

July 13th, 2021.

Craig Scratchley
School of Engineering Science,
Simon Fraser University,
Burnaby, BC
V5A 1S6



Re: ENSC 405W Requirement Specifications for Eye-bex by Eye-bex Inc.

Dear Dr. Scratchley,

Please find the design specifications document for Eye-bex Inc.'s first product, Eye-bex. This document was created according to the ENSC 405W's course instructions. The Eye-bex is a route setter's best friend in an indoor climbing gym. It provides significant data about the popularity, reliability, and difficulty of the routes that have been set and are in use by a gym's climbers. It does this through a camera system that uses intelligent computer vision to analyze the climbers and report back to the route setter through a web dashboard.

This document serves the purpose of outlining the design specification that the team has set for the product in an ordered and organized manner. The proof-of-concept demo will showcase this design in the future. The document consists of alternatives considered, test plans as well as justifications for each decision made in the process of design of this product.

Eye-bex Inc. is a team of 7 engineering students from different disciplines ranging from Computer to Systems Engineering. Our members are Amritha Raj K.R, Arsenen Gervacio, Benjamin Martin, Kay Arellano, Nitish Mallavarapu, Takunda Mwinjilo and Yogesh Mundhra.

Thank you for taking your time to read the Eye-bex design specification document. Any questions or comments in regard to the document can be sent to our Chief Communication Officer, Yogesh Mundhra at ymundhra@sfu.ca.

Best Regards,

Benjamin Martin

Benjamin Martin,
Chief Executive Officer,
Eye-bex Inc.

EYE-BEX INC.

EYE-BEX DESIGN SPECIFICATION



Company 5:

Amritha Raj K.R
Arsenen Gervacio
Benjamin Martin
Kay Arellano
Nitish Mallavarapu
Takunda Mwinjilo
Yogesh Mundhra

Abstract

The Eye-bex system harnesses computer vision to provide feedback to climbing gym route setters in the form of statistics. Our purpose is to assist in mentoring new route setters and improve the current setting team by providing the data requested when consulting climbing gym managers.

This design specification document follows in line with the requirement specification document in mind. Design specifications that stem from the requirements document are outlined in the following document. These design specifications were put in place after thorough examination of each system, subsystem and component present in the Eye-bex and the clients' needs in mind. This included comparing alternatives for each component in the hardware/firmware, computer vision and web app systems. The design specifications are followed by appendices of a test plan for each of the systems and their various parts, alternative design choices, as well as user interface and appearance design.

Change Log

Version number	Description of change	Date made	Made by
1.0	Initial document	7th July 2021	Entire team
2.0	Hardware Section and Appendix Draft Completed	8th July 2021	Hardware Team (Ben, Takunda, Yogesh)
3.0	WebApp Section and Appendix Draft	9th July 2021	Web App Team (Kay, Arsenen)
4.0	Computer Vision Section and Appendix Draft Completed	10th July 2021	Computer Vision Team (Amritha, Nitish)
5.0	Appendix A, Abstract, Introduction, Conclusion, References	10th July 2021	Hardware team
6.0	List of Tables & Figures, Formatting, Glossary, References	11th July, 2021	Computer Vision Team

Approvals

Name (Position):	Benjamin Martin (CEO)
Signature:	<i>Benjamin Martin</i>
Date:	July 11 th 2021

Table of Contents

Abstract	3
Change Log	4
Approvals	5
Table of Contents	6
List of Figures	9
List of Tables	10
Glossary	11
1. Introduction	12
1.1 Background	12
1.2 Scope	13
1.5 Design Classification Convention	14
2. System Architecture	15
3. Hardware	16
3.1 Microcontroller Unit	16
3.2 Camera Module	19
3.3 Casing	21
3.3.1 Casing Design	21
4. Computer Vision	24
4.1 Object detection	24
4.2 Motion Tracking	25
4.3 Cloud Services	26
5. Web Application	28
5.1 Frontend	29
5.2 Server-Side	30
5.3 Database	31
5.4 Deployment	33
6. Conclusion	34
References	36

Appendix A: Test Plan	40
A.1 Introduction	40
Test Purpose	40
Test Coverage	40
Test Methods	40
Test Responsibilities	40
A.2 Hardware Testing	40
A.3 Computer Vision Testing	42
A.4 WebApp Testing	45
Appendix B: Supporting Design Options	48
B.1 Hardware Options	48
B.1.1.1 Microcontroller Options	48
B.1.1.2 Microcontroller Decision	49
B.1.2.1 Camera Module Options	50
B.1.2.2 Camera Module Decision	52
B.1.3.1 Casing Options	53
B.1.3.2 Casing Decision	54
B.2 Computer Vision	54
B.2.1.1 Object Detection Options	54
B.2.1.2 Object Detection Decision	54
B.2.2.1 Motion Tracking Options	54
B.2.2.2 Motion Tracking Decision	54
B.2.3.1 Cloud Services Options	55
B.2.3.2 Cloud Services Decision	55
B.3 Web App Options	55
B.3.1.1 Front End Options	55
B.3.1.2 Front End Decision	55
B.3.2.1 Server-Side Options	55
B.3.2.2 Server-Side Decision	56
B.3.3.1 Database Options	56
B.3.3.2 Database Decision	57
B.3.4.1 Deployment Options	57
B.3.4.2 Deployment Decision	57

Appendix C: User Interface and Appearance Design	58
C.1 Introduction	58
C.2 User Analysis	58
C.3 Technical Analysis	59
C.3.1 Discoverability	59
C.3.2 Feedback	60
C.3.3 Conceptual Models	62
C.3.4 Affordances	62
C.3.5 Signifiers	62
C.3.6 Mappings	63
C.3.7 Constraints	63
C.4 Engineering Standards	64
C.5 Analytical Usability Testing	65
C.6 Empirical Usability Testing	66
C.7 Conclusion	68

List of Figures

FIGURE 1	Two climbers scaling a bouldering wall	Pg. 12
FIGURE 2	Overview of the Eye-bex design	Pg. 15
FIGURE 3	Raspberry Pi Zero W with ports labelled	Pg. 16
FIGURE 4	The Raspberry Pi Camera V2	Pg. 19
FIGURE 5	Raspberry Pi casing	Pg. 21
FIGURE 6	Computer Vision System Flowchart	Pg. 24
FIGURE 7	YOLO object detection algorithm in action	Pg. 24
FIGURE 8	Web application system	Pg. 28
FIGURE 9	Database Schema	Pg. 32
FIGURE 10	Home Page Mockup	Pg. 60
FIGURE 11	Route Statistics Page Mockup with Toast Notification	Pg. 61
FIGURE 12	Login Page Mockup	Pg. 64

List of Tables

TABLE 1	Design Specifications For The Microcontroller	Pg. 17
TABLE 2	Raspberry Pi Zero W Specifications	Pg. 18
TABLE 3	Design Specifications For The Camera Module	Pg. 20
TABLE 4	Raspberry Pi Camera Module V2	Pg. 20
TABLE 5	Design Specifications For Casing	Pg. 22
TABLE 6	Physical Specifications for Casing	Pg. 22
TABLE 7	Design specifications for YOLO object detection	Pg. 25
TABLE 8	Design specifications for YOLO and MotPy motion tracking	Pg. 26
TABLE 9	Design specifications for computing data	Pg. 27
TABLE 10	Design Specifications For Frontend	Pg. 29
TABLE 11	Design Specifications For Server-Side Software	Pg. 30
TABLE 12	Design Specifications For Database	Pg. 31
TABLE 13	Design Specifications For Deployment	Pg. 33
TABLE 14	Alternate Microcontroller Specifications	Pg. 49
TABLE 15	Alternate Camera Specifications	Pg. 51
TABLE 16	Alternate Casing Specifications	Pg. 54
TABLE 17	Alternate DatabaseSpecifications	Pg. 57

Glossary

Term	Definition
API	Application Programming Interface
AruCo Markers	Synthetic square marker composed by a wide black border and an inner binary matrix that determines its identifier
AWS	Amazon Web Services
CSI	Camera Serial Interface
Data Visualization	Graphical representation of information and data
EC2	Elastic Compute Cloud
Figma	Online UI tool to create, collaborate, prototype and handoff
GPIO	General Purpose Input/Output
JWT	JSON Web Token
LED	Light Emitting Diode
MERN	Javascript stack used for easier and faster deployment of full-stack web applications, comprises of MongoDB, Express, React and Node.js
MotPy	Library for track-by-detection multi object tracked implemented in python
PaaS	Platform as a Service
Route Setter	Climbing Gym employee responsible for setting routes
S3	Simple Storage Service
Schema	Structure described in a formal language supported by the Database Management System
SD	Secure Digital
USB	Universal Serial Bus
YOLO	You Only Look Once

1. Introduction

1.1 Background

Indoor rock climbing is a growing industry that provides physical activity for people of all ages [1]. Climbing gym walls have a collection of rocks that form routes. Typically each route is distinguished by a unique colour from its neighbors. In the figure below, a member can be seen climbing the red route on the left, and another member is climbing the purple route on the right.



Figure 1: Two climbers scaling a bouldering wall

Designing routes that are accessible and interesting for all body types is the challenge of professional route setters employed by climbing gyms. All gyms aspire to provide accessibility, and previous to our device the only tools were route setter's individual intuition and experience. Through consultation with local climbing gym owners and route setters, we have devised the idea for Eye-bex. To assist route setters with this task, our

company will be developing Eye-bex: a camera-based system that utilizes computer vision to track activity in climbing gyms and provide new metrics for route management.

The Eye-bex system can simply be mounted to view the targeted climbing wall and track the movement of a climber and their interaction with certain holds. It can then analyze this data to provide feedback on different routes which gyms can access through our web application that will display the data Eye-bex has processed.

1.2 Scope

This document lays a foundation for the design of the Eye-bex and includes all considerations made. These considerations are followed with rationale for the proof-of-concept version of Eye-bex as well as a later engineering prototype.

The design sections specify the functionality of each subsystem, the factors of the design choices, and the final selection of components. Appendix A provides a supporting test plan for each subsystem, including the test procedure, the expected outcome, and room for additional notes. Appendix B details additional design alternatives, and the justification of the final design choice.

1.3 Design Challenges

Several design challenges arose in the deliberation for each system. In terms of hardware, there was a question on whether a single-camera system would be enough to gauge a climb holistically. One of the data points deemed necessary after consultation with the client was a climber's height. A climbing wall is slanted with the top portion closer to the camera than the bottom of the wall. A single camera provides a two-dimensional image, so a simple conversion from pixels to distance units is not a viable solution. Several open-source computer algorithms have recently become available to meet these demands with the addition of stickers of predetermined dimensions present in the frame of the video. These stickers can be placed on the empty portions of the wall to provide calibration for the size conversions of actual objects. Climbing gym owners encouraged this solution as a branding opportunity.

A major decision that the team had to make was that of on-device processing versus cloud processing. The benefits of on-device processing of video included no recurring fees associated with the cloud, reliability and keeping the system as local as possible to the

gym. However, there were drawbacks because of the lack of scalability, the need for maintenance, the cost of the processing hardware and the inability to collaborate on the project remotely for all team members due to the pandemic. Cloud processing entailed scalability, lower initial costs for the processing power required and the novelty of the solution given the rise of cloud computing.

The overall functionality of the Eye-bex hinges on successful computer vision processing. Some challenging aspects include tracking individual instances of climbers through video frames, determining an algorithm for detecting when a hold is being used, and the obscuring of the climber's hands by their body.

1.5 Design Classification Convention

This section outlines the convention used for labelling and numbering specific requirements.

$$D-X.Y.Z.\{a\}/\{b\}$$

where the D is a constant for all design specifications listed and X, Y and Z are used for section, subsection, and sub-subsection, respectively. The letter a represents the requirement is intended for the alpha or proof-of-concept. These will be demonstrated at the ENSC 405W demo. The letter b is to indicate that the requirement is for the beta or prototype stage.

2. System Architecture

The Eye-bex is made up of 3 systems: the hardware system, the computer vision system and the web application (web app) system. The end-user will interact with the web-app and the hardware system is installed by Eye-bex for each particular gym. The client will have access to all the necessary statistics for each route that they have set and be able to make decisions for routes in the future. The bulk of the processing of the data/statistics presented to the user takes place in the computer vision system. Here, recorded video is received from the hardware system through a wireless connection and analyzed to produce raw data. The hardware system is responsible for recording and uploading video in chunks to a cloud computing setup where the computer vision system resides. This hardware system is a simple setup including a camera connected to a microcontroller with wireless access to the climbing gym's WiFi network. The in-depth system overview is presented in Figure 2.

