Compsci 399 Capstone Semester 1 2024

Team 14 Neox

Cloud Connected Light Final Report

02.06.2024

Name	Roles
Lucia Kim	Team Lead, App Developer, ML Engineer
David Auna	Embedded Systems Developer, Web Developer
Gabriel Lisaca	Server Lead
Hrichik Sircar	App Lead
Kevin Lu	Embedded Systems Lead, App Developer
Lewis Choi	Web Lead

Executive Summary

Myopia - also known as near-sightedness - is one of the most widespread eye conditions affecting ~34% of the global population and expected to affect almost 50% by 2050 (Holden et al., 2016). The earlier myopia occurs in a child's life, the more likely it will develop into high myopia or extreme near-sightedness. To mitigate the progression of myopia in children, it is paramount to increase their daily exposure to bright light as well as their time spent outdoors. The recommendation is two hours of outdoor activity everyday. However, as children spend most of their day at school while parents attend work, it can be challenging for parents to keep track of their child's time spent outdoors.

Our main goal was two-fold. To help parents keep track of their children's outdoor time to help slow the progression of myopia and to help researchers gather valuable data to help in future research. The first goal was completed by the Neox Sens device and the Neox app. The device is used by the child to gather data from a variety of sensors which is sent to the app. The parents can then use the app to keep track of their child's outdoor time through statistics and graphs. They can also decide if they want to contribute to research in myopia. If they do, they can send data to the server which the researchers are then able to access via our website.

We adopted a free-form workflow based on agile concepts such as sprints and regular meetings. We ensure that we distributed the workload fairly and continuously provided value to clients as scheduled. We were also able to identify and adapt to evolving client needs and unforeseen problems.

The four components of our final product, the Neox Sens, Neox App, Neox Cloud and Neox Labs, were all completed to a high standard. We accomplished all our goals. We were also able to devise more features that could be implemented in the future, should the clients decide to carry on development of our product in the future.

Table of Contents

Introduction

Aims and Objectives

Our project aims to develop the Neox Sens, a wearable light monitor prototype based on an Arduino, which can assist in understanding the correlation between light exposure patterns and myopia development in children. The collected data can be synchronised with a cloud database through a dedicated mobile phone application. This application allows parents to monitor and analyse their child's light exposure patterns. Additionally, researchers are able to create studies that parents can opt into, allowing researchers to access and analyse their data. By providing this system for monitoring and analysing users' light exposure patterns, we expect it to contribute to slowing the progression of myopia for children.

Our aims are to:

1. Use light sensors that measure light intensity, including UV, over time to determine the user's light exposure amount and the duration spent outdoors.
2. Transmit the collected data to a dedicated mobile application via Bluetooth Low Energy for ease of use and improved power efficiency.
3. Display graphs in our mobile application, giving parents an overview of their child's outdoor time.
4. Allow researchers to set up studies through a web interface that parents can opt into.
5. Allow researchers to download data for specific studies in CSV file format through the web interface.

Target Audience

Our target audiences are:

Parents of children who are at risk of developing myopia

We aim to offer a non-obtrusive, inexpensive, and behavioural solution to the parents who are concerned about near-sightedness in their children. As well as targeting parents who have children with myopia, we also want to offer Neox as a preventative measure to parents with non-myopic children.

Myopia researchers

Our second target audience are myopia researchers who want to collect data for insights into the myopia development process. We aim to contribute to myopia research by facilitating the data collection process.

Scope of the Project

Our scope of the minimal viable product is setting up the data collection pipeline.

1. Hardware device that detects light intensity to classify indoor and outdoor environments
2. App which syncs sensor data from the hardware device to a cloud database
3. Website for researchers to download the data

We have offered further extensions to the MVP to offer more value to the target audiences.

1. Additional hardware sensors: acceleration, red green blue clear chromatic components of light, uv sensor
2. Local machine learning classification on app
3. Participate in studies on app
4. Web portal for researchers to create and manage studies, as well as other researcher accounts
5. Download sensor data based on study

Approach Used

We initially planned to use a Kanban board through the Agile methodology to develop our project. For this, we created a list of tasks and organised them into weeks that we would have them done by. Our general goal was to set up each component and have them communicating with each other by the first few weeks. Then we aimed to have data analysis done shortly after, and focus on security, user interaction, and refinements in the final weeks. However, as we progressed, we diverged from the Kanban approach and took a largely freeform approach. To make this work, we had weekly team meetings to discuss our progress and plan the next week's goals, ensuring that we were not falling behind. We also had weekly meetings with our clients to ensure that their requirements were being met. Although the principles of Agile methodology was not strictly followed, our group aimed to deliver continuous value to the customers. For each fortnight of the project duration, we focused on delivering these values:

Sprint 1: Preparation (Week 3-Week 4)

Sprint 2: Project Architecture (Week 5-Break Week 1)

Sprint 3: Sensor Data Pipeline (Break Week 2 - Week 6)

Sprint 4: User Interaction (Week 7 - 8)

Sprint 5: Research Pipeline (Week 9 - 10)

Sprint 6: Refinement (Week 11 - 12)

Important Outcomes

We have achieved significant goals across the four domains of the project. We have achieved significant outcomes to reduce the progression of myopia in children, and contribute to myopia research for academia.

