Real-time AI Object Detection Web App
Software Design Document

Version 1.0
7 March 2024

Rowan University
Bradley Gathers

Document Approvals

Prepared By:
Bradley Gathers
Senior Student
Department of Computer Science, Rowan University

Approved By:
TBD
TBD
TBD

Table of Contents

1. Introduction

The Real-time AI Object Detection Web App (RAODWA) is a Web App that will be an advanced real-time interface for Real-time artificial intelligence (AI) detectors. It will allow users to customize their real-time detections in real-time and adjust many tunable parameters and toggleable features. In addition to this, the app will display real-time metrics of the AI model back to the user. Supporting these features will be easy-to-understand documentation that aims to allow users to better understand what each feature, setting, and outputs do. The collection of these features will allow a user to gain a better understanding of the complex models used in real-time object detection and gain knowledge through experiential learning.

2. Purpose

The purpose of this document is to introduce specifications and define purpose, assumptions, scope, design, technical architecture, configurations, and responsibilities. It will outline the overall design of the Real-time AI Object Detection Web App in detail and explain in depth the technical decisions and architecture.

3. Assumptions and Principles

3.1 Assumptions

The document and product make the following assumptions

3.1.1 Project-Specific Assumptions

- When referring to “the model” or “the AI model”, it is assumed to be the YOLO Real-time Object Detection model.
- When referring to “the product”, it is assumed to be the RAODWA

3.1.2 Architecture Assumptions

- The product will follow a Model-View-Controller (MVC) architecture.
- It will be programmed using React(Javascript (JS) framework), specifically utilizing the NextJS framework for the frontend and a Python backend.
- Styling will primarily be done statically through CSS, with some dynamic CSS elements.
- For UI components, the JoyUI framework will be used alongside React.
- The product will also utilize the existing implementation of YOLOv8 (and possible future YOLO versions).
- The YOLO model implementations from the Ultralytics Python package.

3.1.3 Communication Assumptions

- Communication between the frontend and backend will occur via a WebSocket in combination with REST API endpoint(s).
- The WebSocket will be implemented using the SocketIO library in both JS and Python.

3.1.4 User Assumptions

- Users of the interface are assumed to have basic computing knowledge and a basic understanding of AI networks.
- Users are not expected to know how YOLO functions or any research that the AI model utilizes.
- The users are not expected to understand Object Detection and its workings.

3.1.5 Performance and Hardware Assumptions

- Users are assumed to have a good connection to the web applications.
- Sufficient hardware resources (such as a GPU) will be available to run object detection in real-time
- The web app will achieve a relatively real-time experience (with a rate of >= 15), allowing for 1-5 seconds of allowable delay due to potential internet complications.

3.2 Principles

Architecture and Designs will follow the “Keep it stupid simple” (KISS) methodology. This will aim to avoid unnecessary complexity. For performance, the product will prioritize speed over memory usage. The look and feel of the product will mimic the Material Design standards for UI design, with slight modifications to follow the feel of the JoyUI Framework being utilized. The Design of the framework will also follow component-based UI. This will allow each section to be built independently and break work into chunks.

4. Scope

4.1 Requirements Tracability Matrix

4.1.1 Component Definitions

Component	Component ID	Description
Main Page	M	The main page shows off a collection of panels which the user can interact with and see the models object detection results.
Webcam Panel	W	The panel which shows the model output stream to the user.
Information Panel	I	The panel shows off basic information metrics to the user.
Feature Settings Panel	F	The panel shows off settings inputs that enable features to the user.
Feature Specific Settings Panel	S	The panel shows off settings specific to features to the user.
Model Specific Settings	P	The panel shows off settings to configure the model to the user.
Top Prediction Panel	L	The panel shows off the top predictions to the user.

4.1.2 Functional Requirements

4.1.2.1 Main Page

Requirements ID	Statement	Component	Test Case
M.1	The main page shall be scalable and adhere to responsive CSS design, via no clipping and dynamic sizing of elements.	Main Page	T.M.1
M.1.1	The main page shall be responsive on mobile devices (Breakpoints at Phone, Ipad, Laptop, and 16:9.)	Main Page	T.M.1.1
M.2	The main page shall support dark and light modes.	Main Page	T.M.2
M.2.1	The main page shall allow the user to toggle between dark and light mode manually.	Main Page	T.M.2.1
M.2.2	The main page shall default to darkmode.	Main Page	T.M.2.2
M.4	The main page shall display popups to show user alerts and important status changes.	Main Page	T.M.4
M.4.1	Information popups shall clear themselves over time.	Main Page	T.M.4.1
M.4.2	Information popups shall be displayed in clearly defined levels (info, warning, success, error)	Main Page	T.M.4.2
M.4.3	There shall only be one information popup at one time (per Material Design standards)	Main Page	T.M.4.3