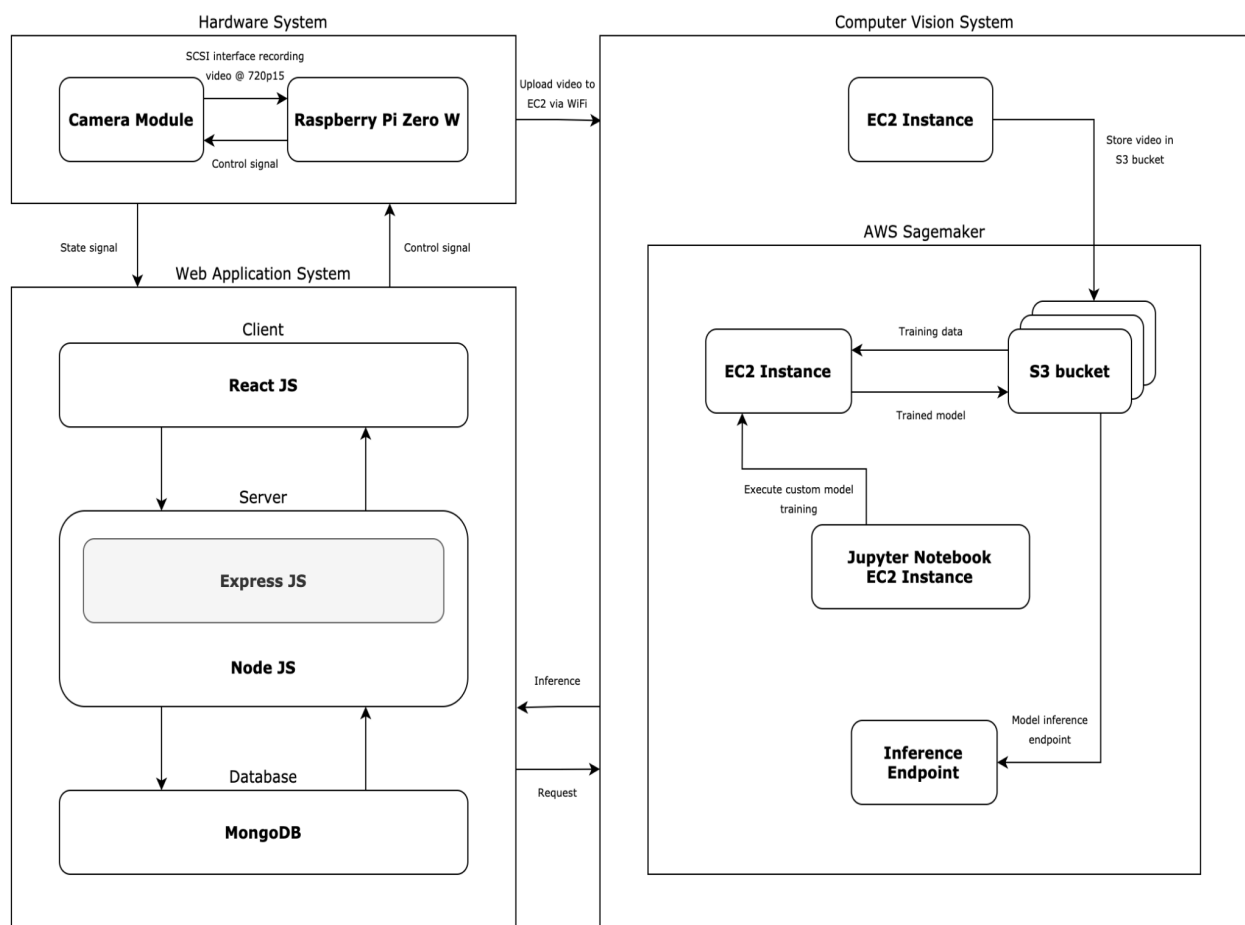


Figure 2: Overview of the Eye-bex design

3. Hardware

The hardware components are chosen to satisfy the physical and firmware requirements of the system. They include the design choices in microcontroller, camera module, and casing.

3.1 Microcontroller Unit

For this project, since the video processing is conducted on the cloud, the choice of a microcontroller with wireless capabilities was paramount. Additionally, the microcontroller itself need not be powerful enough to do computer vision calculations, however, it must be able to record and store high-quality video. The microcontroller that is chosen must be able to function without any connection to a laptop/computer since Eye-bex is a self-contained unit that only needs a power supply and access to a wireless network. Since the microcontroller will be storing large quantities of high quality video, it should also have storage abilities to support that.

With those considerations in mind, the choice of microcontroller was the Raspberry Pi Zero W. This is an affordable but powerful microcontroller with a small footprint that is powered through one of 2 available microUSB ports. The Pi Zero W is a variant of the Pi Zero with WiFi as well as Bluetooth capabilities. On top of its physical specifications, there is a large amount of online documentation available as well as projects to help in development for the Eye-bex. The Pi Zero W also features a camera port using a CSI camera connector. This was a key factor in the choice of this microcontroller as it allows for a range of cameras to be used with it.

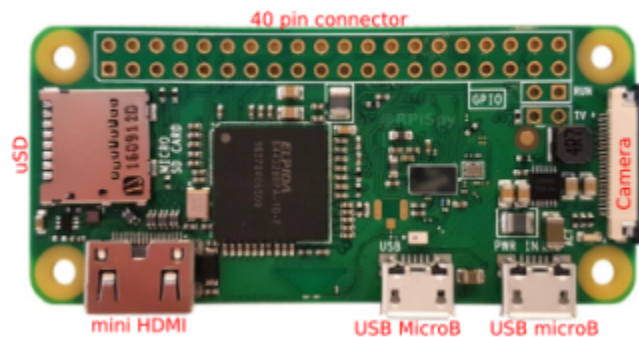


Figure 3: Raspberry Pi Zero W with ports labelled

TABLE 1
Design Specifications For The Microcontroller

Specification ID	Specification Description	Requirement Reference ID
D-3.1.1.a	The microcontroller will be able to communicate via 2.4GHz wireless connection	R-3.7.2.a
D-3.1.2.a	The microcontroller will support a 8MP camera through a CSI connector	R-3.4.2.b
D-3.1.3.a	The microcontroller will be powered through a microUSB port.	R-3.4.1.b
D-3.1.4.a	The microcontroller will support a microSD card to store videos	R-3.6.1.a
D-3.1.5.a	The microcontroller will allow the user to set up a wireless connection via the microUSB port.	R-3.4.4.a R-4.1.2.a R-5.4.1.a
D-3.1.6.a	The microcontroller will allow Python scripts to automate functionality.	N/A
D-3.1.7.a	The microcontroller will receive and execute commands from the remote cloud computer (EC2) when needed.	R-5.4.2.a
D-3.1.8.a	The microcontroller will power LED lights through GPIO pins for status updates.	N/A
D-3.1.9.b	The microcontroller will recover data and restore to its previous state in case of a power interruption.	N/A
D-3.1.10.b	The microcontroller will recover data and restore to its previous state in case of a wireless connection interruption.	N/A

TABLE 2
Raspberry Pi Zero W Specifications

Specification type	Specification
Physical dimensions (length x width x height)	66.0mm x 30.5mm x 5.0mm [2]
Weight	9g [2]
Processor	Broadcom BCM2835 @1Ghz single-core [2]
Memory	512MB RAM [3]
I/O	HAT-compatible 40-pin header [3]
SD Card Support	microSD card with a maximum 256GB boot partition [4]
Wireless Connectivity	<ul style="list-style-type: none"> • 802.11 b/g/n wireless LAN [3] • Bluetooth 4.1 [3] • Bluetooth Low Energy [3]
Input Power	Micro USB power [3]
Environment	0°C to 70°C [5]
Ports	<ul style="list-style-type: none"> • CSI camera connector [3] • Mini HDMI [3] • USB data/power port [3] • USB power port [3]
Operating System	<ul style="list-style-type: none"> • Raspberry Pi OS with desktop • Release date: May 7th 2021 • Kernel version: 5.10 [6]

3.2 Camera Module

The camera module was a key piece of the overall project. From the requirements document, it must be high quality enough to gauge details on a climbing wall from a 20 feet distance. However, it must also satisfy weight constraints and be compatible with the microcontroller of choice with ease. The choice of the camera module naturally followed the choice of the microcontroller, in this case, the Pi Zero W. The Pi Zero W's CSI interface allows a range of Pi cameras to be chosen from. Of these, we wanted a camera that reliably records video in at least 720p resolution and 15 frames per second. The camera module choice also had to take into account availability in Vancouver.

The camera module that was selected for this project was the Raspberry Pi V2. This camera is able to record 1080p video at 30 fps, which exceeds the base requirements that were in place. The Pi Camera V2 is a general-purpose camera, in the sense that it does not have any specialized lenses such as fisheye, wide-angle, or night vision. The camera is a light-weight module which is helpful in reducing the overall weight of the device. This camera module was relatively affordable and available readily. The camera module also required the purchase of a ribbon cable as the CSI connector on the Pi Zero is slightly smaller than other Pi Models.



Figure 4: The Raspberry Pi Camera V2

TABLE 3
Design Specifications For The Camera Module [7]

Specification ID	Specification Description	Requirement Reference ID
D-3.2.1.a	The camera module will record video in colour at 720p and 30 frames per second.	R-3.3.1.a R-3.3.2.a R-3.3.3.a
D-3.2.2.a	The camera module will connect directly to the microcontroller with a CSI cable.	R-3.4.2.b
D-3.2.3.a	The camera module will be supported by the OS running on the Pi Zero W.	N/A
D-3.2.4.a	The camera module will have adjustable focus.	N/A
D-3.2.5.a	The camera module will recover from a power failure.	N/A

TABLE 4
Raspberry Pi Camera Module V2 [8]

Specification type	Specification
Physical dimensions (length x width x height)	25mm x 23mm x 9mm
Weight	3g
Resolution	8MP
Sensor	Sony IMX219 image sensor
Max image quality	3280 x 2464 pixels
Video Quality supported	1080p at 30fps 720p at 60fps 480p at 90fps
Interface	Camera Serial Interface (CSI)

OS Support	Raspberry Pi OS
Horizontal field of view	62.2 degrees
Vertical field of view	48.8 degrees

3.3 Casing

The physical casing houses the Pi Zero W and the Pi camera, as well as the ribbon cable, and provides a mount for the device. It also provides an added layer of protection from the environment and potential falls.

3.3.1 Casing Design

The Eye-bex casing design is required to be low weight, mountable, and provide space for output status LEDs. The casing should support screw mounting to the wall, and provide access to all necessary input ports used in the design which are specified in section 3.1. Ideally, it should be 3d-printable for affordability, local accessibility, and customizability. It should protect the device from any reasonable impact, and have vents to cool the device.

