability and speed of use there is a request from the product users that there are keyboard shortcuts added to the text-boxes in the UI. As well as adding a this shortcut adding automatic element focusing so a user can does not need to click on the text-box.

To accomplish this there is a small open source JavaScript library that can be included called shortcut.js. In the library there is a simple function called shortcut.add(). The function takes two arguments. The first of which is a string argument. The string will be the keyboard command you would like to bind. This command needs to use the correct keywords in order to interpret the desired key correctly. For this implementation we will just be looking for the ’enter’ key and that the text-box is in focus.

The second argument is a JavaScript callback function that will do the desired behavior for when the shortcut is executed. Since this is a web page the only interaction we care about is a click. But since the second argument is just a JavaScript function we can do anything that JavaScript allows when a shortcut is executed. [10]. The draw back is that there needs to be an additional file loaded into the DOM when we use this library and the order of including will matter.

As well as the the keyboard binding it is requested to focus on the text-box on start up. The text-box is not a static element on the page. It changes based on a tabbed structure. Since there are two different tabs it will need to be checked which is active and then change focus based on this. The active notation comes in the form of a HTML class. So in JS we can check whether this object has a class 'active' and then change focus onto its first input child element. This behavior will need to happen on both tab, but since the layout of the tabs are the same this should be a simple callback function when a click event is detected on the tab or when the tab's classes change.

System Status Indicators: The current system has no status indicators to provide a user feedback when an action is taken. One of the main examples of this is when the user clicks the "Process" button. This button initiates the network exchange of the annotated image from the client side to server side. Once the image is on the server side machine learning algorithms are working to identify the marked up object from the image and annotations the user had created. This process can take up to ten seconds. During this time the user has no idea if their process request has been sent. The UI remains in the same state until it until the processed annotation returns and is displayed superimposed over the image in the UI. During this network communication period it would be best to show visually a loading or waiting symbol as well as disable the "Process" button. To do this in JS we can use document.body.style.cursor='wait' call when the network communication is initiated. This will change the cursor to be a known operating system specific loading/waiting symbol, one that the user will recognise. We will also disable the "Process" button during this time which will visually grey out the button. Then when the network call response comes in we can reset the cursor and the button back to their normal state. The important part here to note is that we will begin our UI changes once the user click on the process button has begun. Then we only end when we receive a response from the back-end. This implementation will work even if the back-end processing does not and we get no response from the back-end. We can assume this as a response will come, just in the form of a timeout response.

3.2.2 Information Viewpoint

3.2.2.1 Viewpoint Description: This viewpoint concerns the storage of persistent data and data structures.

3.2.2.2 Design Concerns: We will be looking into the use of a hosting system, CyVerse, to allow for persistent storage of annotation objects in the form of XML.

3.2.2.3 Design Elements: Points to consider when making solution decisions.
CyVerse Hosting System: CyVerse is not a big scale hosting service. CyVerse's mission is to "To design, deploy, and expand a national Cyberinfrastructure for Life Sciences research, and to train scientists in its use."[1]. CyVerse offers data storage and web based application hosting from cloud infrastructure servers for computation. Cyverse offers data storage for users into their Discovery Environment.[5]. With access to their data storage each user could save off their annotations to the CyVerse Discovery Environment under their personal account. This would allow for access to previously made annotations to be reviewed or continued.

For a user to save a our XML files to their Discovery Environment they would just have to hit the save button and CyVerse will automatically pop open its file explorer. This is one of the advantages of CyVerse, apps are integrated into their system. This XML document will include classes, objects, masks, labels, and image data for the current working

annotation. There is already functional version of creating the XML file included in the current system. The difficult part of implementing integration into CyVerse is in the docker container build. CyVerse allows users to submit custom web applications in the form of a docker container which they provide a base for. It will be this project's task to integrate the dependencies for the image annotator into a docker container that is compatible. There will be need to do port forwarding so we can reach the outside world from our docker.

The biggest challenge is that we will to put together a minimal dependency list given the CyVerse hardware. The Image Annotator uses Tensorflow-gpu to speed up its neural net computations since it is Cuda enabled. The compatibility of the version needed will depend on CyVerse hardware. This will make future maintenance difficult if CyVerse update's their GPU skews.

4 Technical Review

4.1 Introduction

4.1.1 Purpose

This document is intended to inform and explain the implementation decisions by this group. There will be a break down of three technologies that the project requires for its completion. Each technology will be broken down into up to three different implementations and why they are suitable for this project. From these implementations one from each will be chosen and explained why it was selected.

4.1.2 Scope

The product will be used by biologists for annotation of plant tissues. The product will serve as a tool to aid researchers in their gathering of data. The product must be able to segment objects in an image accurately and be intuitive to use for all researchers. The product must be available on a hosted platform, which will provide resources to researchers allowing for easy startup costs.

4.1.3 Role

In this document the technologies that are discussed are solely my, Nicholas Kiddle, responsibility and my area of expertise for the remainder of the project. I will be responsible for the research and planning of these areas and take initiative in seeing through the completion of their implementation. These areas are implementing a hosting service, developing with the most efficient JavaScript Framework, and creating keyboard shortcuts for UI elements.

4.2 Technologies

4.2.1 Hosting Systems

4.2.1.1 Constraints: For the scope of this project there are two main components to focus on, computing and data storage. The project is a web based application that runs off a python platform so any operating system should be capable of running the web back-end. There are heavy machine learning aspects of this project that require use of GPU systems to improve image processing times to be under 10 seconds. It would be best if there was an emphasis on AI ready GPUs.

