



# **Software Design Specifications Document**

**Dannosource**

*By*

Ehtisham Ali      297363

Syed Ali Jaseem      321917

Supervisor:

Dr. Hasan Ali Khattak

*Co-Supervisor :*

Dr. Bilal Ali

Dr. Zuhair Zafar

**Bachelor of Engineering in Software Engineering (2019-2023)**

Department of Computing

School of Electrical Engineering and Computer Science

National University of Sciences & Technology

# Table of Contents

Dannosource

Table of Contents

List of Sequence Diagrams

1. Introduction	<b>1</b>
1.1 Purpose	1
1.2 Scope	1
1.3 System Overview	1
2. Design Methodology and Software Process Model	<b>2</b>
2.1 Design Methodology	2
2.2 Process Model	2
3. Architecture	<b>3</b>
3.1 Overview	3
3.2 Architectural Design Strategy	3
3.3 Components Description	4
4. Design Models	<b>6</b>
5. Data Design	<b>11</b>
5.1 Data Dictionary	12
6. User Interface Design	<b>13</b>
6.1 Login/Sign Up	13
6.2 Home Page/Dashboard	14
6.3 Job Search Page	15
6.4 Annotation Workbench	16

## **List of Sequence Diagrams**

Sequence Diagram of the complete generic workflow	6
Annotating an Image Sequence	7
Reviewing an Annotation Sequence	8
Assigning a task Sequence	9
Exporting annotations Sequence	10

**Revision History**

Name	Date	Reason for changes	Version

## Application Evaluation History

Comments (by committee) *include the ones given at scope time both in doc and presentation	Action Taken

**Supervised by**

**Supervisor's Name**

Signature\_\_\_\_\_

# **1. Introduction**

## **1.1 Purpose**

The purpose of the Software Design Specification (SDS) document is to detail the technical aspects of the image annotation platform project, as specified in the previously created Requirements Specification (SRS) document. This document outlines the system architecture and infrastructure, serving as a guide for the development team to build a robust and efficient solution. The SDS builds upon the SRS to provide a clear understanding of the project's technical specifications and design decisions.

## **1.2 Scope**

The scope of the image annotation platform project is to build a web-based platform for image annotation, where users can upload their datasets and freelancers can apply for annotation jobs. The platform will be built using the MERN stack. The cloud storage for the image datasets will be provided by DigitalOcean. The platform will have the following features:

- Login/Registration
- Dashboard
- Image annotation tool
- User profiles
- Job postings and applications
- Payment system

The platform will be designed to ensure that the user data and image datasets are secure and protected. The platform will have a robust authentication and authorization system to ensure that only authorized users have access to the platform and its features.

## **1.3 System Overview**

The project is a web-based solution that provides a tool for image annotation required in the field of machine learning and artificial intelligence. The platform will have a user-friendly interface for users to upload their image datasets and for freelancers to apply for annotation jobs. The platform will be built using the MERN stack. The cloud storage for the image datasets will be provided by DigitalOcean.

The platform will have a variety of features to make the image annotation process easy and efficient for both users and freelancers. Users will be able to create an account, post jobs, and manage their datasets. Freelancers will be able to create a profile, apply for jobs, and use the annotation tool to annotate images. The platform will also include a payment system to allow users to pay freelancers for their work.

In summary, Dannosource is a comprehensive solution that provides a tool for image annotation, a platform for users and freelancers to connect and collaborate, and a secure and efficient system for storing and managing data. The platform will be built using the latest technology and design principles to ensure that it is user-friendly, scalable, and robust.

## **2. Design Methodology and Software Process Model**

### **2.1 Design Methodology**

The Dannosource project is being developed using the MERN stack, with the front-end utilizing ReactJS. Currently, the front-end is following the Procedural paradigm, utilizing functional components instead of class-based components. This approach has been chosen to keep the code organized and maintain its readability, as functional components are a simpler and more straightforward way to build UI components in ReactJS.

As for the back end, the logic and implementation are yet to be determined. It will be decided based on the requirements and what paradigm suits best. The server-side logic could be implemented using either Object-Oriented Programming (OOP) or Procedural programming, whichever suits the needs of the project. The backend code will interact with the database, manage the data, and handle business logic. The choice of paradigm will depend on the complexity of the business logic and data management, as well as the team's preference and expertise.

### **2.2 Process Model**

The software development process model being followed for the Dannosource project is the Agile/SCRUM model. The Agile methodology provides a flexible and adaptive approach to software development, where requirements and solutions evolve through collaboration between self-organizing and cross-functional teams. SCRUM is one of the most widely used Agile frameworks, and it provides a set of practices and roles for managing the product development process. This model is suitable for the two-person development team of the Dannosource project, as it prioritizes regular and rapid iteration, collaboration, and transparency in order to deliver a high-quality product that meets the evolving needs of the stakeholders.

### 3. Architecture

#### 3.1 Overview

This project follows a 3-tier architecture pattern, with the presentation layer as the client-side ReactJS components, the application layer as the NodeJS server with ExpressJS handling the routes, and business logic and the database layer consisting of MongoDB/Firebase storing the data.