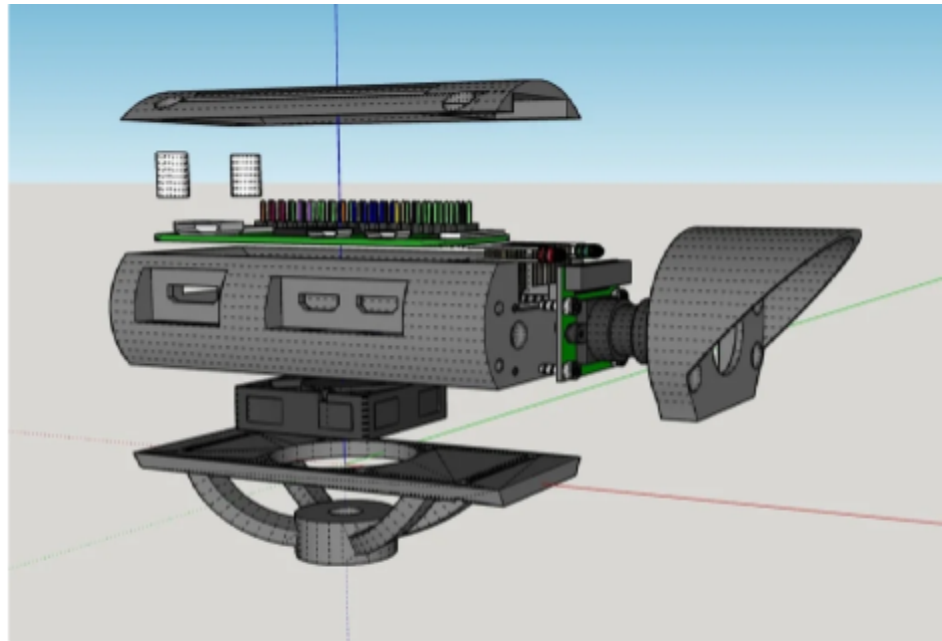


Figure 5: Raspberry Pi casing

TABLE 5
Design Specifications For Casing [9]

Specification ID	Specification Description	Requirement Reference ID
D-3.3.1.b	The case will encompass the camera module, the microcontroller, the ribbon cable as well as the wiring for 2 status LEDs.	R-3.2.4.b
D-3.3.2.b	The case will have both USB ports exposed for use.	R-3.4.4.a R-3.7.1.a
D-3.3.3.b	The case will not expose the SD card, or any physical circuitry with the exception of the ports.	R-3.2.4.b
D-3.3.4.b	The case will be wall mounted by screws.	R-3.2.5.b
D-3.3.5.b	The case will include vents for cooling.	R-3.2.1.b

TABLE 6
Physical Specifications for Casing [9]

Specification type	Specification
Physical dimensions (length x width x height)	110.44mm x 44.2mm x 57.65mm
Weight	47.7g
Tools and parts required for assembly	<ul style="list-style-type: none"> • 4x 2mm x 8-10mm long hex head bolts (pan or socket) • 4x 3mm x 35mm long hex head bolts (pan or socket) (bolts top, main body & bottom) • 2x 3mm x 5mm long hex head bolts (pan or socket) (holds one end of the zero down) • 4x 3mm x 8-10mm long hex head

	<p>bolts (pan or socket) (holds front to main body)</p> <ul style="list-style-type: none"> • 4x 3mm x 10 - 12mm long head head bolts (pan or socket) (holds fan to base) • 4x 3mm nuts • 1x Press in threaded insert.
--	--

4. Computer Vision

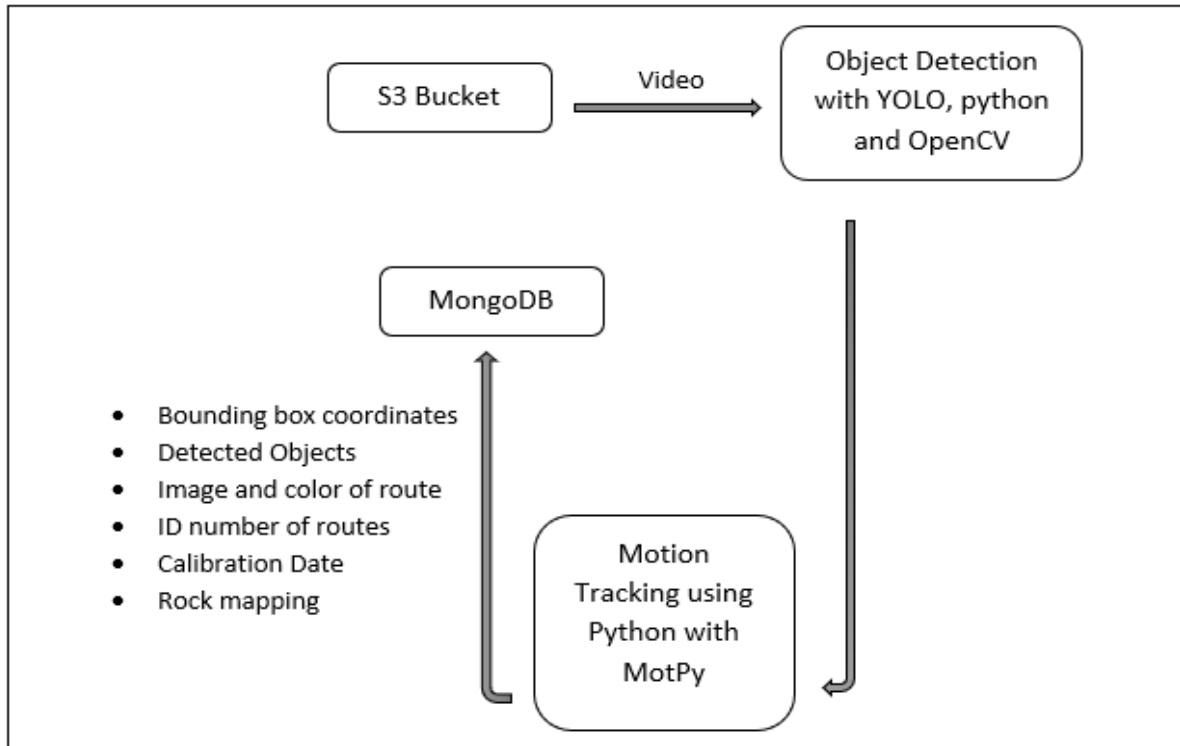


Figure 6: Computer Vision System Flowchart

4.1 Object detection

Object detection is one of the classical problems in computer vision where you work to recognize *what* and *where* — specifically what objects are inside a given image and where they are in the image. Our product Eye-Bex uses object detection and tracking for the analysis of the data from climbing gyms. Using input videos from the camera on the Eye-Bex hardware, Python and the OpenCV library will be utilized to implement the YOLOv3-416 algorithm.

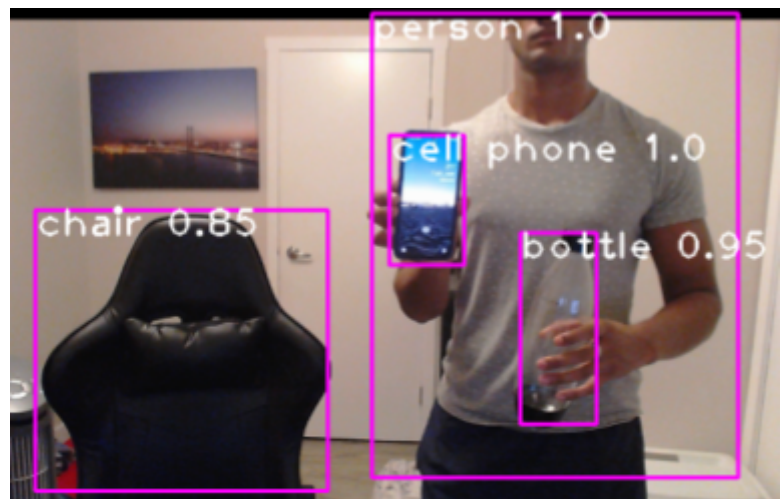


Figure 7: YOLO object detection in action

You Only Look Once created by Joseph Redmon and Ali Farhadi, is a real time object detection tool which can be used to identify specific objects in videos/images. YOLOV3 is fast and accurate for real time applications. It predicts an objectness score for each bounding box using logistic regression and multilabel classification for prediction of classes [10]. We will be building on the pretrained model and custom training it to suit our needs of classify rock features, various limbs, as well as calibration stickers.

TABLE 7
Design specifications for YOLO object detection

Specification ID	Specification Description	Requirement Reference ID
D-4.1.1.a	The object detection tool will use Python and openCV functions to process the images to detect and identify climbers, hands, feet, rocks by color and start/finish hold stickers	R-5.1.1.a R-5.1.3.a R-5.1.6.a R-5.1.8.a R-5.1.9.a R-5.1.10.a
D-4.1.2.a	ArUco markers will be used to calculate climbers height	R-5.2.7.a
D-4.1.3.a	YOLO model will be custom trained to recognize when a climber is actually holding onto a hold vs. hovering over a hold	R-5.2.1.a

4.2 Motion Tracking

The default motion tracking using just YOLO is quite jittery. We will be incorporating a library called MotPy that will enhance the ability to track climbers. MotPy considers previous frames to help smoothen out the YOLO bounding boxes, and give a more accurate bounding box in future frames. Each bounding box is detected and once tracked by MotPy, it shows the track box with an ID.

TABLE 8
Design specifications for YOLO and MotPy motion tracking

Specification ID	Specification Description	Requirement Reference ID
D-4.2.1.a	Motpy library will be utilized for accurate tracking of climbers, hands and feet	R-5.1.2.a R-5.1.5.a
D-4.2.2.a	When a climbers bounding box reaches the finish hold, a successful attempt will be recorded in the database for that specific route	R-5.2.5.a R-5.2.6.a R-5.2.3.a

4.3 Cloud Services

We will be using a variety of Amazon Web Services (AWS) to do all of the computer vision processing on the cloud, rather than the actual Raspberry Pi itself. We will be using the following Amazon Web Services

- *EC2 (Elastic Compute Cloud)*
Amazon EC2 is a web service that provides secure, resizable compute capacity in the cloud. EC2 is the only cloud with 400 Gbps ethernet networking. It allows you to run various virtual computers and manage the same with a single hardware.
- *S3 buckets (Simple Storage Service)*
S3 bucket is a cloud storage service to store and protect data. It is easy to use and manage. It provides a flat, non-hierarchical structure storing data as objects in buckets.
- *Lambda*
Serverless computing service in which several programming languages are supported and lets you build, test, and deploy functions without managing servers or runtimes. Our product eye - bex will be using lambda for all backend services.

TABLE 9
Design specifications for computing data

Specification ID	Specification Description	Requirement Reference ID
D-4.3.1.a	The data will be processed and available in cloud for access	R-5.2.9.a
D-4.3.2.a	Video from the Raspberry Pi will be uploaded to the EC2 server in chunks	N/A
D-4.3.2.a	Lambda will handle the computer vision processing of the video input	N/A
D-4.3.3.a	Processed data will be stored in S3 buckets for easy access from the web app	N/A

5. Web Application

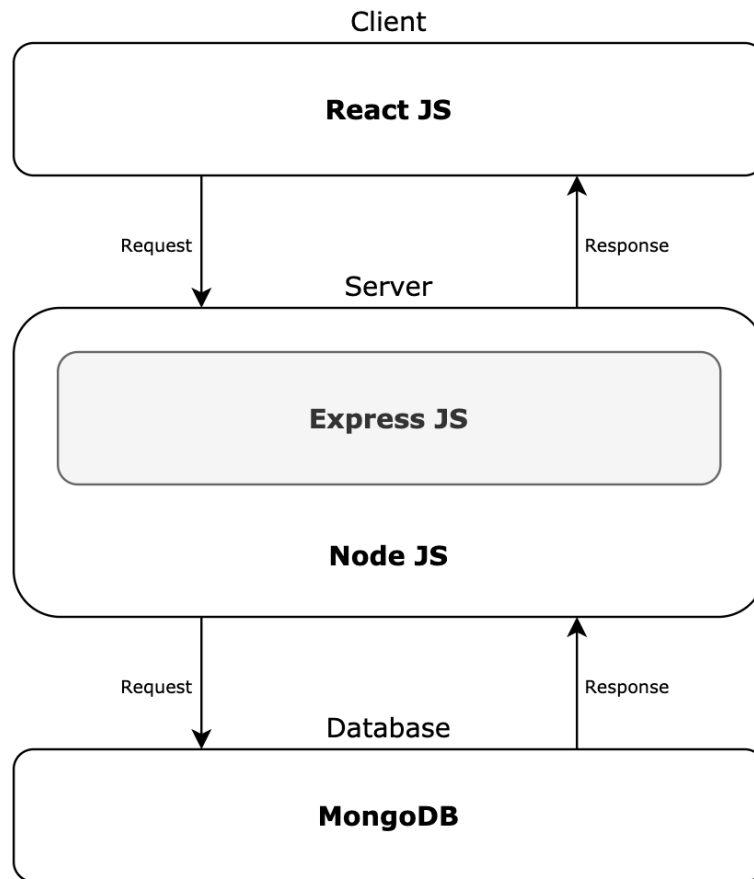


Figure 8: Web application system

Eye-bex's web application, or web app for short, has the purpose of conveying all important data in a manner intuitive to users with a simple interface. As we will be designing a single page web application, we have chosen the MERN tech stack that will suit the needs of our frontend, backend and database requirements. Figure 8 above illustrates the MERN tech stack diagram where we will use MongoDB, Express JS, React JS and Node JS.

The Eye-bex web app provide the following functionalities:

1. Log in as a routesetter of a climbing gym
2. Trigger the calibration of an Eye-bex camera system
3. Track active or archived routes
4. Filter through saved routes by using a tag system
5. View the statistics of a route with data visualization through charts, graphs etc.