4.2.1.2 Amazon Web Services: Amazon Web Services (AWS) is the biggest cloud market in the world and it offers a wide range of capabilities. For the scope of this project there are two main components to focus on, computing and data storage. AWS Compute offers a service called Elastic Compute Cloud (EC2) offers GPU compute instances. These GPUs are NVIDIA V100 Tensor Core GPUs and one instance can be equipped with up to 8 GPUs[1]. The GPU from NVIDIA has been designed with AI in mind. NVIDIA's product page boasts a 3 times faster deep learning training than its last generation hardware[2]. The current implementation of Image Annotator is running on an outdated OSU server which is able to achieve sub ten second processing times from only two GPUs both of which are multiple generations older, circa 2012.

Currently there is no long term storage requirement that needs to be satisfied by the product. This means that no user data will be saved into the cloud, but there is a requirement for instance storage. AWS offers per instance storage so each time someone were to launch the interface from a browser they would have their own instance storage that is

only going to persist as long as they keep the site running[1].

4.2.1.3 Azure: Azure is Microsft's answer to AWS. Azure is the second biggest cloud market in the world and is AWS's direct competitor. Our product is a web app which is one of the product lines that Azure supports, The languages that are supported by Azure Web Apps are HTML, Python (for Linux), .NET, Node.js, PHP, and Java[3]. This is where the first road block is hit. Our product is launched from python which is a supported language for launching a web application on Microsoft's Azure, but they limit it to Python for Linux only. It is more than possible to launch Image Annotator from Linux, but is a constraint to consider. Azure offers multiple GPU skews the low end of which is coming with 1 NVIDIA Tesla P100 card. The highest end skew is their NCv3-series which is outfitted with up to 4 NVIDIA Tesla V100 GPUs[4], which is more than enough for the product that we are considering to launch. Azure's pricing models are more convoluted and complext to disern, they do offer data storage options, but with less flexibility. For their Web Apps there is no per instance pricing [3].

4.2.1.4 CyVerse: CyVerse is not a big scale hosting service. CyVerse's mission is to "To design, deploy, and expand a national Cyberinfrastructure for Life Sciences research, and to train scientists in its use."[5]. CyVerse offers data storage and web based application hosting from cloud infrastructure servers for computation. CyVerse currently does not offer any GPU skews and this is going to be an issue in completing the project. CyVerse has told the client that they intend on adding GPU resources to their infrastructure, but there is no time table on when this may be be added and when the documentation for using such features will be available. Cyverse does offer data storage and integration into their authentication services for each user [5]. CyVerse's biggest offer is the user base that they have. They market software products that are directed towards scientists and researchers which is persisely the target audience of our product.
4.2.1.5 Decision: With all aspects of each project it is best to choose Amazon Web Services to host the project, however, the client has requested that CyVerse be used, so that is what will be implemented. AWS has the best resources for serving and hosting our product as well as being the most flexible and transparent of all the option.

4.2.2 JavaScript FrameWork

4.2.2.1 Constraints: One piece of the project is to improve the usability of the existing system that is in place. To do this effectively a JavaScript (JS) framework would be useful in keeping the methods standard. The web page is simple and only has two screens so no heavy UI lifting will be needed.

4.2.2.2 Skel.js: Skel.js is a simple and lightweight JS framework. Some of the features that it offers is the ability to make CSS break points, a CSS grid system, css shortcuts and plugin support[6]. There is a wide variety of plugins for skel which makes it more flexible than what it offers. Skel would be best used for when implementing a web page for screen sizes that may differ drastically. The framework offers screen adjustments in the CSS. This is best if you expect users to have a varying screen size that they may be accessing the site from. In the code skel will detect the screen size and will determine whether it falls into a user designed category of sizes. This acts like a jump to method. If the screen is of one category then do these layout designs and not the others.

4.2.2.3 React.js: React is another JavaScript framework that is designed for creating anywhere from simple to complex UIs. One of the benefits of react is that its components are independent of other code that you have. "you can develop

new features in React without rewriting existing code.” [7]. This property makes it great for using in a system that has already been implemented without a framework or with another one. One of the biggest features of react is its virtual Document Object Model (DOM).This allows for increased efficency when your web page has a lot of data changes. This allows for the virtual DOM to be updated to the new values without having to re-render each element before moving on to the next[8]. Now all the changes can be pushed at once and this puts less stress on the browser.

4.2.2.4 Vanilla JavaScript: Vanilla JavaScript or just JS is a rarity these days because there are so many frameworks that offer many features and abstractions from what JS is. JS is a great way to get things done with little to no start up time, ”And frameworks come with their own headaches.They need to be setup (often no simple task) and patched. Build systems magically break and stop working.” [9]. The documentation for JS is not that great, but it is easy enough to use and if you have other programming experience. Frameworks are great at doing things that JS isn’t so good at like adding and updating to the DOM, but everything is achievable in JS and it comes down to a cost benefit of whether the time it takes you to implement the same functionality in JS is worth the time it takes you to setup and learn a JS framework.

4.2.2.5 Decision: A combination of Skel.js and vanilla JavaScript will be best for this project. Skel has been already implemented in the current system so undoubtedly knowing it will be needed at some point during the UI renovations. Skel is such a small framework that it alone is going to need some vanilla JS ontop to make everthing else work. Some of the reasons React was not considered was the start up time that it would take to learn a new framework and the fact that its main benefit is for when a site has a lot of data changing. The site that is being worked on is for the most part static and does not contain as much data as say twitter or Facebook.

4.2.3 Keyboard Shortcuts

4.2.3.1 Constraints: A usability request by the client was to add keyboard shortcut mappings to most of the UI elements to aid quicker navigation while annotating an image.

4.2.3.2 shortcut.js: This is an open source library for JavaScript that has a function for adding shortcuts to a web page. It is a simple function called shortcut.add(). The function takes two arguments. The first of which is a string argument. The string will be the shortcut command you would like to bind. This command needs to use the correct keywords in order to interpret the desired key correctly. For example the usage for the control key would look like ”CTRL” and not ”Control”. Then this would need to be followed by a ”+” and then another key on the keyboard.