M.4.4	Popups shall clear out over time automatically.	Main Page	T.M.4.4
M.W	The main page shall display model output.	Main Page	T.M.W
M.I	The main page shall display inference and postprocessing outputs.	Main Page	T.M.I
M.F	The main page shall allow features to be enabled.	Main Page	T.M.F
M.S	The main page shall display feature specific settings.	Main Page	T.M.S
M.P	The main page shall display model specific settings.	Main Page	T.M.P
M.L	The main page shall display top predictions to the user.	Main Page	T.M.L
W.1	The webcam panel shall be updated in real-time (100-200ms)	Webcam Panel	T.W.1
W.2	The webcam panel shall display bounding boxes around objects.	Webcam Panel	T.W.2
W.3	The webcam panel shall display class and confidence around objects.	Webcam Panel	T.W.3
I.1	The information panel shall be updated in real-time.	Information Panel	T.I.1
I.2	The information panel shall display the inference time in ms and fps.	Information Panel	T.I.2
I.3	The information panel shall display post processing time in ms.	Information Panel	T.I.3

I.4	The information panel shall display the number of objects detected.	Information Panel	T.I.4
F.1	The feature settings panel shall provide the ability to enable features [Tracking, Pose Estimation, Heatmap, and Segmentation Maps]	Feature Settings Panel	T.F.1
S.F.1	The feature specific settings panel shall allow for customization of the enabled features	Feature Specific Settings Panel	T.S.F.1
P.1	The model settings panel shall allow user to change the source fed into the model.	Model Specific Settings Panel	T.P.1
P.2	The model settings panel shall allow user to change model size (Nano, Small, Medium, Large, Extra Large).	Model Specific Settings Panel	T.P.2
P.3	The model settings panel shall allow the user to change the confidence filter for detected objects.	Model Specific Settings Panel	T.P.3
P.4	The model settings panel shall allow the user to change the IOU (intersection over union) filter for detected objects.	Model Specific Settings Panel	T.P.4
P.5	The model settings panel shall allow users to filter which classes are detected.	Model Specific Settings Panel	T.P.5
L.1	The top prediction panel shall be updated in real-time	Top Prediction Panel	T.L.1
L.2	The top prediction panel shall allow users to view	Top Prediction Panel	T.L.2

	the top (to be resolved) number of predictions.		

4.2 Usage

4.2.1 Overview

The whole product is designed around the YOLOv8 Ultralytics model. Since it will be modifying the model and displaying the output, the most important components will be the Model Settings Panel and the Webcam Panel. When the user selects or inputs a model source, it will kick off a chain of events that causes updates to the Webcam Panel, Information Panel, and Top Prediction Panel. The User is the main actor, but there will be some cross-component actors as well to help trigger stateful behavior.

The Feature Settings Panel and Feature Specific Settings Panel are closely coupled. The Feature Settings Panel will be responsible for hiding and disabling specific features and their respective settings panel within the Feature Specific Settings Panel.

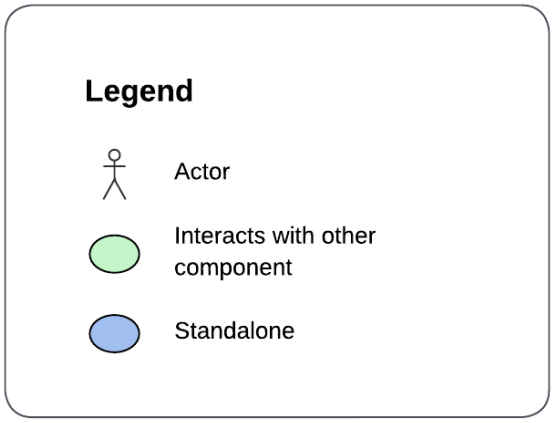
Additionally, most inputs and some of the outputs will have help buttons which allow for the user to click them and open up a specific documentation page that explains it. Documentation is only able to be triggered by the user actions.

