**Helwan University Image Annotation System (HIAS)**

**Product Requirements Document (PRD)**

**1. Project Overview**

This project is a web-based image annotation tool designed to facilitate efficient labeling of object classes within images. Inspired by LabelMe, the system will harness a simple and intuitive interface for annotators, aiming to support research in computer vision and machine learning.

**2. Purpose and Objectives**

* **Purpose**: To provide an efficient and scalable platform for annotating images with rich and precise object labels for computer vision datasets.
* **Objectives**:
  + Collect high-quality image annotations using an intuitive web interface.
  + Enable easy data access and instant sharing for research purposes.
  + Facilitate collaboration and ensure that annotations are scalable for large datasets.

**3. Target Audience**

* Computer vision researchers and machine learning practitioners who require annotated datasets for training and evaluation.
* Data scientists and engineers working on object detection and recognition models.

**4. Features and Requirements**

**4.1 User Management and Authentication**

* **Sign Up & Login**: Secure user registration and authentication, using JWTs for session management.
* **User Roles**:
  + **Annotators**: Users who can annotate images and manage their contributions.
  + **Admins**: Users who oversee data management, user roles, and platform configurations.

**4.2 Image Management**

* **Upload and Manage Images**:
  + Users can upload images (JPG, PNG) in bulk or individually.
  + **Image List View**: Display a grid or list of all uploaded images, showing annotation status.
* **Categorization and Organization**: Organize images into projects for structured management.
* **Image Viewer**: A robust viewer for loading images with essential features like zoom and pan.

**4.3 Annotation Tools**

* **Drawing Interface**:
  + **Polygon Tool**: Users can outline objects by clicking along the object's boundary to form a closed polygon, similar to the LabelMe interface.
  + **Bounding Box**: Draw rectangular boxes around objects as an alternative annotation method.
* **Annotation Workflow**:
  + **Label Selection**: A dropdown or text field for selecting or entering object class names.
  + **Undo/Redo**: Support for reversing or reapplying recent actions.
  + **Edit and Delete**: Ability to modify or remove existing annotations.

**4.4 AI-Powered Automatic Annotation**

* **Object Detection Model**: Use a pre-trained AI model (e.g., YOLO or Faster R-CNN) to suggest annotations.
* **User Review**: Annotators can accept, modify, or discard AI suggestions.
* **Training Data**: The system can improve over time by retraining on corrected annotations (future enhancement).

**4.5 Annotation Management**

* **Save Annotations**: Annotations are saved in real-time or when users confirm their work.
* **Load and Display**: Retrieve and render annotations when an image is loaded for viewing or editing.
* **Export Annotations**: Users can download annotations in JSON or XML format, making them easy to use in machine learning workflows.
* **Annotation Quality Check**: Admins can review and validate annotations for consistency and accuracy.

**4.6 Data Quality and Usability**

* **Quality Control**: Implement checks to ensure annotations are complete and accurate, potentially using review mechanisms where admins or other users verify the quality.
* **Guidelines and Tutorials**: Provide new annotators with instructions or video tutorials to improve data consistency.

**4.7 Performance and Scalability**

* **Efficient Image Loading**: Optimize the backend for handling high-resolution images with minimal latency.
* **Scalability**: The system should handle hundreds of thousands of images and annotations, inspired by the growth model of the LabelMe database.

**4.7 Collaboration and Real-Time Updates**

* **Collaborative Annotation Projects**: Users can work together on projects, with annotations updated in real-time using WebSockets or a similar technology.
* **Activity Tracking**: Keep logs of who annotated what and provide an activity dashboard for admins.

**5. Non-Functional Requirements**

* **Security**: Protect user data and images using encryption and secure authentication methods.
* **Usability**: Design the interface to be user-friendly, ensuring even novice annotators can use the tool effectively.
* **Reliability**: The tool should handle downtime gracefully and maintain high availability.
* **Data Privacy**: Follow best practices to protect and anonymize image data if needed.

**6. Technical Requirements**

**6.1 Frontend**

* **Framework**: React.js or Vue.js for a dynamic and responsive user interface.
* **Libraries**:
  + HTML5 <canvas> or an SVG library for drawing polygons.
  + Material-UI or Tailwind CSS for a clean, responsive design.

**6.2 Backend**

* **Framework**: Node.js (Express.js) or Django for a robust API.
* **Database**: PostgreSQL for structured data and a file storage system (like AWS S3) for images.
* **Image Processing**: Use cloud-based services or libraries like Sharp for image resizing and optimization.

**6.3 APIs and Data Flow**

* **RESTful APIs**: For managing images, annotations, and user data.
* **WebSocket Integration**: For real-time updates if collaboration features are implemented.

**7. Milestones and Timeline**

**Month 1: Planning and Project Setup**

* Define project structure, set up frontend and backend frameworks, and implement user authentication.

**Month 2: Core Annotation Features**

* Develop image upload and management features.
* Build the annotation tools (polygon and bounding box) and set up data saving/loading mechanisms.

**Month 3: Usability and Export Options**

* Refine the UI for ease of use and add annotation export features.
* Implement quality control and review features for data validation.

**Month 4: Testing, Optimization, and Deployment**

* Test for performance and security.
* Optimize the backend for scalability and deploy the application on a cloud platform.

**8. Risks and Challenges**

* **User Engagement**: Ensuring users provide accurate and consistent annotations.
* **Performance Bottlenecks**: Handling large datasets and real-time features efficiently.
* **Security Concerns**: Protecting sensitive image data and user information.

**Mitigation Strategies**

* Conduct user testing to improve the interface.
* Optimize database queries and use caching where necessary.
* Use secure authentication and data encryption methods.