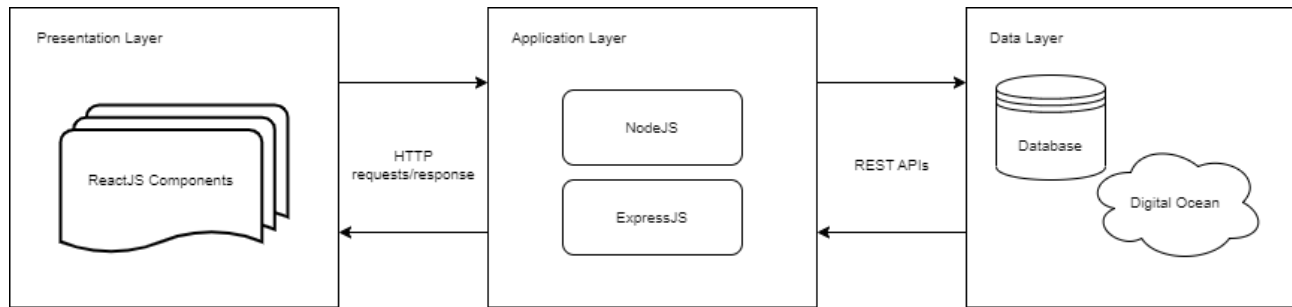


Figure 1 3-tier architecture of the app

- **Presentation layer:** Handles the user interface and user interactions, built using React.
- **Application layer:** Handles the business logic and communication with the backend, built using Node.js and Express.
- **Database layer:** Stores the data and serves it to the application layer, using a cloud-based database solution.

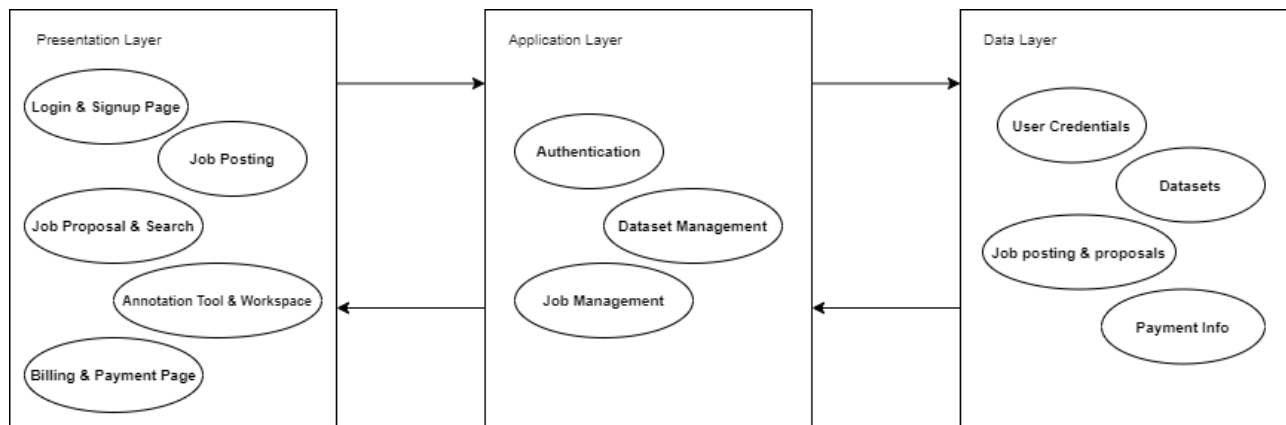


Figure 2 Architectural design decomposed in components.

#### 3.2 Architectural Design Strategy

This project is best suited for the Microservices Architecture pattern. The project involves multiple independent functionalities such as Authentication, Datasets Management, Job Posting, Job Assignment, Job Search and Proposal, Annotation Workspace, Job Management, and Billing & Payment. Each of these functionalities can be treated as a separate microservice, thus improving the maintainability, scalability, and overall system reliability. Additionally, microservices architecture can also allow you to use the best technology stack for each service, making it easier to handle future changes and upgrades.



### 3.3 Components Description

Authentication Component	
<b>Purpose</b>	To allow users to sign up and create a new account.
<b>Responsibility</b>	To validate the user's input for unique username, email, password, and then re-enter the password.
<b>Input</b>	Unique username, email, password, and password confirmation.
<b>Output</b>	Successful account creation

Datasets Management Component	
<b>Purpose</b>	To manage the datasets uploaded by the assignor.
<b>Responsibility</b>	To store the dataset in the cloud storage affiliated with the user.
<b>Input</b>	Datasets uploaded by the assignor.
<b>Output</b>	The annotated dataset with annotations saved, which can be exported in the desired format by the assignor.

Job Posting Component	
<b>Purpose</b>	To enable users to create new jobs.
<b>Responsibility</b>	To redirect the user to a page where they can enter details about the job and link the dataset.
<b>Input</b>	Details about the job and dataset.
<b>Output</b>	New job creation.