4.2.2 User Roles

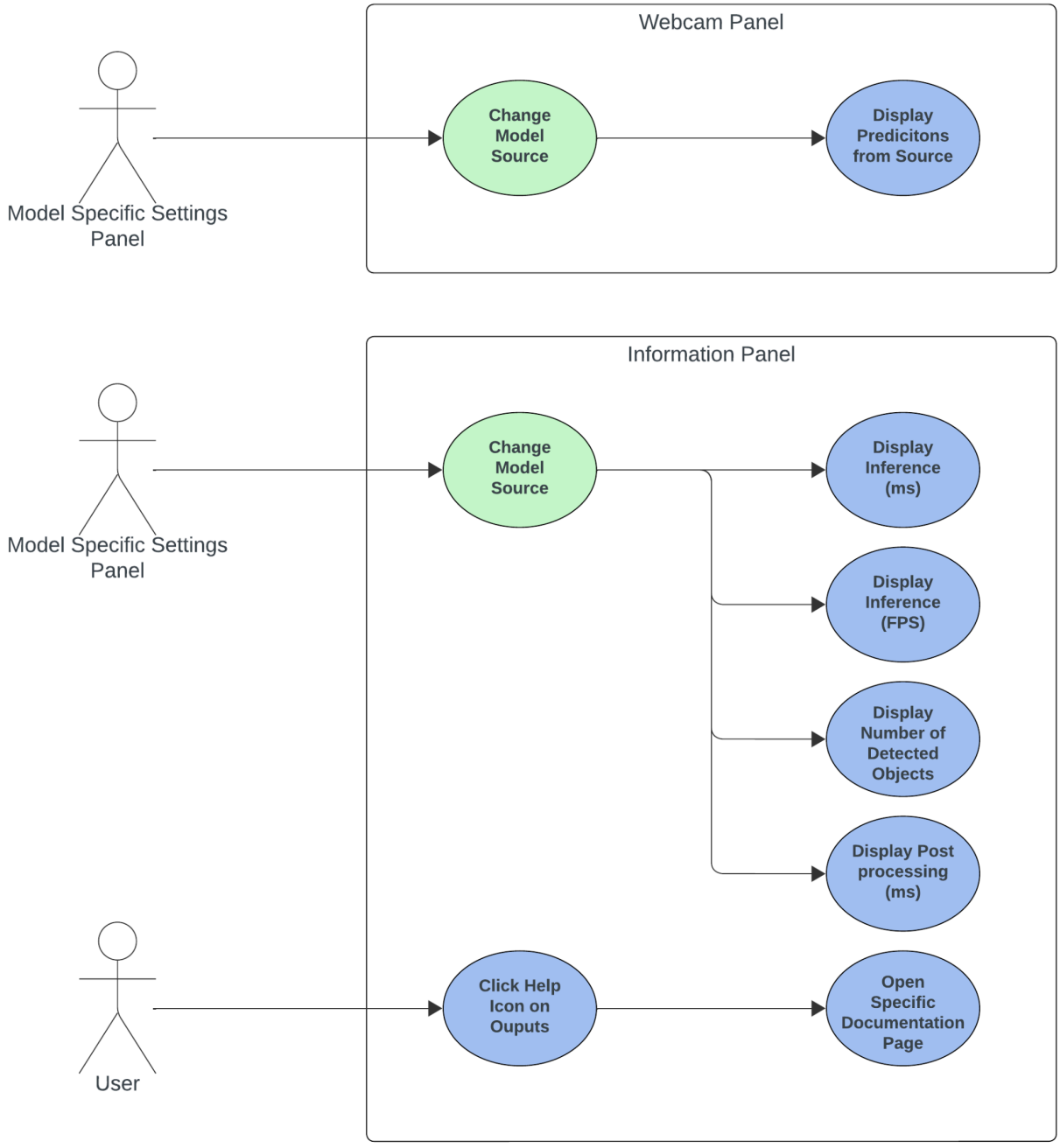
User	Role / Descruser'son
User	The intended user of the product. Will interact with the interface.
Feature Settings Panel	Will interact mainly with the Feature Specific Settings Panel and coordinate settings visibility.
Model Specific Panel	Will interact with other panels automatically via stateful changes to coordinate actions and display updates.