The second argument is a JavaScript callback function that will do the desired behavior for when the shortcut is executed. Since this is a web page the only interaction we care about is a click. But since the second argument is just a JavaScript function we can do anything that JavaScript allows when a shortcut is executed. [10]. The draw back is that there needs to be an additional file loaded into the DOM when we use this library and the order of including will matter.

4.2.3.3 JavaScript Implementation: The other option is to make our own implementation of the library shorcuts.js. This would possibly be beneficial to save space in our program and not have to messy the HTML with more script

includes. Doing our own implementation could allow for more specificity where we do not need to allow for all keys to be available in a shortcut. This would provide possibly some optimizations and could be useful as the program will require lots of keyboard shortcuts, nearly every button. Doing our own JS implementation would be a good security practice over using open source software. If we were to use an outside library there could be a vulnerability introduced into the system and this could make integrating with our host much harder if they are to vet our code.

4.2.3.4 Decision: shortcuts.js is the obvious choice. This does save a lot of time from implementing our own version of a JavaScript shortcut handler. This implementation also would be easy to implement on the existing system. We can simulate a click on the element in the DOM that has been bound to the shortcut. We already have click event handlers in place for these so this would be a simple and low risk implementation. While it may be risking some security and optimizations generally speaking open source code is safe as long as it has been vetted by many others in the community. We can still do our own vetting and optimizations if we feel the need to.

5 Progress Reports

5.1 Fall Term

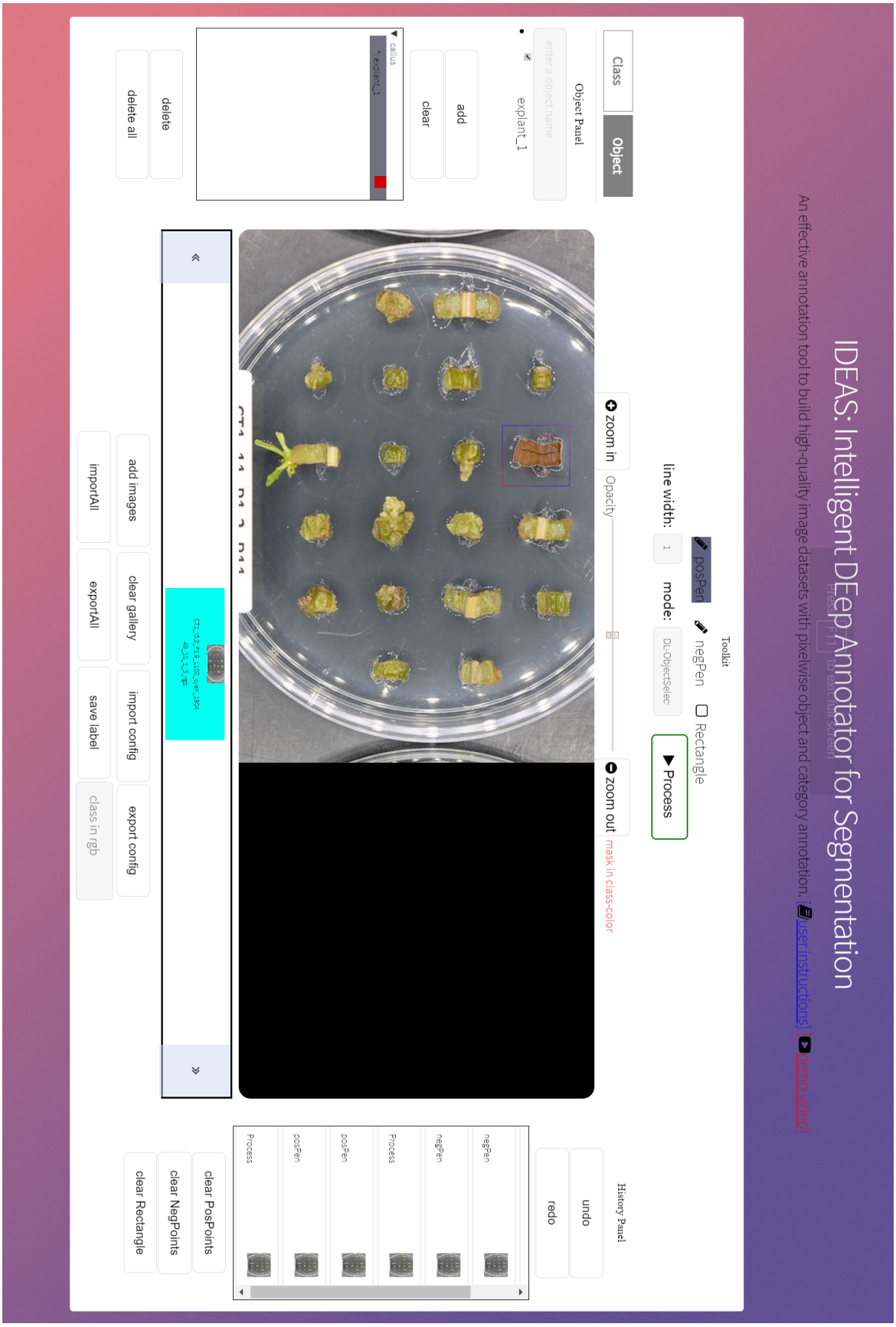
Week	Progress	Problems	Plans
4	I completed the requirements document draft 1 on time and scheduled a meeting with my client for the coming Monday at 4pm.	Still am iffy on my specific requirements for my project. Given all the documentation that has been assigned and being a group of one I have not had anytime to start work on the clients project and have been stuck doing work for the class. This just makes me feel bad about not being able to complete expectation that have been assigned to me.	Going to spend a few hours working on code this weekend for the client. Going to meet with client on Monday.
5	Met with client on Monday and was given a specific set of usability features that they are looking to be added. We also set up regular meeting times to be every other Monday at 4pm. The list was given to me on Thursday. Turned in my Requirements Draft 2 and started on my Tech Review Draft 1.	Not a whole lot of time to start on client code and usability features with all the drafts looming. The requirements list that the client gave me is not too detailed and I will need more information on them to be able to complete.	Start on some client code over this weekend and hopefully finish a usability feature next week. Get the Tech Review draft done too. Also I will ask for clarification the features given to me when I meet with the client again.
6	Finished Tech Review and worked on some client code to get the first usability feature done.	Was a little unsure about the layout of the Tech Review and what needed to be included.	I meet with my client next week on Monday and hope to get some more clarifications on usability features that they have requested.
7	Started working on client code and completed the first of 6 usability tasks given to me. Finished Tech Review and started the Design Document.	Postponed meeting with client this week due to holiday.	Work on more client code and meet with client next Monday. Also finish up the Design Document.
8	Worked on client code and met with client to ask questions about the existing system. Also worked on and turned in the Design Document Draft 2.	Didn't have any issues this week or any blockers coming up.	Fix bug in client code that I discussed with client this week. Compile and send in my documentation to client.
9	Submitted Client Verification documents to the client.	Hoping thanksgiving break give the client enough time to review and respond.	Work on client code and meet with the client on Monday.