5.1 Frontend

The user interface (UI) of the Eye-bex web app will be created using the React JS framework alongside HTML and SCSS. React JS is a popular frontend framework that will allow us to create a single page application (SPA) which will not require frequent page reloading [11]. By eliminating reloads, we can expect faster response times as we will not wait for a requested web page to render from the server. With React JS and its popularity, we can also take advantage of its large online community for resources and the wide array of libraries and packages dedicated to the framework.

As the client facing application of Eye-bex, we will also be utilizing Figma which is a user interface design tool that will help us create visual mockups and prototype our designs [12]. With Figma, we are able to leverage the free education plan granting us collaborative, synchronous design features to help us with our workflow.

TABLE 10
Design Specifications For Frontend

Specification ID	Specification Description	Requirement Reference ID
D-5.1.1.a	The designed mockups will be used as a guide to create UI components with React JS	R-7.1.1.a
D-5.1.1.b	The web app will provide a user login page to send a user authentication request to the server	R-5.3.1.b
D-5.1.2.b	The frontend app will send a request to the server to load an account specific dashboard page upon authentication	R-5.3.2.b
D-5.1.3.a	The web app will utilize a side navigation bar which will link to the settings, routes and route tag page	N/A
D-5.1.4.a	The routes page will display a selected route's related images and its statistics using a data visualization package called Nivo	R-5.3.3.a R-5.3.4.a R-5.3.5.a R-5.3.6.a R-5.3.7.a R-5.3.8.a

		R-5.3.9.a R-5.3.12.a
D-5.1.5.a	The settings page will provide an interface for camera system controls such as camera calibration and status alerts	R-5.3.10.a R-5.3.11.a R-5.3.16.a R-5.3.17.a
D-5.1.6.b	The route tag page will provide an interface to create custom tags	R-5.3.13.b R-5.3.14.b R-5.3.15.b

5.2 Server-Side

For the server-side implementation of Eye-bex we will be using Node.js with the Express.js framework. Node.js with Express is a very popular combination used by many and will allow us to create a flexible and robust server-side for the web app. With this framework we can communicate between the front-end and database by using Application Programming Interface (API) calls, some of which will need authentication [13].

TABLE 11
Design Specifications For Server-Side Software

Specification ID	Specification Description	Requirement Reference ID
D-5.2.1.a	The web server will use Node.js with an Express.js framework	R-5.5.1.b R-5.5.5.b R-7.4.2.b
D-5.2.1.a	Eye-bex will communicate using standard API calls	R-5.3.1.a R-5.4.2.a
D-5.2.2.b	Eye-bex will use protected API calls that require authentication	R-5.3.1.b R-7.7.1.b R-7.7.2.b

D-5.2.3.b	The web server will send an encrypted JSON Web Token (JWT) to the client upon user authentication	R-5.3.2.b
------------------	---	------------------

5.3 Database

The database we chose to use is MongoDB through MongoDB Atlas. MongoDB Atlas is a free cloud-based service that allows us to implement the database server for Eye-bex. MongoDB uses a noSQL structure which stores data in JSON (JavaScript Object Notation) documents. NoSQL is generally simpler and more flexible than SQL databases which is one of the major reasons we will be using MongoDB as it allows us to update and change the database schema as needed while developing without having to create new tables. As we will be using a NoSQL database we do not need to worry about normalization [14].

TABLE 12
Design Specifications For Database

Specification ID	Specification Description	Requirement Reference ID
D-5.3.1.a	Eye-bex will use a MongoDB database	R-5.5.1.b R-5.5.5.b R-7.4.2.b
D-5.3.2.a	The database will follow the Diagram shown in Figure 8	R-5.5.2.a R-5.5.3.a R-5.5.4.b R-5.5.6.a R-5.5.7.b

Figure 8 below is the schema for our database. We will be using 4 collections, “Company”, “User”, “Route”, and “Run”. The properties for each collection are outlined in the figure below.

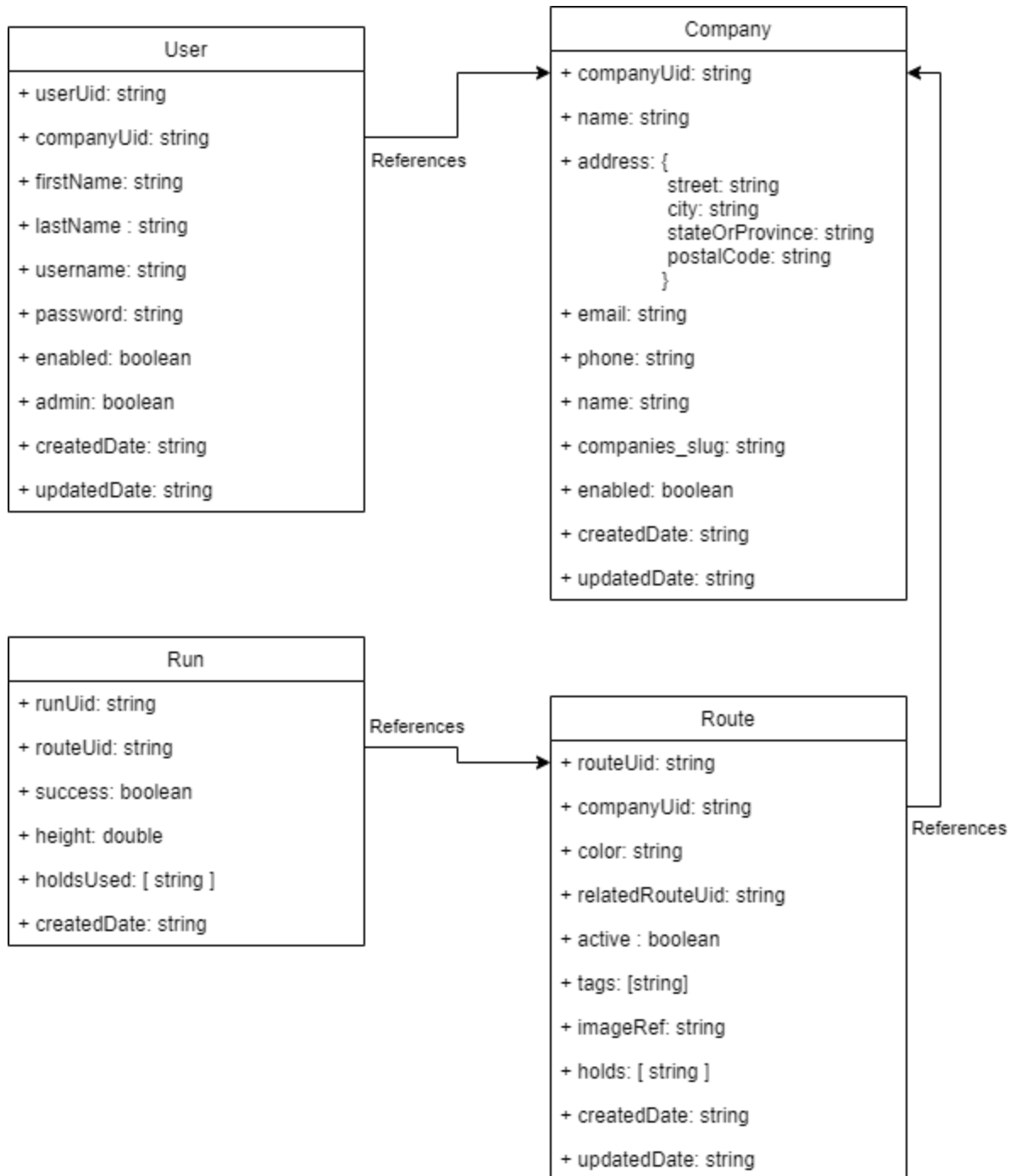


Figure 9: Database Schema

5.4 Deployment

For deployment of our web-app we will be using Heroku, along with a Github repository. Heroku is a cloud platform as a service (PaaS) that allows us to deploy and run the code we have created for our web app online. Heroku supports the 3 technology choices made in 5.1 - 5.3 and allows us a more convenient, all-in-one place to host and monitor our web-app [15].

TABLE 13
Design Specifications For Deployment

Specification ID	Specification Description	Requirement Reference ID
D-5.4.1.a	The web server and frontend will be hosted by Heroku	R-5.4.2.a

6. Conclusion

The Eye-bex consists of several parts developed incrementally and the decisions made for each system are listed below for the proof-of-concept prototype.

1. Hardware Design

a. Microcontroller

- The Eye-bex will be controlled by the Raspberry Pi Zero W microcontroller to record and transmit video to optimize for costs and weight. Raspberry Pi comes with a known standard of quality and is accompanied by relevant documentation to the design goals.

b. Camera

- The camera module will be the Raspberry Pi Camera V2. It is the most affordable option, accessible, and compatible with the microcontroller while meeting all requirements for video quality.

c. Casing

- The casing will be the 3d printable design detailed in section 3.3.1 for its local accessibility, affordability, customizability, appropriate access/cover of ports, and space for LED status lights.

2. Computer Vision Design

a. Object Detection

- YOLOv3 will be used because of its speed and accuracy for smaller objects like holds.

b. Motion Tracking

- MotPy will be used for motion tracking as it meshes well with the object detection algorithm and after some feedback from the Progress Review Meeting #2

c. Cloud Services

- AWS and the EC2 were chosen due to the widely available detailed documentation, gentler learning curve and affordability.

3. Web Application Design

a. Front End

- ReactJS was chosen due to the gentler learning curve and having a team member with previous experience in developing using it.

b. Server side

- NodeJS was chosen as the server-side framework to provide a ubiquitous approach using JavaScript and also the flexibility it provides
- In tandem with NodeJS, Express JS is used as the backend framework to complete the MERN tech stack as it supports single page applications well.

c. Database

- MongoDB Atlas was chosen as it is the most popular noSQL (non-relational) database available for free.
- The non-relational database approach allows us to have a flexible schema that can change as required by the client or by development.

d. Deployment

- Heroku was chosen as the cloud-based deployment platform as it is simpler to set up compared to alternatives such as AWS and provides enough functionality.

References

- [1] S. Farooqui, "Rock climbing's Olympic debut, and its growing popularity come with challenges," *CTVNews*, 11-Dec-2019. [Online]. Available: <https://www.ctvnews.ca/sports/rock-climbing-s-olympic-debut-and-its-growing-popularity-come-with-challenges-1.4726631>. [Accessed: 11-Jul-2021]
- [2] "Raspberry Pi Zero Guide: Projects, Specs, GPIO, Getting Started | Tom's Hardware", *Tomshardware.com*, 2021. [Online]. Available: <https://www.tomshardware.com/features/raspberry-pi-zero>. [Accessed: 08-Jul-2021]
- [3] "Buy a Raspberry Pi Zero W", *Raspberry Pi*, 2021. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-zero-w/>. [Accessed: 08-Jul-2021]
- [4] "SD cards - Raspberry Pi Documentation", 2021. [Online]. Available: <https://www.raspberrypi.org/documentation/installation/sd-cards.md>. [Accessed: 08-Jul-2021]
- [5] "FAQs - Raspberry Pi Documentation", *Raspberry Pi*, 2021. [Online]. Available: <https://www.raspberrypi.org/documentation/faqs/#pi-performance-temps>. [Accessed: 08-Jul-2021]
- [6] "Operating system images", *Raspberry Pi*, 2021. [Online]. Available: <https://www.raspberrypi.org/software/operating-systems/>. [Accessed: 08-Jul-2021]
- [7] "Buy a Camera Module V2", *Raspberry Pi*, 2021. [Online]. Available: <https://www.raspberrypi.org/products/camera-module-v2/>. [Accessed: 08-Jul-2021]
- [8] "Raspberry Pi Camera v2", *Farnell*, 202. [Online]. Available: <http://www.farnell.com/datasheets/2056179.pdf>. [Accessed: 08-Jul-2021]
- [9] A. Gilliam, "Raspberry Pi Zero W Security Camera", *Thingiverse*, 20-Sep-2017. [Online]. Available: <https://www.thingiverse.com/thing:2544275>. [Accessed: 08-Jul-2021]
- [10] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement", *Pjreddie.com*, 8-Apr-2018. [Online]. Available: <https://pjreddie.com/media/files/papers/YOLOv3.pdf> [Accessed: 10-Jul-2021].
- [11] K. Chinnathambi, "Creating a Single-Page App in React using React Router", *KIRUPA*, 2017. [Online]. Available: https://www.kirupa.com/react/creating_single_page_app_react_using_react_router.htm. [Accessed: 09-Jul-2021]