Job Assigning Component	
<b>Purpose</b>	To enable the assignor to assign annotators to their dataset.
<b>Responsibility</b>	To accept proposals from annotators and select the batch size for each annotator.
<b>Input</b>	Proposals from annotators
<b>Output</b>	Annotators assigned to the dataset.

Job Search and Proposal Component	
<b>Purpose</b>	To allow users to search for jobs and send job proposals.
<b>Responsibility</b>	To enable users to search for jobs and send job proposals to the client.
<b>Input</b>	Search criteria and job proposal.
<b>Output</b>	Matching jobs and sent job proposals.

Annotation Workspace Component	
<b>Purpose</b>	To allow annotators to annotate data.
<b>Responsibility</b>	To provide assisting tools for annotation and review the annotated data.
<b>Input</b>	Dataset for annotation, annotation type, and annotation classes.
<b>Output</b>	Annotated data.

Job Management Component	
<b>Purpose</b>	To manage the annotation work.
<b>Responsibility</b>	To review the annotated data, accept or reject the work and provide remarks.
<b>Input</b>	Annotated data.
<b>Output</b>	Reviewed annotated data.

Billing and Payment Component	
<b>Purpose</b>	To manage billing and payments.
<b>Responsibility</b>	To redirect the user to the payment page, confirm billing details and pay the promised fee.
<b>Input</b>	Accepted annotated data.
<b>Output</b>	Payment made to the freelancer.



**Annotating Images Sequence:** This sequence diagram shows the interaction between the annotator, the annotating tool, and the image being annotated. The annotator first selects an image to annotate. Then, they select the annotating tool they want to use. Next, they add annotations to the image using the selected tool. Finally, they save the annotated image. This sequence demonstrates the steps involved in the annotating process.

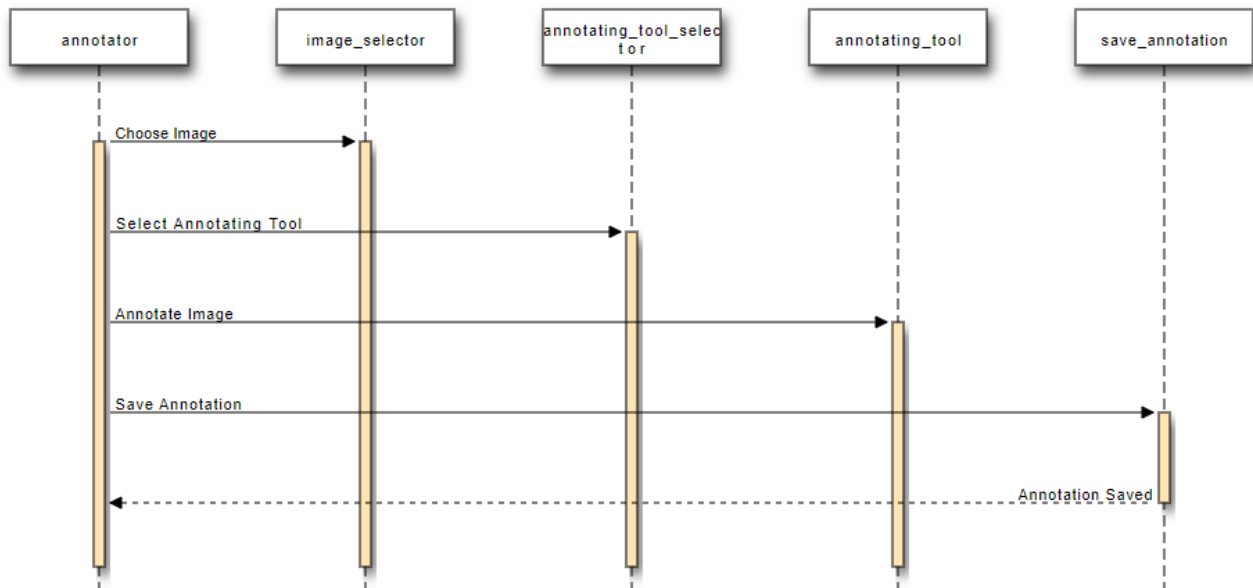


Figure 4 Annotating an Image Sequence

**Approval/Review Sequence:** This sequence diagram illustrates the interaction between the assignee, the supervisor, and the annotating tool. The assignee starts by annotating an image. Then, the supervisor reviews the annotations made by the assignee. Finally, the supervisor approves or rejects the annotations. This sequence demonstrates the steps involved in the review and approval process of the annotations.