5.2 Winter Term

Week	Progress	Problems	Plans
1	In correspondence with a hosting provider to see if they have the resources the project needs. Finished the first feature, a bounding box for image segmentation. Then went to my first team meeting and met everyone involved on the project. This helped with getting more context of the project.	I have a lot of meetings that will take up valuable time for capstone.	I have to still find out about hosting resources and can start looking into the next feature, quality of life improvements for the UI.
2	This week was a slow progress week. I went to the weekly meeting the research group has and did a little debugging. I also finished email correspondence with the hosting site we are exploring.	Sent hosting site information I had gathered to the client but have not received a response.	Hoping to meet with the client soon to talk next steps.
3	Met with Richard my TA this week and addressed my concerns with not having enough work to present due to uncertainties the client has with what he wants me to do. Met with a graduate student who also works on the project and we solved an issue with the image annotator code after merging a dev branch with master.	Client is unsure whether I will be implementing the hosting system adaptation. This was the main part of the project and I have not been able to start due to client. I am concerned that there will be little work to do on the project if this falls through and if it doesn't it could mean a lot of work to meet the requirements in such a short period of time.	Got in contact with client and have a meeting next week, hoping to get clarification of my requirements.
5	Started to look into the CyVerse Docker requirements.	Hoping I can get CyVerse approval in time for beta or even expo.	Finish poster draft 1 and look into CyVerse docker more and get one made up.
6	This week I completed my poster draft 1 and took a deeper dive into what I need to do in order to get a CyVerse docker instance created for our project.	I don't foresee many. Timing of getting CyVerse hosting up and running could be an issue as once I submit it to them it is out of my hands.	I plan on trying to complete the CyVerse docker then I just have a few nagging bugs to iron out.
7	Gave design review presentation this week. But more importantly I got a preliminary docker	I know that it is now possible to replicate the docker container I need but I am having trouble	I emailed CyVerse and they gave me a docker container for Nvidia to try

	set up working for CyVerse Integration.	getting nvidia dependencies to install as they require command line input that I cannot pass into the container.	to use. Hopefully this solves the issue.
8	I submitted my poster for review and created a github repository for my new docker container build guide to live in.	None, yay!	Upload my docker container to DockerHub and work on bug fixes for the existing code that I have.
9	Did not make any progress this week. The previous week I put more time in because I knew I would be busy this week.	None, the project is shaping up nicely.	Upload docker container to DockerHub, fix bugs in code (about 2 days of work), submit docker container to CyVerse for hosting.
10	Re-did poster for a higher grade. Implemented last quality of life feature to the UI.	None!	Write final report and make video demonstration.

6 User Interface Showcase



7 DOCUMENTATION

7.1 Building a New Docker Container

7.1.1 Requirements

- Host system with NVIDIA GPU
- 10 - 20GB of storage space
- Host system with internet connection
- Tested on Ubuntu 19.04 and 19.10

7.1.2 Installation

- git clone this repository
- image-annotator/ directory to be placed in image-annotator-docker/
  - git clone image-annotator/ to image-annotator-docker/ from https://bitbucket.org/JialinYuan/image-annotator/src/master/ (no install needed)
  - Download and extract model.zip to image-annotator-docker/image-annotator/deep_interactive/
    - https://oregonstate.box.com/s/dq4tc94f1299lml52a7bh7dztjkhvrj5
- Modify image-annotator-docker/image-annotator/config.py
  - find line: __C.PosNeg_Model_weight_Path = './deep_interactive/model/PASCAL/pos_neg/model.ckpt-80000' and change it to: __C.PosNeg_Model_weight_Path = '/image-annotator/deep_interactive/model/PASCAL/pos_neg/model.ckpt-80000
- cuDNN 7.1.4 and place "cudnn-9.0-linux-x64-v7.1.tgz" in image-annotator-docker/
  - https://developer.nvidia.com/rdp/cudnn-archive
  - Select "Download cuDNN v7.1.4 (May 16, 2018), for CUDA 9.0"
  - Download "cuDNN v7.1.4 Library for Linux"
- Docker to be installed on host machine
  - https://docs.docker.com/engine/install/ubuntu/
  - Complete postinstall steps: https://docs.docker.com/engine/install/linux-postinstall/
- nvidia-docker (2.0) installed on host machine
  - https://github.com/nvidia/nvidia-docker/wiki/Installation-(version-2.0)

7.1.3 Build and Run

- $ ./start_img_ann (wait a long time for the build)
- Open browser to 0.0.0.0:5000/

8 Technical Resources and More

Listings in order of most helpful to least:

- CyVerse App / Tool Creation - https://learning.cyverse.org/projects/vice/en/latest/developer_guide/building.html
- CUDA Library Archive - https://developer.nvidia.com/rdp/cudnn-archive
- NVIDIA Docker Installation Guide - https://github.com/nvidia/nvidia-docker/wiki/Installation-(version-2.0)
- CyVerse's Docker Guide - https://cyverse-creating-docker-containers-quickstart.readthedocs-hosted.com/en/latest/

9 Conclusions and Reflections

During the course of this project I have managed to pick up some new skills and learned some new values. Coming into the project I had very little experience working on a project that was this large. There was much code already written and my resources for learning what had been done by others was quite limited. This taught me the importance of documentation, code comments, and self-documenting code. To integrate with CyVerse I needed to learn Docker, which I found to be a great tool to know. As the year went on my new knowledge in Docker helped me in other course projects. Working on a project alone made me appreciate being able to ask others for help, there were many times where I would have like to bounce ideas off of someone else, but did not have the opportunity.

APPENDIX

.1 Code Listings

.1.1 Dockerfile

Builds a Linux based container that will start the webserver on the ”docker run” command.

```
FROM nvidia/cudagl:9.0-runtime-ubuntu16.04

# Add some metadata to describe our container
LABEL maintainer="Nicholas Kiddle"
LABEL maintainer_email="kiddlen@oregonstate.edu"
LABEL version="0.0.1"

### Moving initial files into position
COPY ./image-annotator /image-annotator
COPY ./cudnn-9.0-linux-x64-v7.1.tgz /

### Updating Ubuntu Packages
RUN apt-get update && yes|apt-get upgrade
RUN apt-get install -y wget bzip2
RUN apt-get install -y libsm6 libxext6 libxrender-dev
# Add sudo
RUN apt-get -y install sudo
# Add user ubuntu with no password, add to sudo group
RUN adduser --disabled-password --gecos '' ubuntu
RUN adduser ubuntu sudo
RUN echo '%sudo ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers
USER ubuntu
WORKDIR /home/ubuntu/
RUN chmod a+rwx /home/ubuntu/

### Install cudnn 7.1.4 for tensorflow-gpu 1.12.0
RUN tar -xzvf /cudnn-9.0-linux-x64-v7.1.tgz
RUN sudo mkdir /usr/local/cuda/include
RUN sudo cp cuda/include/cudnn.h /usr/local/cuda/include
RUN sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64
RUN sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn*

### Install Anaconda
RUN wget https://repo.continuum.io/archive/Anaconda3-5.0.1-Linux-x86_64.sh
```

```
RUN bash Anaconda3-5.0.1-Linux-x86_64.sh -b
RUN rm Anaconda3-5.0.1-Linux-x86_64.sh

### Set path to conda
ENV PATH /home/ubuntu/anaconda3/bin:$PATH

### Getting python dependencies set for Image Annotator
RUN conda install python=3.6
RUN pip install --upgrade pip
RUN pip install flask pillow easydict opencv-python scipy waitress tensorflow-gpu==1.12.0

### Forward port 5000 from within docker container to outside docker container
EXPOSE 5000

ENTRYPOINT ["python", "/image-annotator/grabcut.py"]
```

.1.2 Bounding Box

Get the starting x and y coordinate for the user's bounding box (activated on mouse down event).

```
    drawRectBegin: function(x, y){
        var self = this;
        self.mousePressed = true;

        // clear line drawn for annotation if the user wants new bounding box
        // avoids out of bounds error in DLearning_PosNeg
        self.posPoints = new Array();
        self.negPoints = new Array();
        self.ctxP.clearRect(0,0,self.width,self.height);
        // clear the previous bounding box if there is any
        if (self.bbox.isBox){
          self.clearRectFromCanvas();
        }

        self.bbox.start_x = Math.round(x / self.scaleCanvas);
        self.bbox.start_y = Math.round(y / self.scaleCanvas);
        self.bbox.isBox = true;
    }
```

Get the ending x and y coordinate for the user's bounding box (activated on mouse up event).

```
    drawRect: function(ctx){
```

```
    var self = this;

    // check if the box is valid
    if (self.bbox.isBox===false||(self.bbox.start_x===self.bbox.end_x &&
                                  self.bbox.start_y===self.bbox.end_y)){
      self.bbox.start_x = 0;
      self.bbox.end_x = self.width;
      self.bbox.start_y = 0;
      self.bbox.end_y = self.height;
      self.bbox.bboxData = null;
      self.bbox.isBox = false;
      return;
    }

    // draw rectangle
    var h = self.bbox.end_y - self.bbox.start_y;
    var w = self.bbox.end_x - self.bbox.start_x;
    var gradient=ctx.createLinearGradient(self.bbox.start_x,
                                          self.bbox.start_y,
                                          self.bbox.end_x,
                                          self.bbox.end_y);
    gradient.addColorStop("0","magenta");
    gradient.addColorStop("0.5","blue");
    gradient.addColorStop("1.0","red");

    ctx.lineWidth = Math.round(1/self.scaleCanvas)
    ctx.strokeStyle = gradient;
    ctx.strokeRect(self.bbox.start_x, self.bbox.start_y , w, h);
}
```

.1.3 Autofocus Tabs

Check for class 'active' then focus element and assign key bind.

```
function callback(mutationsList, observer) {
      if(mutationsList[0].target.className === "active"){
        var currentInput = document.getElementById('classname');
        currentInput.focus();
        $('#classname').keypress(function(e){
          if (e.which == 13){
            if(currentInput.value.length > 0){
```