- [12] “About Figma, the collaborative interface design tool”, *Figma*, 2021. [Online]. Available: <https://www.figma.com/about/>. [Accessed: 09-Jul-2021]
- [13] “Express - Node.js web application framework”, *Express*, 2021. [Online]. Available: <https://expressjs.com/>. [Accessed: 09-Jul-2021]
- [14] “What Is MongoDB?”, *MongoDB*, 2021. [Online]. Available: <https://www.mongodb.com/what-is-mongodb>. [Accessed: 09-Jul-2021]
- [15] “Cloud Application Platform | Heroku”, *Heroku*, 2021. [Online]. Available: <https://www.heroku.com/>. [Accessed: 09-Jul-2021]
- [16] “Arduino Nano 33 IoT | Arduino Official Store”, *Arduino Official Store*, 2021. [Online]. Available: <https://store.arduino.cc/usa/nano-33-iot>. [Accessed: 08-Jul-2021]
- [17] “ARDUINO NANO 33 IoT WITH HEADERS”, *Lee’s Electronic Components*, 2021. [Online]. Available: <https://leeselectronic.com/en/product/109980-arduino-nano-33-iot.html>. [Accessed: 08-Jul-2021]
- [18] “Raspberry SC15184 Pi 4 Model B 2019 Quad Core 64 Bit WiFi Bluetooth (2GB) : Amazon.ca: Electronics”, *Amazon*, 2021. [Online]. Available: <https://www.amazon.com/Raspberry-Model-2019-Quad-Bluetooth/dp/B-07TC2BK1X>. [Accessed: 08-Jul-2021]
- [19] “Raspberry pi 4 Model B 2GB RAM”, *Lee’s Electronic Components*, 2021. [Online]. Available: <https://leeselectronic.com/en/product/167112-raspberry-pi-4-model-b-2gb-ram.html>. [Accessed: 08-Jul-2021]
- [20] “Raspberry Pi 4 Model B specifications”, *Raspberry Pi*, 2021. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/>. [Accessed: 08-Jul-2021]
- [21] “Raspberry Pi”, *Wikipedia*. 2021 [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi. [Accessed: 08-Jul-2021]
- [22] “Camera Module - Raspberry Pi Documentation”, 2021. [Online]. Available: <https://www.raspberrypi.org/documentation/hardware/camera/>. [Accessed: 08-Jul-2021]
- [23] “Arducam 12MP*2 Synchronized Stereo Camera Bundle Kit for Raspberry Pi and Pi zero, Two 12.3MP IMX477 Camera Modules with CS Lens and Camarray Stereo Camera HAT”, *Arducam*, 2021. [Online]. Available: <https://www.arducam.com/product/arducam-12mp2-synchronized-stereo-camera-bundle->

[kit-for-raspberry-pi-two-12-3mp-imx477-camera-modules-with-cs-lens-and-arducam-camera-array-stereo-camera-hat/](#). [Accessed: 08-Jul-2021]

[24] “OpenCV AI Kit: OAK-D/1 Camera Buy and Customize”, *Arducam*, 2021. [Online]. Available: <https://www.arducam.com/oak-opencv-ai-kit-camera/>. [Accessed: 08-Jul-2021]

[25] “Buy a Raspberry Pi Zero Case”, *Raspberry Pi*, 2021. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-zero-case/>. [Accessed: 08-Jul-2021]

[26] S. Daityari, “Angular vs React vs Vue: Which Framework to Choose in 2021”, *CodeinWP*, 10-Jan-2019. [Online]. Available: <https://www.codeinwp.com/blog/angular-vs-vue-vs-react/>. [Accessed: 09-Jul-2021]

[27] “Flask vs Django: How to Understand Whether You Need a Hammer or a Toolbox”, *Steel Kiwi*, 04-Oct-2019. [Online]. Available: <https://steelkiwi.medium.com/flask-vs-django-how-to-understand-whether-you-need-a-hammer-or-a-toolbox-39b8b3a2e4a5>. [Accessed: 09-Jul-2021]

[28] “What is the MERN Stack? Introduction & Examples”, *MongoDB*, 2021. [Online]. Available: <https://www.mongodb.com/mern-stack>. [Accessed: 09-Jul-2021]

[29] “About PostgreSQL”, *PostgreSQL*, 2021. [Online]. Available: <https://www.postgresql.org/about/>. [Accessed: 09-Jul-2021]

[30] “Why MySQL?”, *MySQL*, 2021. [Online]. Available: <https://www.mysql.com/why-mysql/>. [Accessed: 11-Jul-2021]

[31] V. Kuprenko, “Heroku vs. AWS: Which Cloud Solution Works Best in 2020”, *Cloud Academy*, 04-Feb-2020. [Online]. Available: <https://cloudacademy.com/blog/heroku-vs-aws-which-cloud-solution-works-best/>. [Accessed: 09-Jul-2021]

[32] D. A. Norman, *The Design of Everyday Things*, Revised and Expanded ed. New York, New York, USA: Basic Books, 2013.

[33] S. Munot, “Toast Notification or Dialog Box?”, *UX Planet*, 24-Jul-2017. [Online]. Available: <https://uxplanet.org/toast-notification-or-dialog-box-ae32ad53106d>. [Accessed: 09-Jul-2021]

[34] R. Roth, “3 Essentials for Great UX: Affordances, Signifiers & Feedback”, *Career Foundry*, 27-Oct-2020. [Online]. Available: <https://careerfoundry.com/en/blog/ux-design/affordances-signifiers-feedback/>. [Accessed: 09-Jul-2021]

- [35] "ISO/IEC/IEEE International Standard - Systems and software engineering - Engineering and management of websites for systems, software, and services information," in ISO/IEC/ IEEE 23026 First edition 2015-05-15 , vol., no., pp.1-54, 15 May 2015, doi: 10.1109/IEEESTD.2015.7106438.
- [36] "CAN/CSA-ISO/IEC 26557:18", Standards Council of Canada, 2018. [Online]. Available: <https://www.scc.ca/en/standardsdb/standards/29417>. [Accessed: 11- July- 2021].
- [37] "ISO/IEC TR 12182:2015", iTeh Standards, 2021. [Online]. Available: <https://standards.iteh.ai/catalog/standards/iso/dd4d54fe-b335-4b03-ad59-8424b1dc4752/iso-iec-tr-12182-2015>. [Accessed: 11- July- 2021].
- [38] "ISO 9241-161:2016 Ergonomics of human-system interaction -- Part 161: Guidance on visual user-interface elements", *CSA Group*, 2021. [Online]. Available: https://www.csagroup.org/store/product/iso_060476/. [Accessed: 11- Jul- 2021]
- [39] "ISO/IEC/IEEE 12207:2017 Systems and software engineering -- Software life cycle processes", *CSA Group*, 2021. [Online]. Available: https://www.csagroup.org/store/product/iso_063712/. [Accessed: 11- Jul- 2021]
- [40] C. Scratchley and M. Sjoerdsma. (2021). Human Factors & Usability Engineering [PowerPoint Slides]. Available: <https://canvas.sfu.ca/courses/63024/files/folder/Powerpoints?preview=16308516>

Appendix A: Test Plan

A.1 Introduction

Test Purpose

The Eye-bex test plan details the steps for acceptance testing when deploying the device with users. These tests also demonstrate meeting the design specifications for our proof-of-concept demonstration.

Test Coverage

The tests detailed in Appendix A cover the hardware, web-app, and computer vision subsystems. The tests also cover a range of priorities from Low to Medium to High to classify which tests are detrimental to overall functionality and which carry a lower significance in case of failure.

Test Methods

The testing will take place in-person at the Hangout Climbing gym facilities in Duncan, B.C. The deployment and testing team will travel to the location, install the prototype, and verify the tests described below on site.

Test Responsibilities

The verification of the subsystems of Eye-bex subsystems will be the responsibility of the Eye-bex Inc. deployment and testing team. At least one member from each of the 3 teams (hardware/firmware, computer vision, web app) will be present onsite at the climbing gym for testing.

A.2 Hardware Testing

Test Name: Connectivity to WiFi and EC2	Priority (Low/Med/High): High
Test Description: Verify that the Raspberry Pi Zero W connects wirelessly to the climbing gym's WiFi network and is able to ping the EC2 IP address and send data to it.	
Expected Outcome: The EC2 will respond to the ping requests and also be able to display test videos sent from the Pi Zero W.	

Passed (Yes/No) :
Notes on outcome:

Test Name: Video Recording Quality	Priority (Low/Med/High): High
Test Description: The video recorded by the Pi should be 720p 15fps, with enough focus to recognize and process height stickers. A test member will verify their height processed by EC2 is accurate.	
Expected Outcome: The height processed of the team member will be within an accuracy tolerance of 5cm.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Power supply interruption	Priority (Low/Med/High): Med
Test Description: Verify that the Pi Zero W recovers from a power cut (power cycle via unplugging the microUSB connector) to its previous state (recording/converting/uploading).	
Expected Outcome: The previous operation is started from the beginning or the microcontroller resumes where it had previously stopped.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Wireless connection interruption	Priority (Low/Med/High): Med
Test Description: Verify that the Pi Zero W recovers from a wireless connection interruption (momentarily turn off wifi and turn it back on manually on the Pi Zero W as an operation is run) to its previous state (recording/converting/uploading).	
Expected Outcome: The previous operation is started from the beginning or the microcontroller resumes where it had previously stopped.	

Passed (Yes/No) :
Notes on outcome:

Test Name: Daily storage cleanup	Priority (Low/Med/High): High
Test Description: Verify that all video once processed by computer vision and relayed to webapp is removed from the microSD card on the Pi Zero W.	
Expected Outcome: No videos on microSD card after clearing function is called.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Casing status	Priority (Low/Med/High): Low
Test Description: Verify that cables and internal circuitry are hidden and that status LEDs are all functioning. Verify the condition of the case is without visible abrasion when installed.	
Expected Outcome: No visible scuffs on case. Red LED is blinking to indicate error (such as a failure in recording/uploading). Green LED is stable to indicate recording. Green LED is blinking for calibration.	
Passed (Yes/No) :	
Notes on outcome:	

A.3 Computer Vision Testing

Test Name: Route Identification	Priority (Low/Med/High): High
Test Description: After initiating Calibration, EC2 must send the route data for routes identified by stickers to the database. It must transfer the image of the route, the colour of the holds, the bounding box coordinates of each hold, an associated sequence number, and the calibration data and time.	

Expected Outcome: Check: Correct number of routes, all rocks in route classified, start and end holds are correctly classified.
Passed (Yes/No) :
Notes on outcome:

Test Name: Height Classification	Priority (Low/Med/High): High
Test Description: From 5 test climbs of climbers at different heights, the EC2 must be able to identify the heights of each climber with an error of margin of up to 5cm.	
Expected Outcome: The webapp correctly displays each climber's height in a different bucket of a histogram which contains 10cm buckets.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Individual Climber Tracking	Priority (Low/Med/High): High
Test Description: From one test user climb, the EC2 must track and upload 1 individual climb's statistics to the database including which route was climbed.	
Expected Outcome: Database is loaded with all required run data for the individual climber.	
Passed (Yes/No):	
Notes on outcome:	

Test Name: Order of holds used	Priority (Low/Med/High): Low
Test Description: From one test user climb, the EC2 must upload the correct sequence of holds used in the climb to the database.	
Expected Outcome: The database correctly displays a list of holds used in the order in which they were used.	
Passed (Yes/No) :	

Notes on outcome:

Test Name: Route Success	Priority (Low/Med/High): High
Test Description: From one test user climb, the EC2 must correctly register if the climb was a successful attempt.	
Expected Outcome: Database displays the time-stamped attempt as a success.	
Passed: Yes/No	
Notes on outcome:	

Test Name: Route Failure	Priority (Low/Med/High): High
Test Description: From one test user climb that fails to reach the end-hold, the EC2 must correctly register the attempt was a failure.	
Expected Outcome: Database displays the time-stamped attempt as a failure.	
Passed: Yes/No	
Notes on outcome:	

Test Name: Simultaneous Climber Tracking	Priority (Low/Med/High): High
Test Description: With 3 test climbers in the frame of the video, the EC2 must correctly track each individual climb, and upload the associated processed statistics to the database.	
Expected Outcome: Database is loaded with three separate entries of the required run data for the individual climbers.	
Passed (Yes/No) :	
Notes on outcome:	