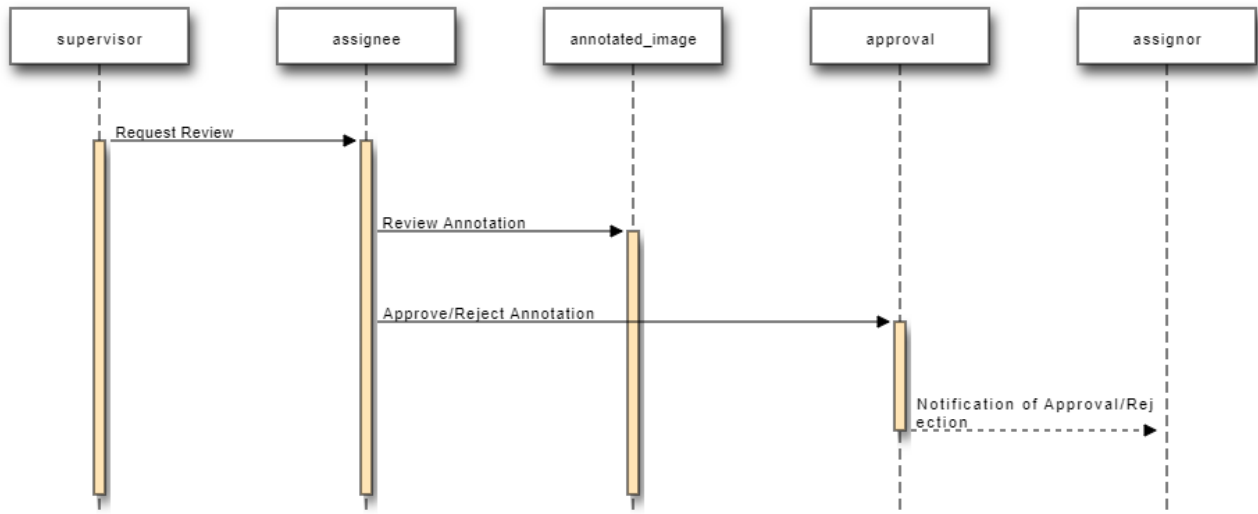


Figure 5 Reviewing an Annotation Sequence

**Assigning Annotation Task Sequence:** This sequence diagram highlights the interaction between the assignor, the annotator, and the annotating tool. The assignor begins by assigning an annotation task to the annotator. Then, the annotator accepts the task. Finally, the annotator starts the annotating process using the annotating tool. This sequence demonstrates the steps involved in assigning and accepting an annotation task.