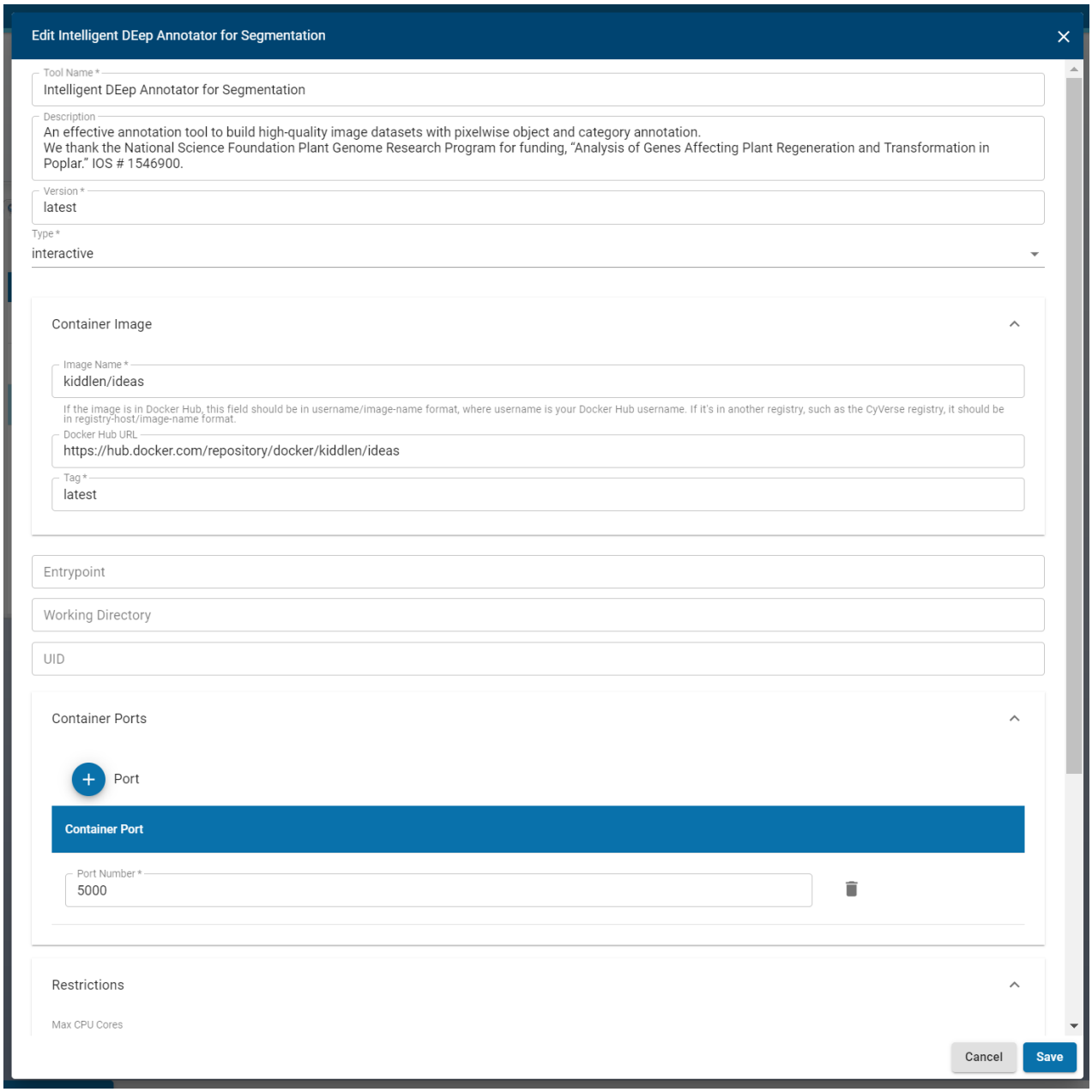
```
            $("#add").click();
          }
        }
      });
    }
    else{
      var currentInput = document.getElementById('hiename');
      currentInput.focus();
      $('#hiename').keypress(function(e){
        if (e.which == 13){
          if(currentInput.value.length > 0){
            $("#addHie").click();
          }
        }
      });
    }
}

const mutationObserver = new MutationObserver(callback);
mutationObserver.observe(
  document.getElementById("class-object-selector").querySelectorAll('li')[0],
  { attributes: true }
);
```

.2 Miscellaneous

You can find the project at https://de.cyverse.org/de/ listed as IDEAS in Apps by user kiddlen.

The docker image is listed at https://hub.docker.com/r/kiddlen/ideas and the github repository for building the docker container is at https://github.com/ngkiddle/image-annotator-docker



Inputs necessary to create a new CyVerse Tool for the Image Annotator.

.3 Code Review

.3.1 Reviewer 1

Category	Reviewer's Comment	Action Taken by Reviewed Group
Build	I can get accessible to the GitHub. There is a readme.md file. Readme file is good to read and follow. I download all files, but I failed to compile.	This failure may have been do to a hardware incompatibility, a NVIDIA GPU is needed. Reviewer did not specify…
Legibility	The code is modular, clean, easy to read, and has no redundancy. The structure between files is also good. I reviewed you code this afternoon, even though you explained your code this morning, I can't understand your code 100 percent. However, I still think your code is concise and good quality so far, because I am not developer and I think these functions are not complex and easy understood.	Thanks! Glad you think it is good.
Implementation	I checked the implementation video. I think your guys have good implementation.	Thanks!
Maintainability	There is one thing you can do is to make your code more understood is giving function information before your functions. I review your code; your team have comments in the code. If you guys can have function details comments that would be very helpful to read. Tell us how this function works for whole project, and that helps group members to maintain and us to understand your structure.	I think you are totally right. This project is very large though and I am just a single person group. I did not author much of this so I do not have the time nor expertise to go through all of the functions and meet my requirements. I will add comment headers to the functions that I worked on though.
Requirements	I checked the requirements document. I think you guys have met the requirements.	Great!