A.4 WebApp Testing

Test Name: Database connectivity	Priority (Low/Med/High): High
Test Description: The web server is able to read data from the MongoDB database	
Expected Outcome: The processed data is displayed correctly on the web app.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Microcontroller connectivity	Priority (Low/Med/High): High
Test Description: The web app will ping the microcontroller, and send a calibration command to the microcontroller.	
Expected Outcome: The web app receives a ping response from the microcontroller, and the calibration date in the database is updated for active routes.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Client connectivity	Priority (Low/Med/High): High
Test Description: The React JS app must be in connection to the web server. Open all pages of the webapp and ensure these pages are populated correctly.	
Expected Outcome: The React JS app is able to dynamically display received data from the web server without needing to reload a web page.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Login test	Priority (Low/Med/High): Med
Test Description: With a pre-assigned username and password, a user must be able to log in to the webapp	
Expected Outcome: The user is authenticated and the home page is displayed on their screen.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Navigation test	Priority (Low/Med/High): High
Test Description: Once a user is logged in, they should be able to navigate from each page to the others and back	
Expected Outcome: User remains logged in through navigating to all pages.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Updated Route Data	Priority (Low/Med/High): High
Test Description: Before the end of a day at midnight, the statistics displayed on the web app should be updated with the daily user data, including the height histogram of climbers and all other route specific statistics.	
Expected Outcome: The last updated date on the web app is accurate and all data from the day before is available to view.	
Passed (Yes/No) :	
Notes on outcome:	

Test Name: Status updates	Priority (Low/Med/High): Low
Test Description: The microcontroller must communicate to the web app whether it is recording video or uploading and the EC2 must communicate to the web app whether it	

is processing or not
Expected Outcome: UI elements displaying the microcontroller's status should be updated accordingly within 1 minute.
Passed (Yes/No) :
Notes on outcome:

Appendix B: Supporting Design Options

B.1 Hardware Options

The hardware components include the design choices in microcontroller, camera module, and casing.



B.1.1.1 Microcontroller Options

While many microcontrollers were capable of controlling a camera to record video, communicating wirelessly, and automation through a script, no option was as affordable or accessible as the Raspberry Pi Zero W.

All other Raspberry Pi Models capable of WiFi connections were larger, more expensive, and came with more processing power than was needed for our project. The raspberry Pi models that didn't come built-in with WiFi support required additional modules for WiFi which were an additional expense that could be forewent with the Pi Zero W.

The Arduino Nano 33 IoT was the lead competitor for the choice of microcontrollers. It is small, affordable, and comes with built in WiFi capability. However, it was unclear what cameras could be attached. The Arduino IDE was viewed as inconvenient compared to the Linux based operating system of Raspberry Pi. The documentation for camera projects was also found inferior to those available to Raspberry Pi users.

TABLE 14
Alternate Microcontroller Specifications

Microcontroller	Technical Specifications	
	Operating Voltage	3.3V [16]
	Clock Speed	48MHz [16]
	DC Current per I/O Pins	7mA [16]
	Input Voltage	21V [16]
	Power Consumption	unspecified
	PCB size	18mm X 45mm [16]
	Weight	5g [16]
	Cost	\$40 [17]
	Operating Voltage	3.3V
	Clock Speed	1.5GHz [20]
	DC Current per I/O Pins	16mA [21]
	Input Voltage	5.1V [20]
	Power Consumption	3W [21]
	PCB size	58mm x 88mm [20]
	Weight	49.89g [18]
	Cost	\$77.5 [19]

B.1.1.2 Microcontroller Decision

The Raspberry Pi Zero W was chosen for the Eye-bex proof-of-concept prototype. The Pi Zero W documentation guaranteed a camera to satisfy all our requirement needs. While slightly larger than the Arduino Nano 33 IoT, the Pi Zero W and the camera module were both available immediately in the Vancouver area. The Pi Zero W was also the cheapest option, which not only reduced the price of the final product, but enabled our group to purchase multiple boards to distribute amongst the team while working remotely, as well as having a backup in case of fatal failure of one of the microcontrollers in use.

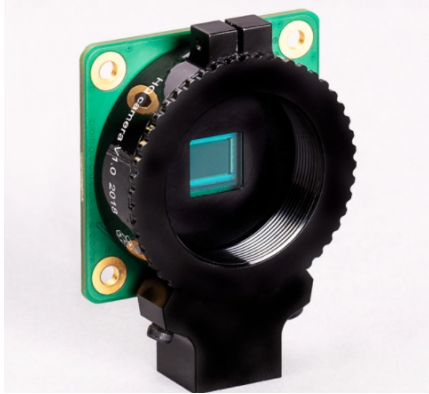
B.1.2.1 Camera Module Options


The camera module's function for this device is recording video at relatively high quality and frame rate. After the approval of the Raspberry Pi Zero W, several Pi camera modules were considered. None of the specialized lenses in the alternative Pi cameras for IR video, wide angle, fisheye, or ultra high resolution offered any advantage for our needs. The additional lenses also require a more expensive camera module, the Pi HQ camera, which would sizably increase costs.


Our initial design for the Eye-bex proof-of-concept demonstration is to use a single camera setup, but if that proves insufficient then we may upgrade to a stereo camera system, from Arducam. Though this was considered, the immediate availability of the Pi camera V2 module proved useful in getting development up and running.

The OAK-1/OAK-D cameras were also attractive options if we were to go for on-device processing of the videos. They were, however, very expensive. The OAK development kits are a very recent development and lack substantial documentation as well as previous projects to reference, which means debugging the product could take up a significant amount of time. The OAK cameras may not be scalable to multiple camera units should we need to capture especially obscured walls where multiple perspectives are needed, as each camera runs it's own image processing on device.

TABLE 15
Alternate Camera Specifications

Camera Module	Technical Specifications	
Raspberry Pi HQ Camera [22]	Physical dimensions (length x width x height)	38mm x 38mm x 18.4mm (excluding lens)
	Weight	Not listed
	Resolution	12.3 MP
	Sensor	Sony IMX477
	Max image quality	4056 x 3040
	Video Quality supported	<ul style="list-style-type: none"> • 1080p at 30fps • 720p at 60fps • 480p at 60/90fps
	Interface	Camera Serial Interface (CSI)
	OS Support	Raspberry Pi OS
	Horizontal field of view	Depends on lens
	Vertical field of view	Depends on lens
Arducam Synchronized Stereo Camera Bundle Kit [23]	Physical dimensions (length x width x height)	HAT size 65x56mm Camera board size 38x38mm

 <p style="text-align: center;">ArduCam®</p>	Weight	53g
	Resolution	12.3MP
	Sensor	2 x Sony IMX477
	Max image quality	4056 x 3040
	Video Quality supported	<ul style="list-style-type: none"> • 2028 × 1522 at 30fps with the Official driver • 4056*2 × 3040 at 6fps with Arducam driver
	Interface	Connects via HAT and CSI cable
	OS Support	Supported by Raspberry Pi OS
Oak-1 [24]	Horizontal field of view	65 degrees
	Physical dimensions (length x width x height)	32 x 58 x 108 mm

	Weight	42.52g
	Resolution	12 MP
	Sensor	IMX378
	Max Image Quality	4056x3040
	Video Quality supported	<ul style="list-style-type: none"> • 4K at 30 fps • 1080p at max 60fps • 720p at max 60fps • 480p at max 60fps
	Interface	USB Type-C
	Horizontal field of view	68.8 degrees
	Vertical field of view	81 degrees

B.1.2.2 Camera Module Decision


The Raspberry Pi Camera Module V2 was selected as the most affordable and available option amongst the camera modules capable of recording video at 720p 30fps. Video processing is being done on the cloud hence the need for an on-board processor like the OAK system is redundant. Both the Arducam and the Pi HQ camera use the same sensor, albeit the Arducam setup uses two of those sensors in parallel with a specific lens mounted on both sensors. They are, however, expensive modules and exceed our minimum requirements by a lot so it is hard to justify their cost.

B.1.3.1 Casing Options

The casing for the Eye-bex is required to protect the device, provide ventilation, and serve as a mount to attach the Eye-bex to the wall. It must also retain access to the ports and provide a space for LED status indicators. The options considered were limited and one of them included the standard Pi Zero case that although includes a hole to house a camera, does not provide any outlets for status LEDs to blink. Any other casing option apart from

the one selected and the generic case, would have to be an in-house design which could take valuable development resources away from the project. Another disadvantage of the Pi Zero Case is the exposure for unused pins which will invite climbing chalk, and there is no space for a fan. The Pi Zero case is also not customizable should our design change, whereas the 3D printable schematic is.

TABLE 16
Alternate Casing Specifications [25]

Camera Module	Technical Specifications	
Raspberry Pi Zero Case	Physical dimensions (length x width x height)	104 x 79 x 41mm
	Weight	30g
	Wall Mountable	No

B.1.3.2 Casing Decision

With the accessibility and customizability of 3D-printing, the Eye-bex proof-of-concept case will be the custom casing design of section 3.3.1. The design incorporates wall mounting, ventilation, I/O protection, space for a fan and status LEDs. The overall weight is well below the requirements, and a single screw can support the weight of mounting the device.

B.2 Computer Vision

B.2.1.1 Object Detection Options

Object detection technique is a combination of object localisation and classification. When it comes to object detection, FR-CNN, SSD and YOLO are the most popular ones. The

team has experience on YOLO made it easier to choose from. YOLO comes with different versions, among which version 3 and 4 are similar in performance. According to our research, for smaller objects the version 3 provides better results in less time as compared to version 4.

B.2.1.2 Object Detection Decision

We chose YOLOV3 as our object detection tool due to its speed and accuracy for smaller objects. A team member having experience with YOLOV3 added value to the decision.

B.2.2.1 Motion Tracking Options

YOLO providing jittery results for motion tracking made us look into further options of tracking. Two options were in consideration - Deep SORT and MotPy. One of the most popular tracking frameworks is DeepSORT, an extension to SORT(Simple Real Time Tracker). Whereas MotPy is a library that works well with YOLO to smoothen out the bounding boxes.

B.2.2.2 Motion Tracking Decision

We chose MotPy library for the motion tracking as it works well with the detection tool we chose. We considered the suggestions from TAs and decided to choose MotPy.

B.2.3.1 Cloud Services Options

We considered two options - cloud processing and on device processing. Cloud processing is fast, reliable and easy to scale as compared to on device processing. AWS ranks the best in cloud computing and platform service providers. Other popular choices are google cloud, oracle cloud and IBM cloud.

B.2.3.2 Cloud Services Decision

We chose the most popular option in cloud computing, Amazon Web Services, due to its ease of use, high performance and affordability. There is also a lot of documentation and resources available for learning.

B.3 Web App Options

B.3.1.1 Front End Options

The main focus of research for the frontend was finding a suitable framework that would suit the needs of a single page web application. Among frontend frameworks, the most popular ones are React JS, Angular JS and Vue JS which narrowed down our options down to three [26]. Among the three options, the team only has experience in using React JS. However, all three of these frameworks are widely used, heavily supported and come with a multitude of online resources.

B.3.1.2 Front End Decision

The React JS framework was chosen for the front end framework since the learning curve was found to be easier compared to the other options [26]. Considering that a team member has worked on a React application, we also hope to leverage this experience in creating our web app.

B.3.2.1 Server-Side Options

The server side options were mainly divided by Python-based web development frameworks and Node JS which is a JavaScript-based framework. For Python-based frameworks, we had researched Flask and Django. Both of these options can be used to build web applications however the latter comes equipped with more tools like an inbuilt ORM (object relational mapping). Django also follows a model view template (MVT) architecture [27]. On the other hand, Node JS is a runtime environment which gives us more options for Node JS based frameworks.

B.3.2.2 Server-Side Decision

With the flexibility that Node JS offers, we chose to use it as the server side framework for the web app as we found the MVT architecture in the Python-based options to be too rigid. This will also allow us to use JavaScript as the consistent language used in both the client and server side of our application. Along with Node JS, we will utilize Express JS as the backend framework to complete the MERN tech stack of the web app. Express JS was chosen as it can support single page applications and is known to be a standard framework used to develop Node JS applications [28].

B.3.3.1 Database Options

The database is required to store all the data for our product. The database must be able to hold various data types such as date, images (or reference to image), and have unique ids. All of the databases researched were able to fulfill the technical requirements for our database, and while there were many more options available, the outlined databases were chosen based on popularity. With a higher popularity we would have access support from online sources as well as compatibility with other technologies. Both of our other options were SQL databases, which while very structured and supported by many, would require us to build the database schema at the start and does not allow for flexibility further in the project if we want to change the data we receive/send.