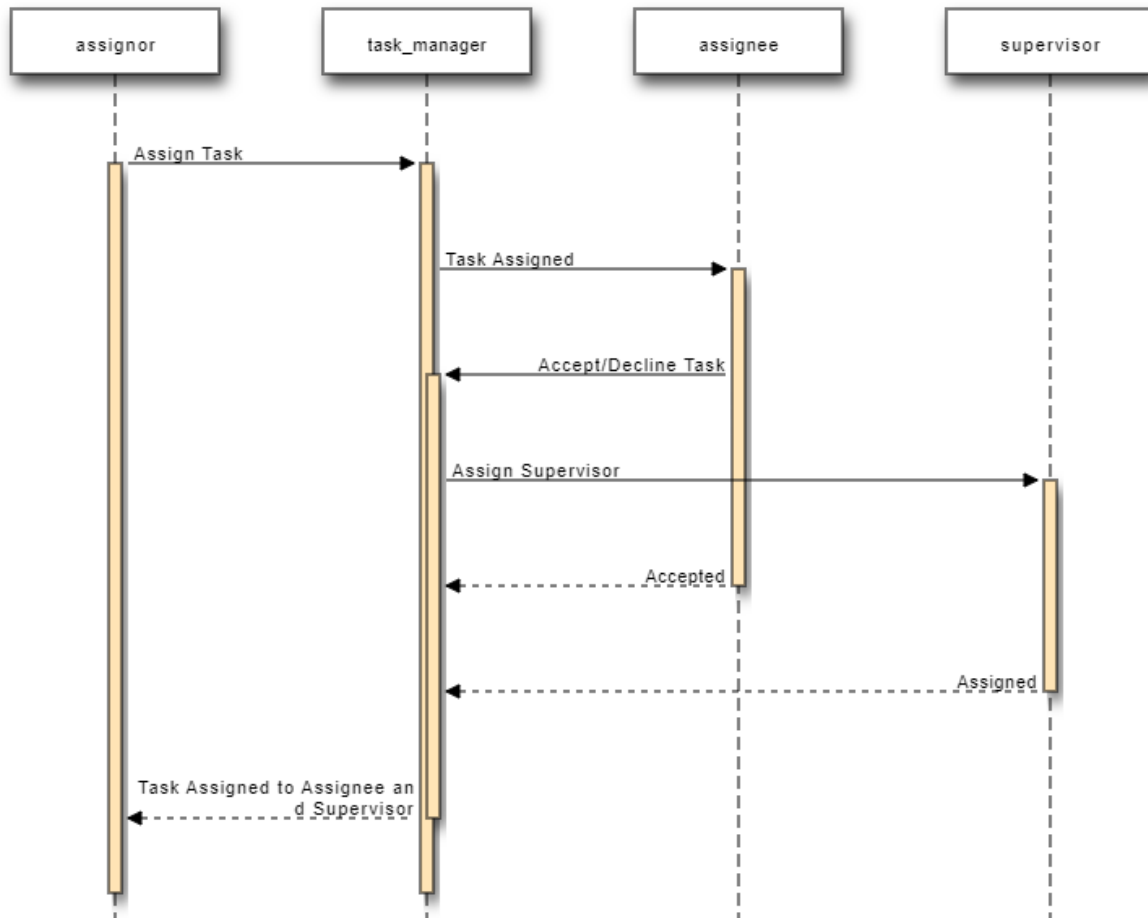


Figure 6 Assigning a task Sequence

**Exporting Annotated Data Sequence:** This sequence diagram illustrates the interaction between the assignor, the annotated data, and the data export tool. The assignor starts by selecting the annotated data to be exported. Then, they choose the format in which to export the data. Finally, they initiate the export process. This sequence demonstrates the steps involved in exporting annotated data.

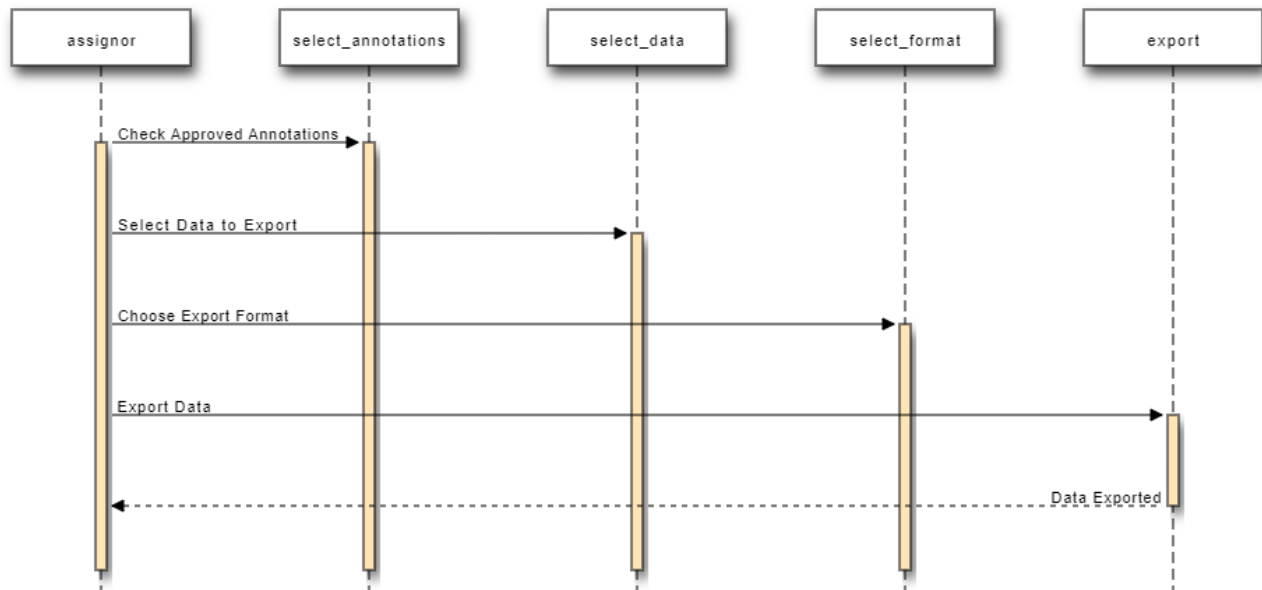


Figure 7 Exporting annotations Sequence

## 5. Data Design

The following Entity Relationship diagram depicts the high-level approach in defining and relating the information and its storage logic.