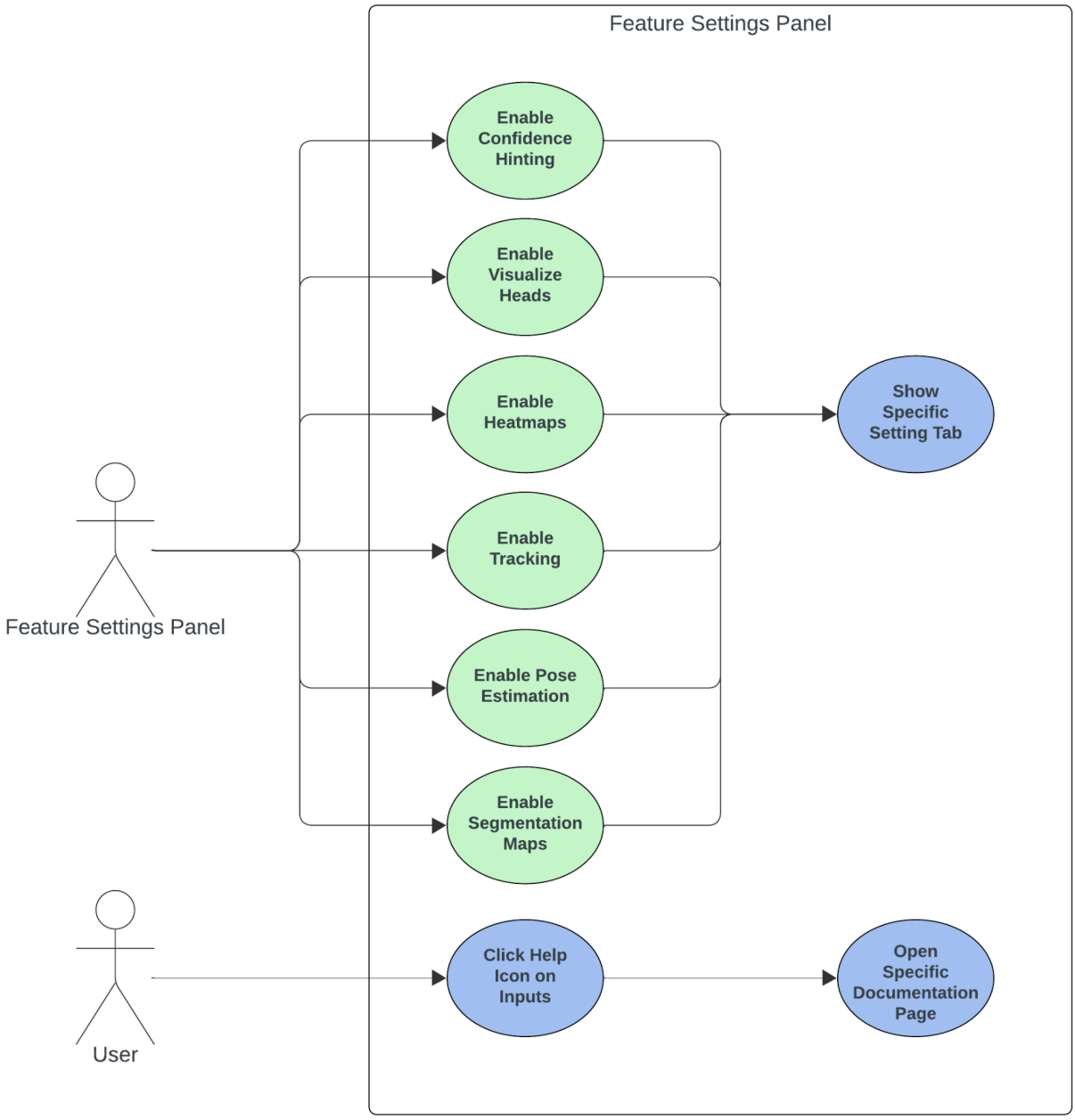
4.2.3 Activity Diagrams

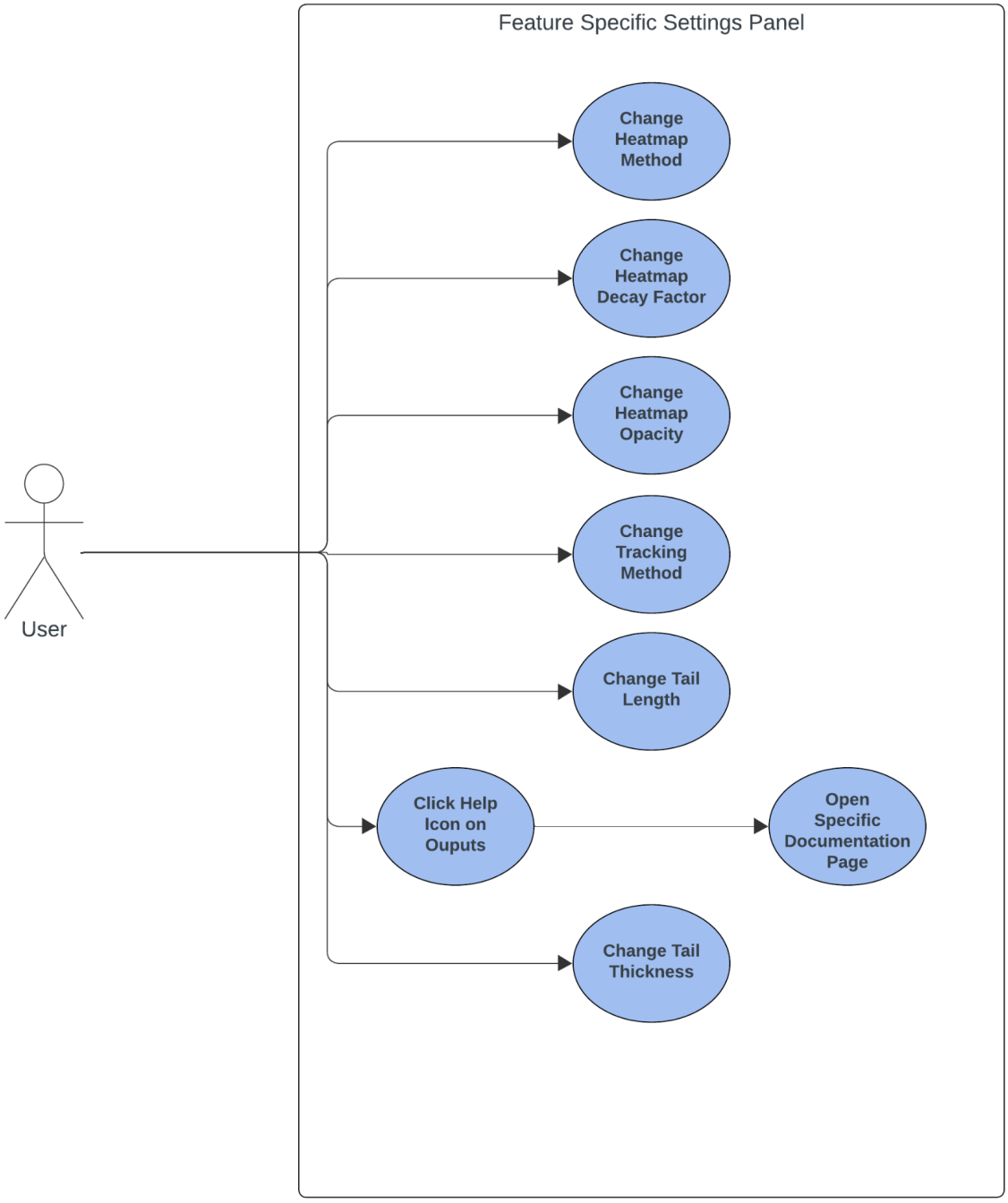
4.2.3.1 Legend

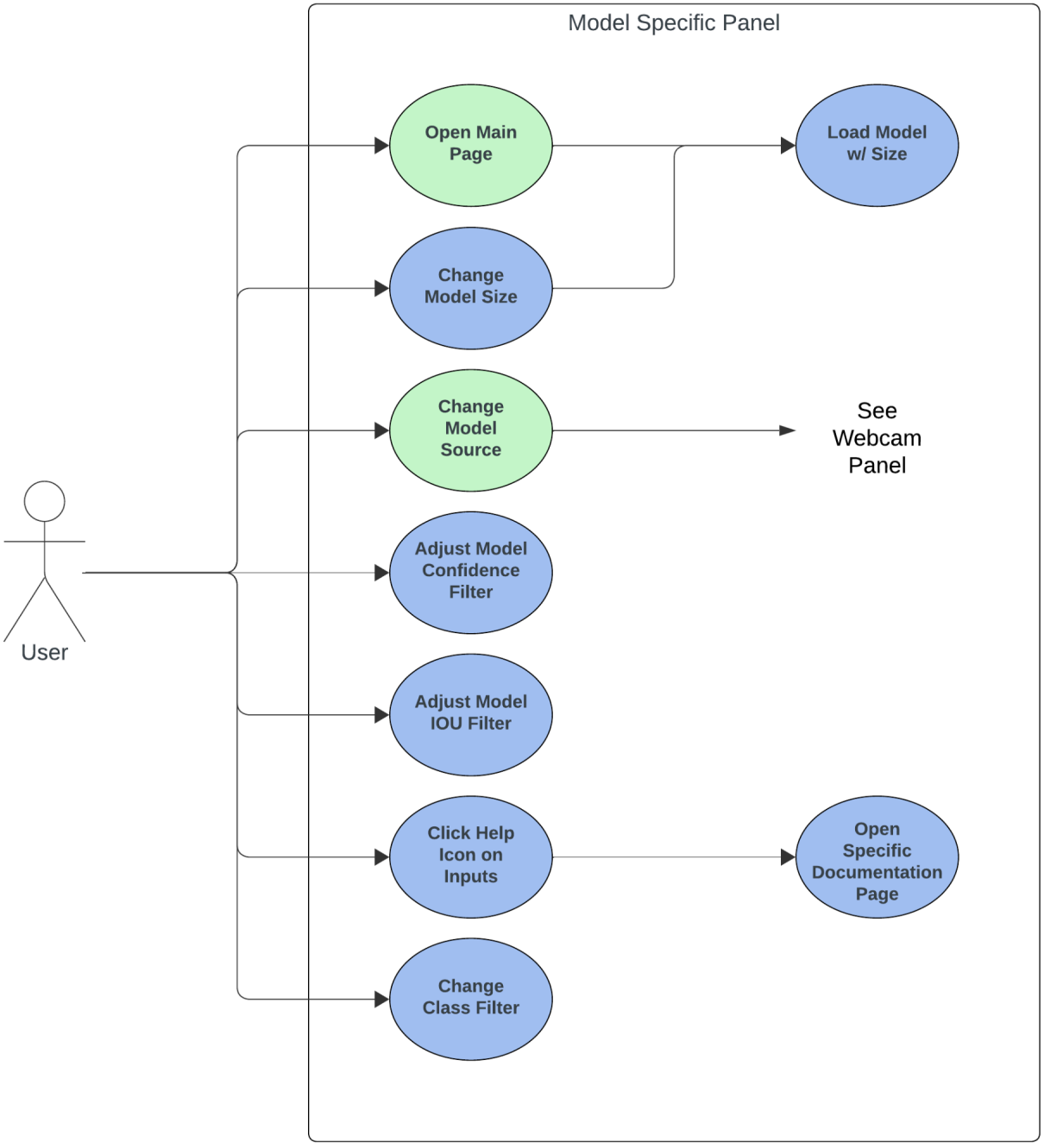


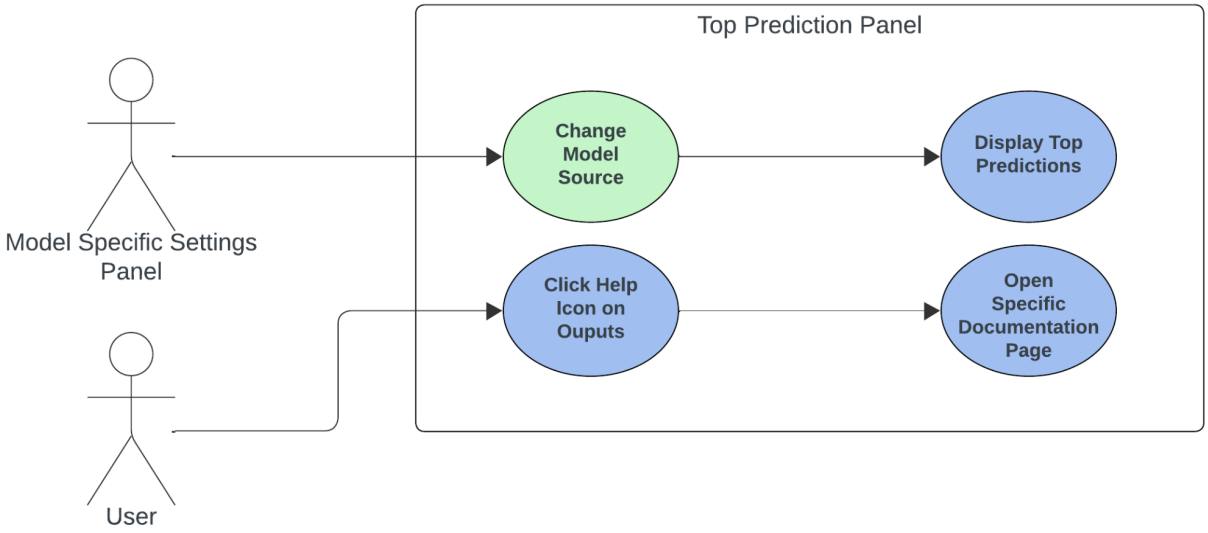
4.2.3.2 Diagram




Feature Settings Panel
Enable Confidence Hinting
Enable Visualize Heads
Enable Heatmaps
Enable Tracking
Enable Pose Estimation
Enable Segmentation Maps
Show Specific Setting Tab
Feature Settings Panel
Click Help Icon on Inputs
Open Specific Documentation Page
User


Feature Specific Settings Panel
Change Heatmap Method
Change Heatmap Decay Factor
Change Heatmap Opacity
Change Tracking Method
Change Tail Length
Click Help Icon on Ouputs
Open Specific Documentation Page
Change Tail Thickness
User


Model Specific Panel
Open Main Page
Load Model w/ Size
Change Model Size
Change Model Source
See Webcam Panel
Adjust Model Confidence Filter
Adjust Model IOU Filter
Click Help Icon on Inputs
Open Specific Documentation Page
Change Class Filter
User