TABLE 17
Alternate Database Specifications

Database	Technical Specifications	
PostgreSQL [29]	Database Type	SQL
	Price	Free
	License Type	Open Source
	Popularity	Very High
	Cloud Service	Various Service ex. Heroku, Google Cloud SQL
mySQL [30]	Database Type	SQL
	Price	\$2000 USD
	License Type	Proprietary
	Popularity	High
	Cloud Service	Heatwave

B.3.3.2 Database Decision

MongoDB Atlas was chosen as it was the most popular noSQL database available for free. We chose to go with a noSQL database as the document structure allows us to change the

schema on the fly. This will help the implementation of our product as we will be able to dynamically change what data is sent and retrieved from the computer vision processing. This allows us to implement the database features in parallel with the computer vision aspects as we can update the schema further in development. In addition MongoDB Atlas fulfils all our performance requirements and is supported by the other technology choices we made.

B.3.4.1 Deployment Options

For our deployment options, we needed a service that was cloud-based that is able to run, and monitor the code for our web-app. Cloud-services allow us to have greater accessibility which is a huge advantage during the COVID-19 pandemic and allows us to work separately on the same application.

For cloud-based services, there were 2 main ones that we found online. Amazon Web Services (AWS) and Heroku. AWS is an Infrastructure as a Service (IaaS) which means there are many services that can help in almost every situation. The downside, however, of using AWS is it is quite complicated to set up and requires someone experienced in development operations to get things going. Heroku is a similar option that requires deployment using git. Both these options had free versions that we were able to use and met all our performance requirements [31].

B.3.4.2 Deployment Decision

We chose to go with Heroku as our cloud based deployment as it is much simpler to set up with a lower learning curve. While AWS does contain more features and complexities, the requirements for the deployment of our product are quite simple and are all fully covered by what heroku can do. As they were both available for free, what really put Heroku over the edge as our choice is the simplicity to set up and ease of use.

Appendix C: User Interface and Appearance Design

C.1 Introduction

In order to make Eye-bex as accessible as possible and to enhance our user experience, the way our User Interface (UI) is designed is of the utmost importance. We want to ensure that our product is easy to understand and use. In appendix C we will explain the design

choice and considerations we made in regards to how our users will interact with the product.

In Appendix C we will cover the following topics:

1. **User Analysis:** Outlines the previous experience of our users and the knowledge and restrictions we expect them to have
2. **Technical Analysis:** Details how our UI will be designed around the “Seven Elements of UI Interaction” from Don Norman’s book “The Design of Everyday Things” [32]
3. **Engineering Standards:** Details the engineering standards we will adhere to
4. **Analytical Usability Testing:** Outlines the testing procedure done from the developers perspective and heuristic evaluations made for our UI
5. **Empirical Usability Test:** Outlines the testing that will be done with users of various iterations of the UI
6. **Graphical Mock-ups:** Visual representations of what our UI will look like

C.2 User Analysis

The target market for Eye-bex is climbing gyms, thus we would expect our users to be familiar with how climbing walls function and operate. Our product must be easy to learn and integrate into the gyms operation as we are aiming for our system to be autonomous with minimal input from the user. We will only have one customer-facing component, the web app. Installation of the hardware will be done by the development support team in person, and will require no input from the users besides where they want it set up.

The web application can be accessed online via our website. From the website users will be able to login with the provided login information given with the hardware. As our users may have different devices this website will be available for any device similar to a typical web page you would visit online. This means, however, that they would have to connect to the web page using wifi, which most gyms already have set up. This web-page will be simple to use for anyone with a basic proficiency with technology and web browsing.

C.3 Technical Analysis

In designing our product, we have taken into account the seven design elements that have been outlined by Don Norman in his book, “The Design of Everyday Things”. These elements focus on creating intuitive and easy to use computer interfaces that prioritize the user experience [32]. The following sections will showcase the different stages of design:

discoverability, feedback, conceptual models, affordances, signifiers, mappings and constraints.

C.3.1 Discoverability

Discoverability refers to how visible an element or a feature is in an interface. The more an element is discoverable, the easier it will be for a user to find it and use its features. If a feature is hidden away or less prominent to a user, it will be less likely that they will even know about it [32]. For Eye-bex, we have designed our product so that its key functionalities are prioritized and organized in a way that is easily visible to our users.

In terms of the web app, creating a simple interface will help users focus on the main features of the application and will eliminate a cluttered web page. One design element that we will utilize is a side navigation bar that will expose and link to different pages of the web app as seen in Figure 10 below. This allows us to highlight specific page views that we want to direct users to like the camera settings page, routes analytics page etc. Another reason to use a side navigation bar is that it also helps us avoid deeply nested web pages which can decrease visibility and create an unpleasant navigation experience. Additionally, styling elements in a web page can also help with discoverability as bright, contrasting or bold colors and text can be used to bring attention to buttons, forms and more.

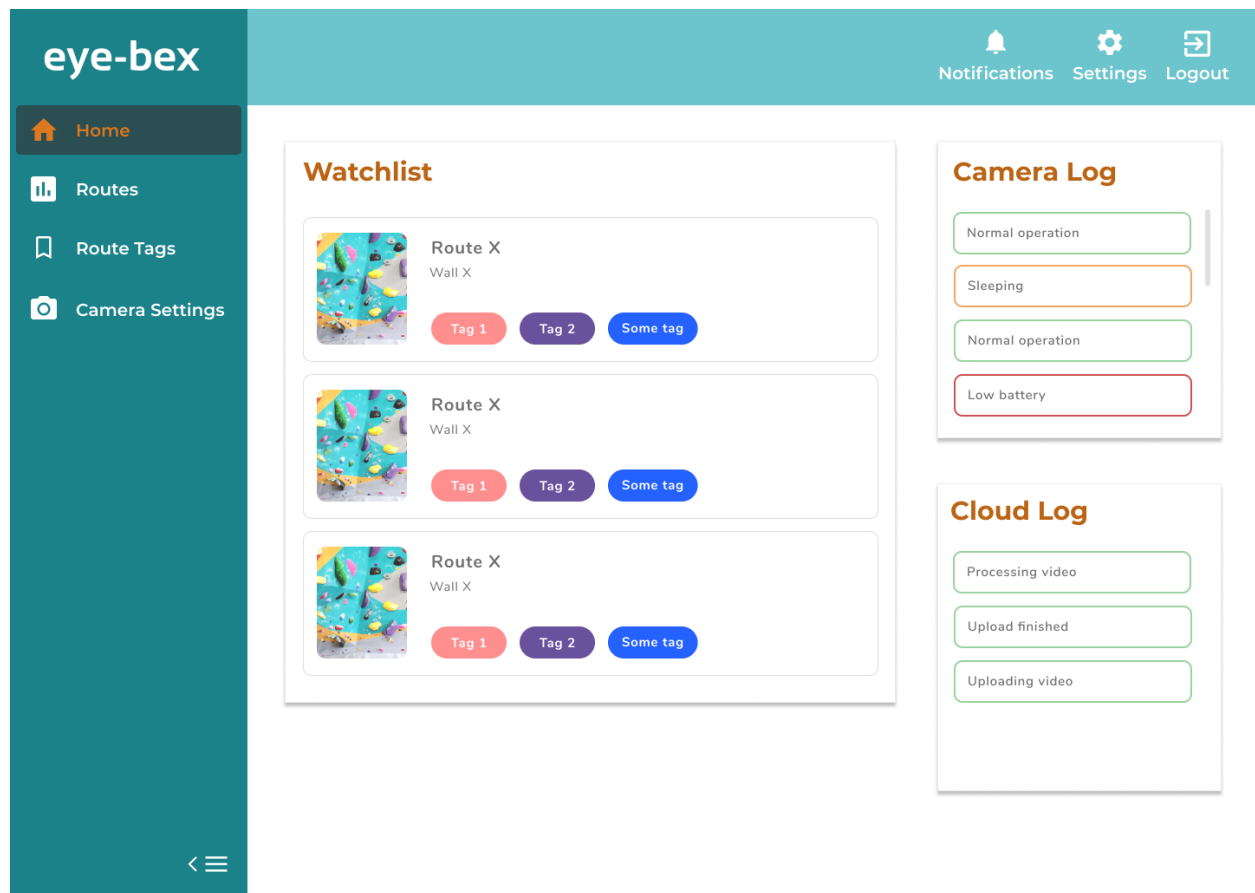


Figure 10: Home Page Mockup

C.3.2 Feedback

Feedback from a product allows us to communicate to the user what a certain action has achieved. This can be done by using visual, audio or tactile elements [32]. Giving feedback to a user action lets the user know that the Eye-bex system is aware that an action was taken and that it is being processed.

With the hardware camera system, its main form of feedback would be through the use of status LEDs. Eye-bex will use red and green LEDs to visually display the current status of the camera system where all the possible statuses are listed below.

1. Static Green: The Eye-bex camera system is recording video and is connected to WiFi
2. Blinking Green: The Eye-bex camera system is calibrating
3. Static Red: The Eye-bex camera system is low on memory storage
4. Blinking Red: The Eye-bex camera system is experiencing one or more of the following errors

- a. Memory storage is full and has stopped recording
- b. Uploading failure due to poor WiFi connection
- c. Camera failure

Similarly, the web app will also utilize visual elements to provide feedback. A simple use case would be a user clicking on a route's analytics and the feedback will act as some sort of confirmation that the web app is processing the request. This type of loading or waiting state will be represented by a loading spinner which is commonly used by most software applications. Another use case in which feedback is used would be when a user creates a route tag. To let the user know that a route tag has been added to a route, the web app will trigger a toast notification to confirm this action with the following message: "Route tag added". Toast notifications are small notifications that appear at the bottom or top of the screen which disappear after a short period of time and can be seen at the top right corner in Figure 11 [33]. This type of notification does not obstruct the current view of the web page while also letting the user know if a request went through or not.