.3.2 Reviewer 2

Category	Reviewers Comment	Actions Take by Reviewed Group
Build	I was able to clone the repo and build from the README. If possible, I suggest using a shell script to set up the environment to simplify the installation process.	Good point. But actually the docker container is now available on dockerhub and that has the entire environment built in! Just requires docker and nvidia-docker to be installed. Then just a simple docker run will pull and run the latest version.
Legibility	The code for this project flowed as expected. Variable names represented their content and function names represented their use case. No suggestions necessary.	Thanks!
Implementation	I lack sufficient knowledge of the technology to improve upon the code base.	Fair enough.
Maintainability	There are no unit tests that I can find. Several functions could be tested using mock data, for example: **init mask from points** function returns a mask that looks like a numpy array. This function could be tested by comparing its output with a mock array checking for equality. I suggest using python's unittest module since it is fairly easy to understand.	The numpy array that is output is pixel data. It would be near impossible to create dummy data to pass in. Let alone predict what the machine vision algorithm will identify as the object. This would end up being a project in of itself. Visual testing is going to work better for this.
Requirements	Yes the requirements are being fulfilled	Cool!
Other	Other then testing and improving the install experience, no I think group 22 did a great job!	Concern addressed above ^^ and thanks!

.3.3 Reviewer 3

Category	Reviewers Comment	Action Taken by Reviewed Group
Build	I did manage to get the project running eventually but it was a major timesink. I had to spend over an hour troubleshooting to get it working. The instructions were pretty clear I think but it's still a relatively complicated process to get started, not sure how this could be improved but if it could it would be very welcome. I initially thought I couldn't build it for example because the instructions linked to the Ubuntu distribution only. Some of the python dependencies weren't mentioned as well. Combining the two repos into 1, if possible, would be good as well. For parsing the actual code of the project, I found the second README from the bitbucket repo to be more useful, but I'm also not 100% clear on what was expected from group 22 based on the information provided and a lot of the information here might have been outside the scope of the project.	As mentioned in the github readme only the first readme needed to be executed. The BitBucket readme did not need to be done. We simply pull our code from there to later stuff into a docker container. So actually no python dependencies needed to be installed. Docker did this for us. The repos are separate since they perform two different tasks. The code in BitBucket is the main project code but we want to be able to separate it from docker since the client does not want to always use docker to run the project. Yes the second readme is just about the project code. The first readme is about building the docker image that contains the project code.
Legibility	The legibility here was subpar in my opinion, mostly just due to the lack of modularization of the code. The Annotator class on it's own spans over 4500 lines of code.	Yes, unfortunately it was like this when I joined. To refactor this it would take one person the a whole year likely and I am just a one person team. This was also outside of the project requirements.
Implementation	In my opinion, this code definitely needs to be modularized more. This falls into the pitfall a lot of OOP languages fall into where you have this giant master class that has become incredibly bloated over time, making it hard to read and making it much easier to just add new functions to it rather than try to sort it out, causing it to grow more and more.	I agree. I will pass this on to the client as a potential topic to work on for the next year. The above comment applies here too ^
Maintainability	There aren't any unit tests I can see and I'm honestly not sure what you would even test for this, without using some kind of default 'test' image that produces consistent results (if that's even possible?).	Yeah a default image would still potentially give varying results because of the machine learning algorithm changes often and is not very predictable. An unfortunate pitfall of working with a black box.
Requirements	The requirements weren't clear to me on what the user was expected to complete for the project. The rectangle tool worked, which he mentioned working on, and the docker environment worked after a significant amount of tweaking.	Not sure what tweaking was required. But the user did try to use the second readme which was not required per the first github readme linked. This is now all built into a public docker container anyways, so building is only 3 steps.
Other	Really the readme workflow and code refactor seem like the biggest things to me.	Addressed in above comments.

.3.4 Reviewer 4

Category	Reviewers Comment	Action Taken by Reviewed Group
Build	I was not able to build the program as it is designed for use on a Linux system, and even if I could have built it I would not have been able to run the code as I do not have the proper GPU. Because of this, my evaluation for this section is based on just reading through the build instructions. The Readme in the Docker repo is well detailed with installation, build, and run instructions for setting up and using the container in conjunction with the larger project code on BitBucket. I do believe if I had the proper OS and hardware I would be able to successfully build the project based on the instructions. Running it is a bigger question as, compared to the Docker file documentation, the readme for the annotator repo is less detailed than it could be, but I am unsure how much of that repo was part of the project or if only the Docker work was, in which case it is unfair to grade the student based on shortcomings in the other code.	No problem. Yes the readme in Bitbucket has since been updated to be more clear. However, the BitBucket readme install instructions are not required to be completed. Docker compiles the whole requirements for us.
Legibility	The Docker file is well commented, making it clear what each part of the configuration is for and how it runs. The annotator file is cohesive in style, and fairly well commented. I would not say the code is easy to follow, but not due to poor design only because I am not familiar enough with this kind of programing to have a clear understanding of what it is all doing. There is clearly a logic to the design, I can see that the components work together, simply not the details of how. As before, judging the capstone work based on the entirety of the massive annotator file would not be right because the student only contributed some of this code. Viewing the commit history and focusing in on the contributions he made, specifically the implementation of the boundary box, the code is clean and fairly easy to follow.	Thanks! I did not design the layout of the annotator file. It certainly should be refactored, but this would be a project in of itself let alone for a one person team. This also was not part of the requirements. Thank you, glad it was readable.
Implementation	Based on the demo video that was provided in the event the code could not be run first hand, efficiency was already improved from how the project was initially received by adding the boundary box in the annotator to only have to process pixels that are of interest. Being unfamiliar with the inner workings of this type of program, I am unable to see any obvious further improvements that could be made beyond this.	Cool!
Maintainability	There are not defined unit tests, but I don't know that there should be for the specific work involved in the	

	capstone project. Unit tests for the annotator would be nice to confirm its functionality, but that seems outside the scope of this project if the student was only responsible for implementing some specific features in the annotator and then wrapping it in the Docker container. The test for that work is simply that it runs.	Yes, I agree. The annotator is difficult to test. It would take a long time to get a suite setup.
Requirements	Yes; the student was not responsible for the entire annotator, they were asked to add the user experience improvement and the efficiency improvement and create a Docker container for running the application, all of which were accomplished.	Great!
Other	The annotator file is so extensive that it may be easier for someone to understand the code if it was modularized a bit; though I don't know how feasible that is for how everything works together, and it would be out of the scope of this student's project to assign such a task to him, but it could be a note for the future.	Yes, outside of the scope of the project and very difficult to do as a single person unfamiliar with all the code. This would take the whole year.