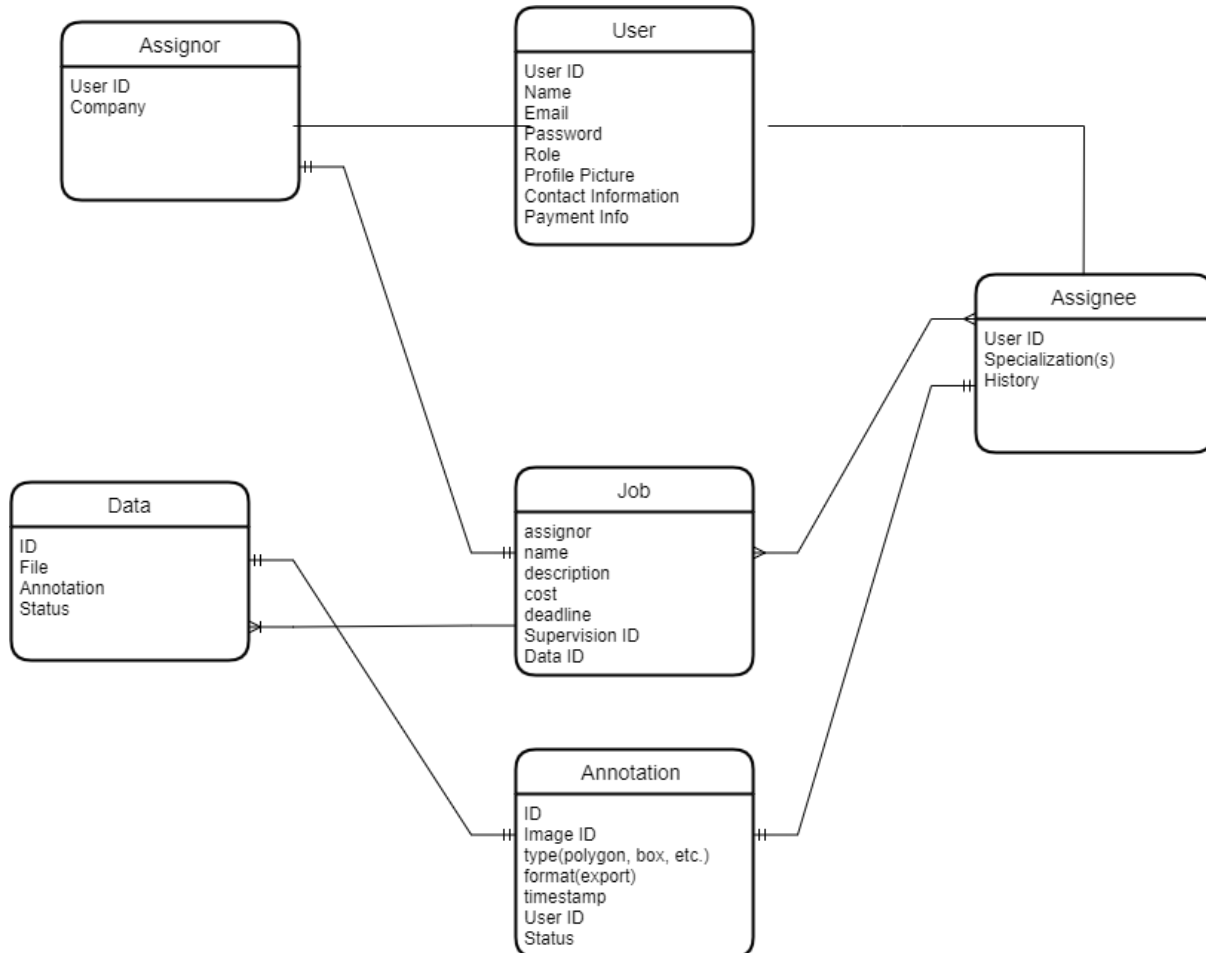


Figure 8 ER diagram for Dannosource



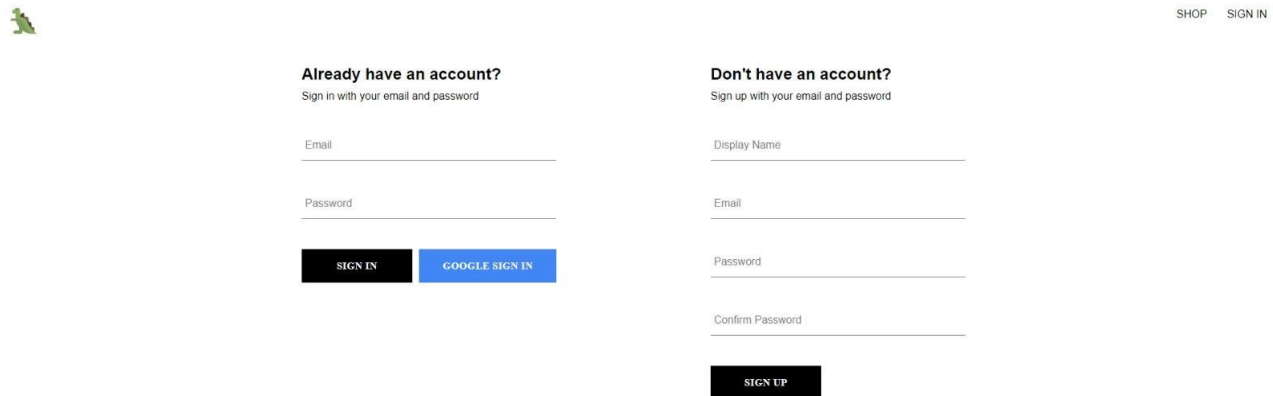
## 5.1 Data Dictionary

Entity Name	Attribute Name	Attribute Type	Description
User	ID	Integer	Unique identifier for the user
	Name	String	Name of the user
	Email	String	Email address of the user
	Password	String	Encrypted password for user authentication
	Role	Enum	Role of the user (Assignor, Assignee, or Supervisor)
	Specialization	String	Specialization of the user (for Assignee)
	Rating	Float	Rating of the user (for Assignee)
Annotation	ID	Integer	Unique identifier for the annotation
	Type	Enum	Type of annotation (Polygon, Boxplot, etc.)
	Image URL	String	URL of the image being annotated
	Format	Enum	Format of the exported annotation (JSON, XML, etc.)
	Annotator ID	Integer	ID of the user who made the annotation
	Annotation Data	JSON/XML	Data of the annotation in specified format