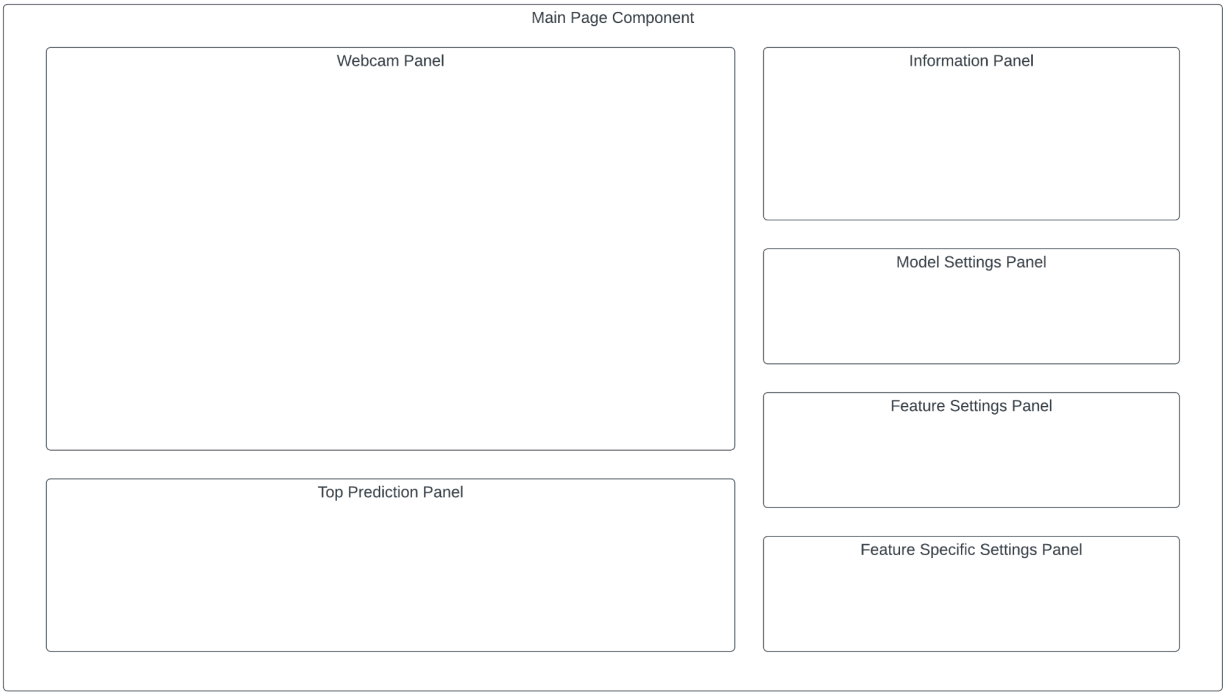
5. Design

5.1 High-Level Overview

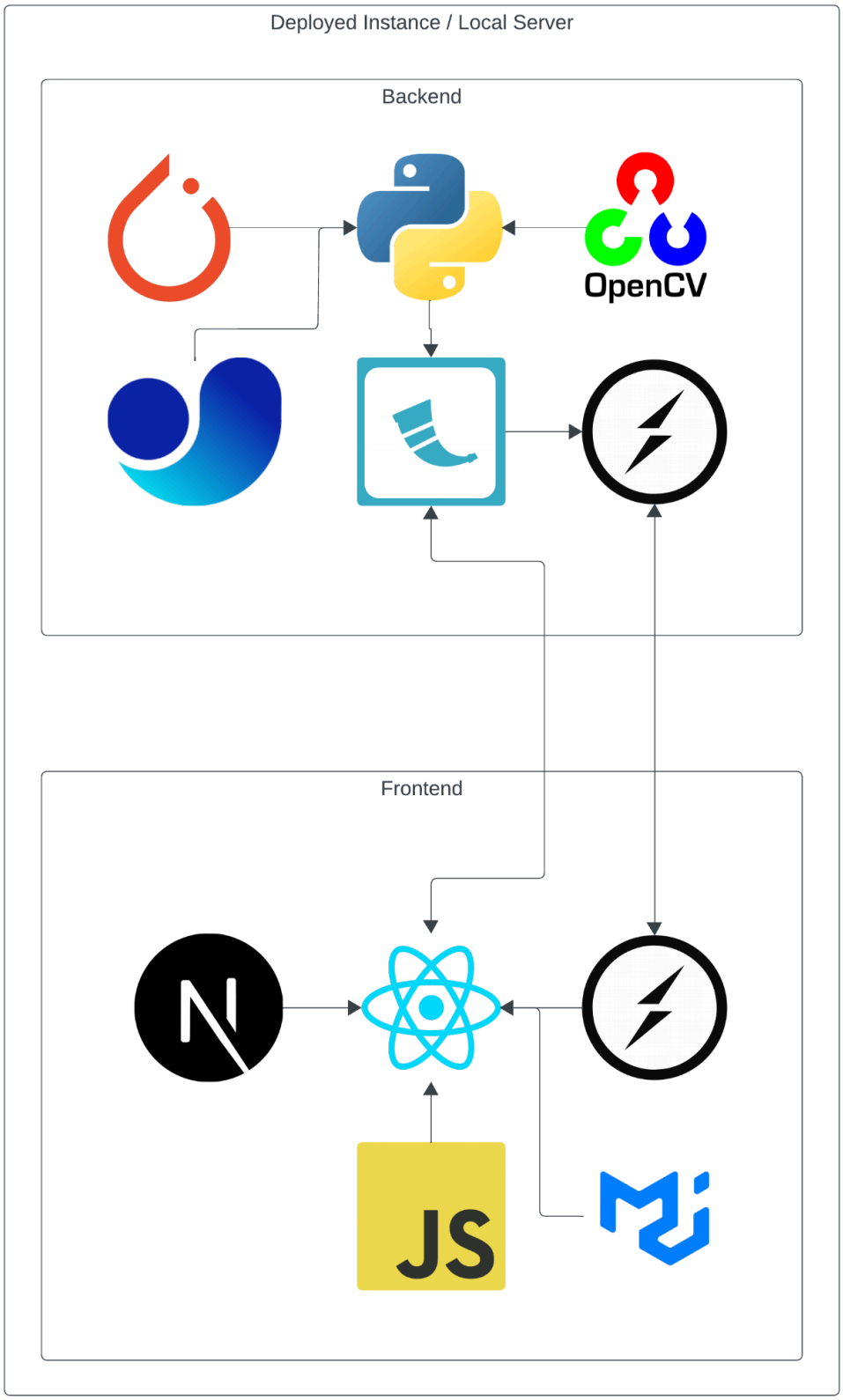
The product will use Python for the Backend, and JS for the frontend. These two sections will communicate with REST API endpoints via Flask, and Websockets for fast real-time data via SocketIO.