.3.5 Reviewer 5

Build	No issues building	Glad to hear.
Legibility	Each function is aptly named with their names describing what they handle (e.g. Point, Label, or Annotator ▪ Scrolling through >1000 lines starts to hinder clarity ▪ Each frontend method is of a good length (i.e. not too long and not too short). It is easy to see what the programmer was trying to do. • I would have personally preferred to have everything declared/initiated at the top of each method if possible so that we can see everything being declared at once instead of having to scroll through. Each line is clear and whitespace is used gratuitously. This makes it each to break off code into sections and makes it easier on the eyes. • Helps with logical breaks as well Indentation is great and consistent. ▪ Echoing what I said earlier, I would have personally preferred to have everything declared/initiated at the top of each method if possible. ▪ Coding style is consistent across all of the student's work. ▪ Could clean up unused code ▪ Use ===/!== to compare with true/false or	Yes, I agree. Refactor would become a project itself though. Great! Fair enough. To be honest though I don't think there is much time for this to be implemented. The project has already been posted online and there are still other requirements that need to be addressed. Doing this would sacrifice completing the requirements. I will note it to the client. Nice. Nice Addressed above ^ Nice Has been done since review. Will do thanks for point that out.
Implementation	Maybe implement dragging image in addition to scroll bars. There might be a library out there that will help you out. o Bounding box is pretty, but I would recommend against using a gradient. Maybe switch to a color that catches the eye (like bright red) or just makes you want to gouge your eyes out. o Move position of elements	Hmm good idea. Would be very complicated though since we already have so many interactions woven into the image canvas. We would need to add a tool for that and then it would defeat the purpose of being toolless and add clicks… Multiple colors prevents the tool from becoming invisible in low contrast situations. A red box would be invisible against a red background. I'll look into this.

	▪ Clear buttons closer to the tools themselves ▪ Remove padding between preview window and "toolbar" o Make the toolbar buttons look like buttons o I'd suggest changing == to === when doing comparisons to numbers. This happens in multiple places. You can probably easily find these if you just do a search all for == o Split up the annotation js file into multiple helper files. 4000 lines is too much and would make it really hard to maintain. Maybe something someone next year could pick up. o Clean up training whitespace where necessary. (VS Code makes this really easy)	Good idea, look into this too. Done in the most recent update! Addressed above ^^^! This refactor would be impossible in the time we have left and for just one person. I will pass it on to the client. Not sure what training whitespace is. But the you mention the indentation is good earlier.
Maintainability	▪ No tests were found. ▪ Maybe use cypress for UI testing ▪ Mocha or Jest for other types of testing. ▪ Lots of commented out code. I would try to clean that up a bit. ▪ Aligned comments are really nice. Much easier to read. ▪ Many comments explain what each variable represents. • Only problem I can see is that they are all declared in the same spot (which is good) but since there are so many, it is a bit daunting and can be potentially hard to read. If you must keep it in the format that the declarations are in at the moment (see function Annotator()), consider adding whitespace before each comment (break) to categorize each of the variables. ▪ Repo contains readme of how to install and deploy ▪ Contains link on how to use the tool ▪ Markdown file could use some fixing ▪ Could potentially use readmes.md in each of the subfolders as well just giving a high-level description and view of what everything does in each of the respective subdirectories.	There is not enough time left for me to implement testing. I am still working on the requirements set by the client. I will suggest these to the client. Has since been cleaned up. Will do if there is time after the requirements are completed since this is a little nitpicky and a personal preference of theirs. Nice. Okay. Not a bad idea will do if there is time since this is a little nitpicky and a personal preference of theirs. Yep Yep

		Not sure which markdown file this is talking about or why it needs fixing. The BitBucket readme has since been updated. The readme in BitBucker has file descriptions in it to describe the more complicated and important files.
Requirements	Section not provided.	
Other	Section not provided.	

REFERENCES

[1] "Amazon ec2 features." Amazon, 2019. [Online]. Available: https://aws.amazon.com/ec2/features/

[2] "Nvidia tesla v100 data center gpu." NVIDIA, 2019. [Online]. Available: https://www.nvidia.com/en-us/data-center/tesla-v100/

[3] "Azure app service documentation - tutorials, api reference." Microsoft, 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/app-service/

[4] "Gpu optimized virtual machine sizes." Microsoft, 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes-gpu

[5] "The project." CyVerse, 2019. [Online]. Available: https://www.cyverse.org/about

[6] "Lightweight responsive css grid system - skel.js." Jquery script.net, 2013. [Online]. Available: https://www.jqueryscript.net/layout/Lightweight-Responsive-CSS-Grid-System-skel-js.html

[7] "React, a javascript library for building user interfaces." Facebook, 2019. [Online]. Available: https://reactjs.org/

[8] H. Mahmood, "Advantages of developing modern web apps with react.js." Medium, 2018. [Online]. Available: https://medium.com/@hamzamahmood/advantages-of-developing-modern-web-apps-with-react-js-8504c571db71

[9] C. Ferdinandi, "Why do people choose frameworks over vanilla js?" Go Make Things, 2019. [Online]. Available: https://gomakethings.com/why-do-people-choose-frameworks-over-vanilla-js/

[10] "Handling keyboard shortcuts in javascript." OpenJS, 2010. [Online]. Available: http://www.openjs.com/scripts/events/keyboard_shortcuts/