## 6. User Interface Design

The following mockup pages have been roughly drawn. The basic requirements and actions for each page are explicitly noted. The developer had decided earlier to design a complete working mockup on a prototyping tool such as Figma. But due to limited time and to avoid the learning curve of Figma, it is decided to directly start building in ReactJS with experimentation of tailwind CSS and other libraries to give elegant design in addition to delivering the necessary functionality. These pages are drawn below with explicit description:

### 6.1 Login/Sign Up



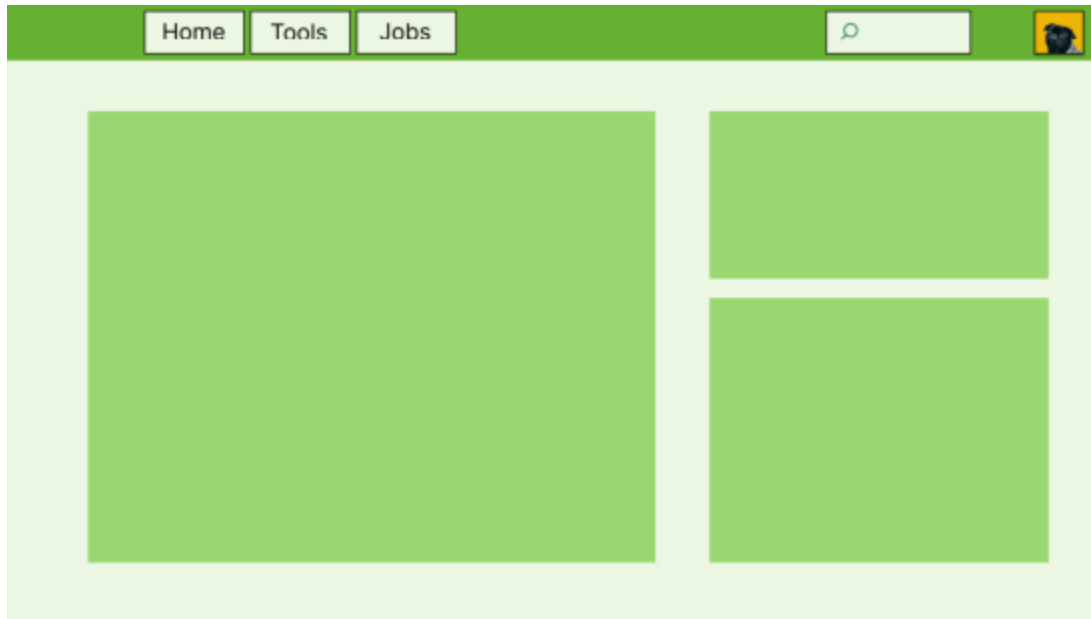
The mockup shows a clean, modern interface. The header is dark with a green dinosaur logo on the left and 'SHOP' and 'SIGN IN' links on the right. The main content area is white with a subtle shadow. The left column is titled 'Already have an account?' and 'Sign in with your email and password'. It has two input fields for 'Email' and 'Password', a black 'SIGN IN' button, and a blue 'GOOGLE SIGN IN' button. The right column is titled 'Don't have an account?' and 'Sign up with your email and password'. It has four input fields: 'Display Name', 'Email', 'Password', and 'Confirm Password', and a black 'SIGN UP' button.

*Figure 9 Implemented Sign Up page*

- A header with the logo of the app.
- A form in the center of the page with fields for entering the email address and password.
- A “Login” button below the form.
- A link to the “Forgot Password” page.

## 6.2 Home Page/Dashboard

Once the user logs in as an assignor, they will be redirected to the Assignor Dashboard where they can post a new job and manage existing ones. Once the user logs in as an assignee, they will be redirected to the Assignee Dashboard where they can see all the available jobs and apply for them.