The frontend in addition to python will be using Flask to serve the REST API endpoints. Using Flask will also make it easier to create these endpoints and has plenty of documentation and support for modern python versions. Flask is also compatible with SocketIO. The AI model side of the backend will be handled with Pytorch. Pytorch is a standard AI framework which allows users to easily utilize CUDA cores of a variety of GPU compute hardware, while also abstracting away much of the complexities of AI models. In conjunction with Pytorch the backend will use the Ultraytics which abstracts away the complexities of the YOLO model and provides a OOP interface to easily control model setup, inference, and pipeline creation.

5.1.1 Rough UI Layout



5.1.2 Overall Tech Stack



5.2 Main Page

The main page will hold all other panels. It will be the first place the user loads into and will be the not be navigated away from. The main page will also be flexible and control the layout of the other panels to fit on the users screen. The main page will also

5.3 Webcam Panel

Will constrain the webcam panel and a button to change from light mode to dark mode. This is the primary component in the display and is the main “output” the user will be looking at as it's the direct output of the model. The webcam panel will be updated in real time with images from the backend. This will be achieved with a multi-part stream.

5.4 Information Panel

Will display real-time inference metrics, such as post-process, pre-process, inference, and overall FPS with detected objects. All 6 of these will be updated at once every frame process on the backend and emitted back to the frontend client via a websocket connection.

5.5 Feature Settings Panel

The feature settings panel will display buttons which the user can press to enable or disable features. The features will only be enabled one at a time. Includes options like segmentation maps, pose estimation, tracking, and heatmaps.

5.6 Feature Specific Settings Panel

The feature specific settings panel contains all the available settings for features in the feature settings panel. For Heatmaps it will allow the user to change the applied opacity, and heatmap method, and decay factor. For Tracking it will allow the user to adjust track length, thickness, and tracking detection method.

5.7 Model Specific Settings Panel

The model specific settings panel has options to change the model size (decreasing performance but increasing the accuracy), the source,

5.8 Top Prediction Panel

6. Technical Architecture

6.1 Deployment Instructions

The product will be deployed either on a dedicated instance or a local host. For local environment setup, Install latest version of React.js and Node.js, along with the latest python

version. Then utilizing the configuration files, install required packages and then run the provided start script via the script folder. This will launch the product via its specified port. If deploying to a dedicated instance, further setup will be required to configure firewalls to portforward users to the product. This is not needed in a local environment.

7. Configuration

The configuration will be broken into frontend and backend. The frontend will only need React.js. And Node.js. Node will handle the rest of package installations and management of the configuration via the provided files on the github.

The backend configuration all controlled via pip via a requirements.txt file with all required packages. Ultralytics will handle the AI model setup and configuration.

7.1 Configuration Files

Please see the product github for the configuration files.

8. Roles and Responsibilities

Name	Role
Bradley Gathers	SM, PL, PO, Developer

9. Terms and Definitions

9.1 Terms

Name	Description
YOLO	A Real-time Object detection model introduced 4 years ago. Has has many iterations usually (but not always) denoted by a version number. I.e. YOLOv3, YOLOv4, YOLOv7, YOLOv8. There have been large technical changes across these versions that distinguish them from each other. Generally, the accuracy of object detection in real-time goes up as the version numbers increase. The

	computational size of the models is denoted by (s, n, m, xl, e, c) with some letters abbreviations of size (s for small, n for nano) and others as just denote of different sizes with no meaning. The computational size of the models does not change much across versions.
WebSocket	A WebSocket is a direct communication protocol that allows for real-time communication between client and server via a raw TCP socket. There are typically packages and libraries that abstract out the low-level features with high-level implementations and supporting functions.
Ultralytics	One of the foundational creators of YOLOv3, YOLOv5, YOLOv7, and YOLOv8. A large development team and the company responsible for abstracting down SOTA research in YOLO and integrating it into a well-designed, organized, and well-abstracted Python package.
REST API	An API that follows the REST ideologies of endpoints.

9.2 Abbreviations

Name	Abbreviation
Real-time AI Object Detection Web App	RAODWA
Artificial Intelligence	AI
You Only Look Once	YOLO
Application Programming Interface	API
JavaScript	JS
Product Owner	PO
Product Lead	PL
Scrum Master	SM