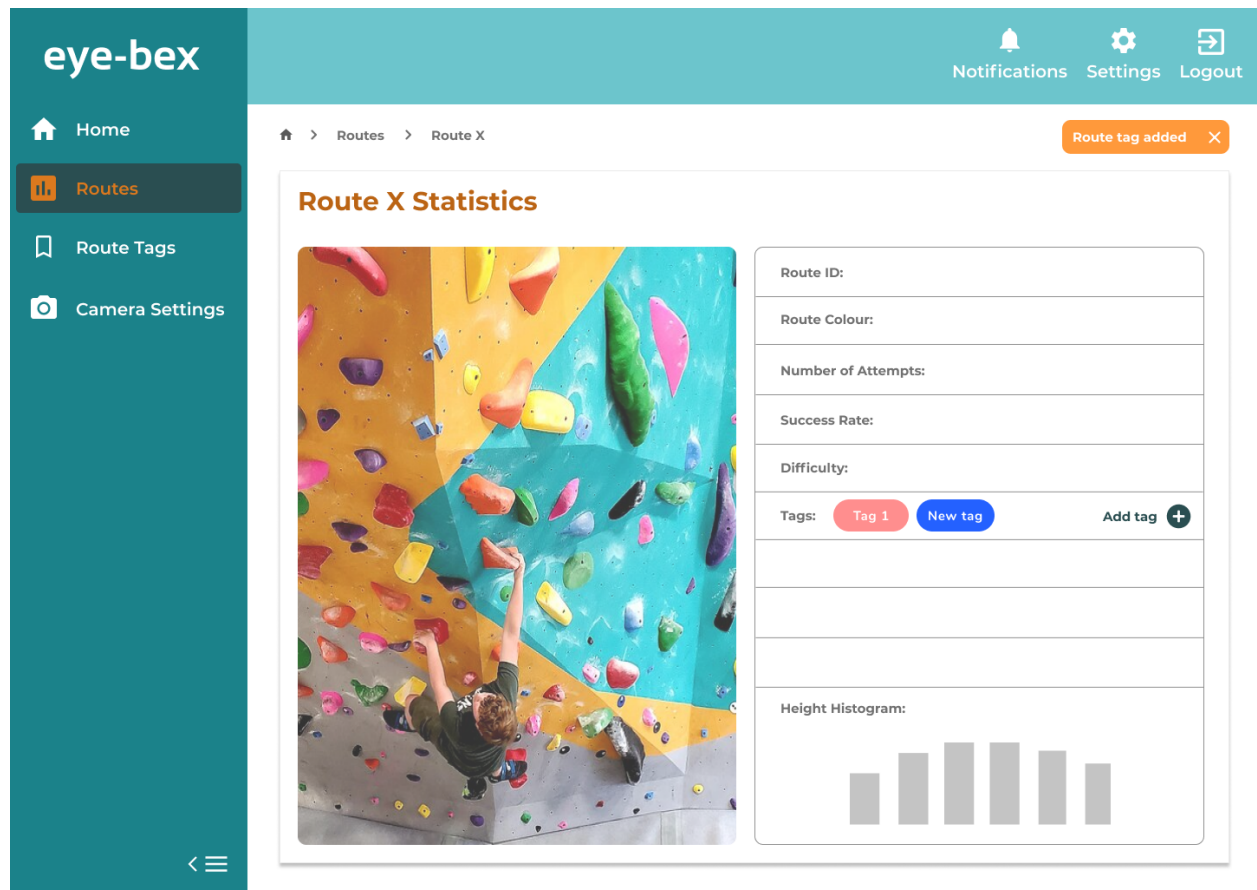


Figure 11: Route Statistics Page Mockup with Toast Notification

C.3.3 Conceptual Models

Conceptual models are developed by users to understand how processes work since it abstracts actual design elements with their perceived concepts and ideas which may be flawed [32]. To create a good system design, Eye-bex must help users develop the most accurate mental model on how the product works without needing all of the technical details. Users of Eye-bex will only need to interact with two of the three main systems, the camera and web app system. A user can easily create a mental model by observing that the camera records video footage and the route statistics are displayed in the web app. This completely leaves out the computer vision system that actually analyzes the video footage that the camera system recorded. To avoid this inaccurate representation of the Eye-bex system, we will display cloud processing status information in the web app to ensure that users know when the computer vision system is processing data and when it is finished. Though it is not a system that they interact with, users will be aware it exists and can develop a conceptual model that is most similar to the Eye-bex system.

C.3.4 Affordances

Affordance pertains to the perceived actions that can be made by interacting with a UI element [32]. By providing strong clues on how UI elements should be interacted with, we can aid users in how to use the web app and avoid confusing them with our interface. Some obvious affordances that are used in our web app are buttons and text fields. Buttons can show that a user can click on it while a text field affords text input provided by the user.

C.3.5 Signifiers

Signifiers work with affordances as the actual clues that provide the visual information perceived by the user [34]. These signifiers are what makes elements intuitive as it can determine how a user will perceive a UI element and how one would interact with it. As too much copy or text on web pages can lead to content overload, Eye-bex will use intuitive icons as a sort of visual aid along with most labels. For example, the universal gear icon will be used to indicate the settings menu in the web app which can be seen in the top right of Figure 11.

C.3.6 Mappings

Mapping is used to describe the relationship between system functionalities and its real world counterpart. Using physical analogies from the real world can help users understand web app functionalities without being explicit. From the web app, users have the ability to turn a camera on and off with a click of a button much like pressing a physical power button.

C.3.7 Constraints

Constraints are restrictions placed upon the users of the system to stop them from taking a certain action which may lead to undesirable consequences [32]. By enforcing constraints, we can safeguard the Eye-bex system from possible errors that can be caused by users of the system.

Such constraints that are enforced on our users relate to our camera and computer vision systems. By leaving the camera system set up to the Eye-bex team and providing on and off functionality on the web app, we can limit accessibility to the camera and avoid possible tampering from users to limit errors. Eye-bex also isolates the computer vision system from users by only sending status updates and ensuring that analyzed data cannot be modified by users.

One other constraint that Eye-bex will enforce is to only allow users to access route statistics that are associated with the gym they belong to. This can be done by creating an account management system such that the entry point in the web app would require users to login with the appropriate credentials as seen in Figure 13.

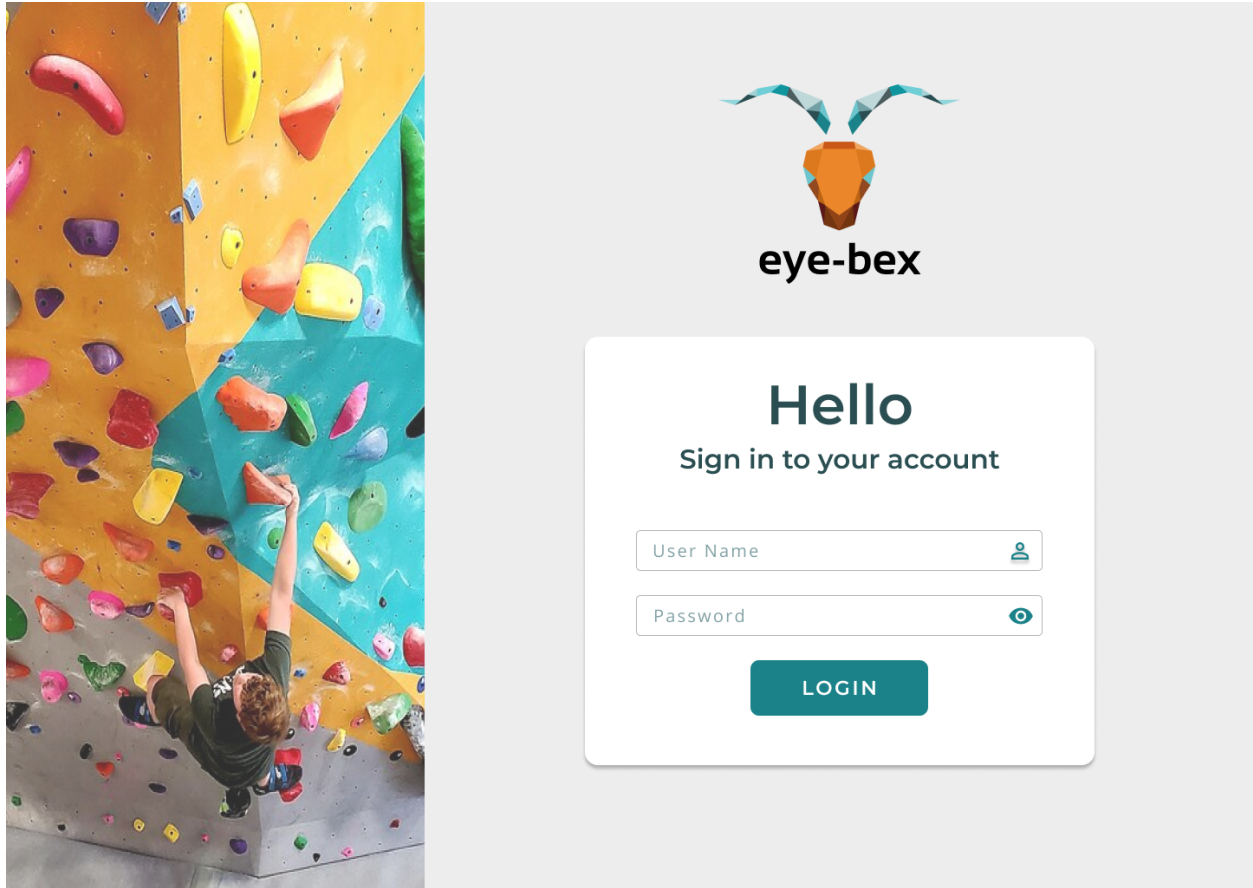


Figure 12: Login Page Mockup

C.4 Engineering Standards

Eye-bex will be following the standards set by CSA, IEEE and ISO. Below are the specific standards for the software and UI components of Eye-bex.

1. IEEE/ISO/IEC 23026:2015 [35]

Systems and software engineering - Engineering and management of websites for systems, software, and services information

Defines system engineering and management requirements for the life cycle of websites, including strategy, design, engineering, testing and validation

2. CAN/CSA-ISO/IEC 26557:18 [36]

Tools and methods of variability mechanisms for software and system

Defines processes and their sub-processes for operating variability mechanisms for software and system

3. ISO/IEC TR 12182:2015 [37]

Systems and software engineering - Framework of categorization of IT systems and software, and guide for applying it

Defines the manner in which categorization of IT systems and software are organised and expressed

4. ISO 9241-161:2016 [38]

Ergonomics of human-system interaction

Describes visual user-interface elements presented by software and provides requirements and recommendations on when and how to use them

5. ISO/IEC/IEEE 12207:2017 [39]

Systems and software engineering - Software life cycle processes

Provides processes that can be employed for defining, controlling, and improving software life cycle processes within an organization or a project

C.5 Analytical Usability Testing

This section will cover the analytical testing we will perform internally. The goal of the analytical usability tests is to perform in-depth testing to find inaccuracies, bugs and other errors that may hinder a user's experience. These tests will help us determine when the product meets the design goals and what changes are needed to reach them.

We will base the usability tests around the heuristics: learnability, efficiency, memorability, errors and satisfaction [40]. We will use a point-based system and assign a value for each of these on a scale from one to five, one being needs significant work and five representing exceptional. Through this scale, we can analyse which aspect needs more work and further investigate how we increase the score. We will test these heuristics on the following aspects of our webpage:

1. Login/Logout Authentication
2. Route Selection
3. Route Statistics
4. Route Calibration
5. Custom Route Tag Creation/Deletion
6. Account Status and Management

In addition to the heuristic based usability tests, we will also perform end-to-end tests performing the basic functions mentioned in Appendix A.4. These end-to-end tests give us a perspective into how each sub-system in our project will interact compared to the system as a whole. We will also include end-to-end tests to deal with error handling in which we test edge cases such as deliberately inputting false values and performing actions without proper authentication. These tests will help improve the security and reliability of our system.

The analytical usability tests will be performed prior to empirical tests. Performing the usability tests in this order allows us to ensure that external testers will be able to provide feedback more concisely and effectively as there will be fewer errors.

C.6 Empirical Usability Testing

Empirical Usability Tests will be performed by end users without prior knowledge of our product. These will be end-to-end tests that observe how new users may interact with our web app and how intuitive the web app is to use. During these tests, developers will keep track of certain metrics specified for each test such as time taken and actions done to gain information used to analyze our UI. In addition to feedback from users which will be asked after each test, the empirical usability tests will allow us to gain insight on the UI from an outside source to help us achieve our goals of having an intuitive and effective user interface. Below are the lists of tests we will have users take and the feedback we expect in return.

Test Name: Login Page
Developer Prompt: Login to the web application (provides login credentials)
Expected Outcome: User is able to navigate to the web page and login using a given credential
Passed (Yes/No) :
Time Taken:
User Feedback:

Test Name: Navigate to the Statistics Page of a Route
Developer Prompt: Navigate to the statistics page of any active route
Expected Outcome: User is able to navigate to Active Route Selection and then navigate to its statistic page
Passed (Yes/No) :
Time Taken:
User Feedback:

Test Name: Statistics Readability Tests
Developer Prompt: State 4 statistics of the route
Expected Outcome: User states any of the following <ul style="list-style-type: none"> - Route ID - Color - Number of Attempts - Difficulty - Percentage Success - Average Time Taken - Height Histogram
Passed (Yes/No) :
Time Taken:
User Feedback:

Test Name: Create a Tag
Developer Prompt: Create a custom tag called "Test Tag"
Expected Outcome: User is able to create a custom tag
Passed (Yes/No) :
Time Taken:
User Feedback:

Test Name: Add Tag
Developer Prompt: Add a custom tag “Test Tag” to a chosen route
Expected Outcome: User is able to add a custom tag to a route
Passed (Yes/No) :
Time Taken:
User Feedback:

Test Name: Calibration Test
Developer Prompt: Calibrate the camera for the whole wall
Expected Outcome: User is able to navigate back to the home page and press the “Calibrate” button
Passed (Yes/No) :
Time Taken:
User Feedback:

C.7 Conclusion

The User Interface design is an integral part of Eye-bex’s success. With our goal to make Eye-bex as accessible as possible, the web-app UI and customer-facing hardware components must be easy to use for a commonplace user. To ensure our UI is able to keep up with our goal, we took into consideration Don Norman’s philosophy of “Seven Elements of UI Interaction” as well as several engineering standards to design mock-ups of the web-app. As we develop the product the UI will be tested using both analytical usability testing internally and empirical testing with our clients. For the proof-of-concept we plan on having met all our UI standards and tests. For any further features added in alpha and beta phase, as well as adjustments during development, we expect to keep up with our tests and standards as we dynamically run through different iterations and gain feedback from our empirical usability tests.