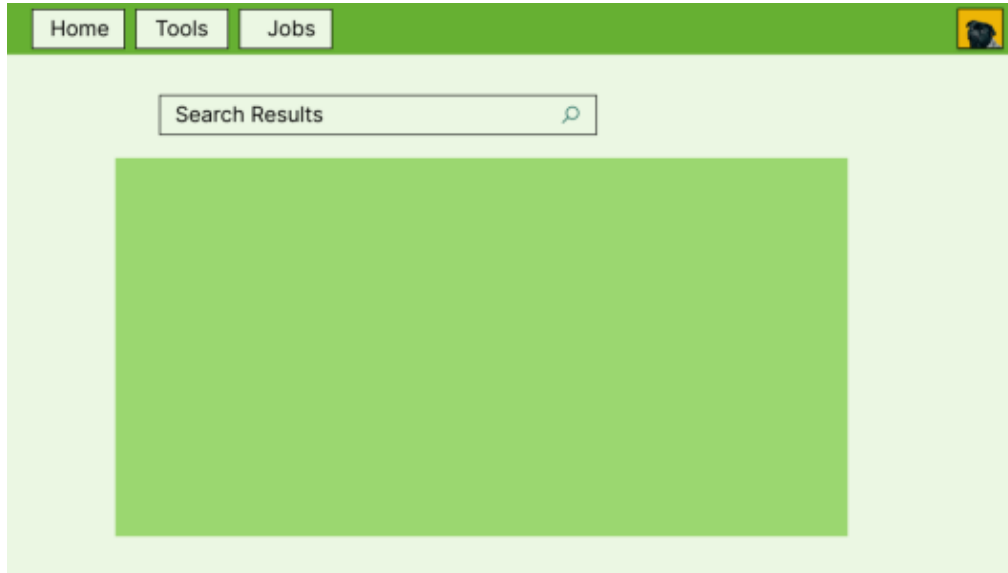


*Figure 10 Home Page MockUp*

- A header with the logo of the app.
- A menu with options for navigating to different parts of the app.
- A dashboard with an overview of the current status of jobs.
- A section for displaying recent activity.
- A footer with links to important pages.

### 6.3 Job Search Page

Once users enter a keyword into the search bar, displayed on the navbar, they will be redirected to a search page, that will the appropriate search results.

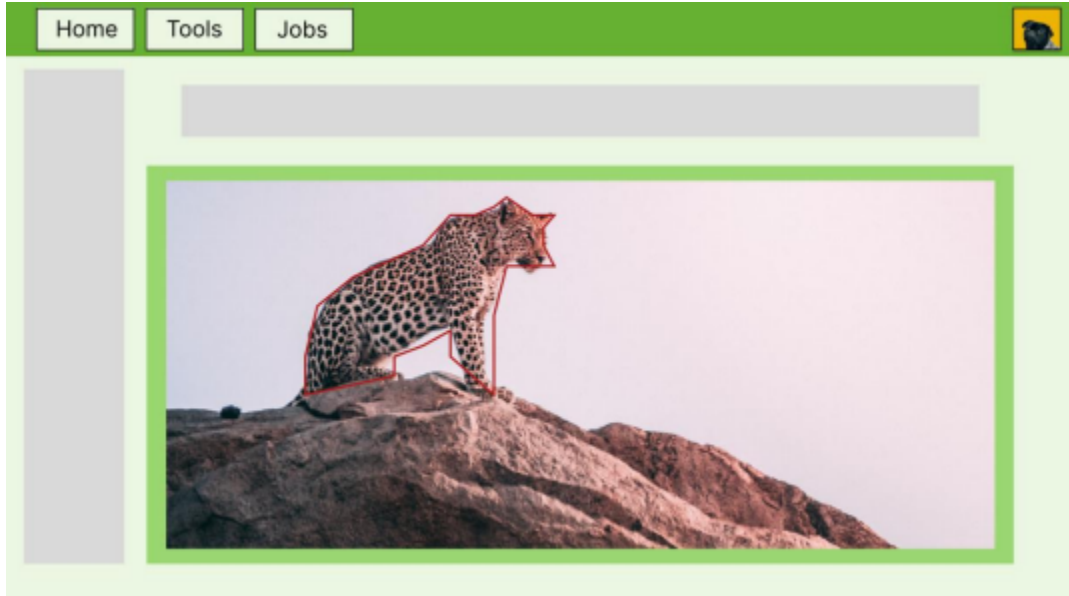


*Figure 11 Job Search Page Mockup*

- A horizontal navigation bar at the top displaying the logo of the application and a search bar.
- A vertical filter bar on the left side of the page.
- A main area in the center of the page where the horizontal cards of the jobs will be displayed in a list.

## 6.4 Annotation Workbench

The Annotation Workbench page is where assignees will be annotating the data assigned to them by the Assignor. This page should have a clean, intuitive, and functional design to facilitate the process.



*Figure 12 Annotation Workbench MockUp*

- The canvas should be the center point of the page, taking up a large portion of the screen.
- On the left side of the canvas, there should be a set of tools for the user to annotate the image, such as rectangles, polygon, and brush tools.
- On the right side of the canvas, there should be various buttons and features that aid in the annotation process.
- Next Image Button: A button that allows the user to navigate to the next image in the annotation project.
- Previous Image Button: A button that allows the user to navigate to the previous image in the project.
- Feedback Comment Section: A section where the assignor can leave feedback for the assignee regarding their annotations.
- Image Information: Information about the image being annotated, such as its file name and size.
- Annotation Save Button: A button that allows the user to save their annotations for the current image.
- Annotation Discard Button: A button that allows the user to discard their annotations for the current image.
- Zoom In and Zoom Out Buttons: Buttons that allow the user to zoom in and out on the image being annotated.
- Annotation Tools: A set of tools such as rectangles, polygons, and brush tools that allow the user to annotate the image.
- Image Thumbnail Navigation: A small strip of image thumbnails that allows the user to quickly navigate to a specific image in the project.