Neox SeSns

The Neox Sens implements multiple sensors needed for indoor-outdoor classification. The wearable device has a compact and lightweight form factor. The low-power hardware device offers robust Bluetooth LE functionalities to sync with Android and iOS. We have achieved a high standard of security through our unique security protocols.

Neox App

Through the Neox App, we have allowed parents to seamlessly monitor their child's outdoor activity. They can know exactly how much time the children spend outdoors through our dedicated hourly, daily, weekly, monthly, and yearly views. We offer engaging interactions with the app, so that the parents are incentivised to encourage the child to go outdoors.

Neox Cloud

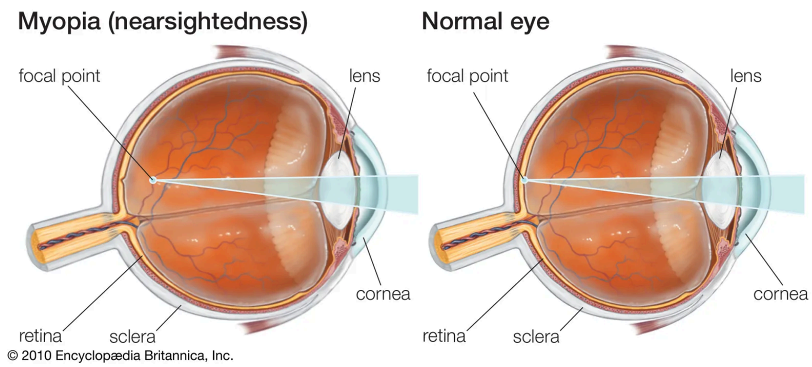
The Neox Cloud offers robust security features to safely store the data. The AWS server manages services for databases and authentication. The parents can backup the data to the app. The AWS server offers serverless hosting of the website.

Neox Labs

The Neox Labs is offered to accelerate myopia research. We have allowed researchers to collect the data from parents through an opt-in method. All interactions are performed securely, and parents can be sure of where their data is going.

Background and Rationale

Myopia occurs when the light entering the eyes is focused prior to reaching the retina, leading to blurred long-distance vision. Myopia begins in early childhood between the ages 8 to 12. The eyeball elongates during the ocular development process during the teenage years. This increases the axial length out of proportion, causing the most common type of myopia: axial myopia (Recko & Stahl, 2015). In New Zealand, 29.8% of patients visiting aged 18 years and younger are diagnosed with myopia (Ashby, 2023).



Note. From *Myopia*, by R. Lewis, 2024 (https://www.britannica.com/science/myopia). Copyright 2024 by Encyclopedia Britannica.

Early intervention is important to slow the progression of myopia. Although the blurred vision of myopia can be corrected with glasses, the abnormal eye structure leads to higher risk of severe eye conditions later in life. These include open-angle glaucoma, cataract, retinal detachment, and retinal tears (Williams & Hammond, 2019). Furthermore, myopic macular degeneration has been noted as the most frequent cause of blindness in cross-sectional studies conducted in Taiwan, Hong Kong, Japan, and the Netherlands (Holden et. al, 2013).

There are several clinical interventions in New Zealand for myopia, including atropine eye drops, corrective hard lens, and red light therapy. However, these solutions are expensive and impractical. Atropine eye drops have side effects such as irritation and light sensitivity. Contact lenses increase the risk of eye infections. Red light therapy is a novel treatment, and the long term safety issues have not been properly explored. Furthermore, the clinical interventions require a high level of compliance from the child. Considering that the onset of myopia begins at a young age, the best interventions are through environmental and behavioural modifications. Several non-genetic factors have been identified as accelerating the progression of myopia, such as reduced outdoor light exposure, increased near work, and reduced physical activity. Among these, outdoor light exposure has been identified as the most significant mitigating factor (Recko & Stahl, 2015)

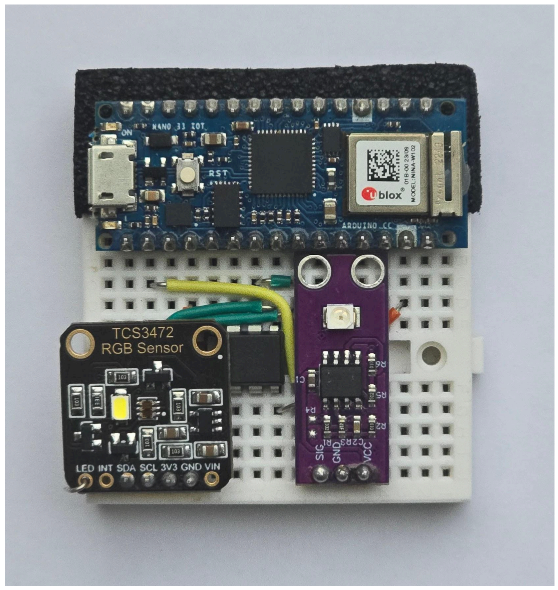
Outdoor activity is the greatest measure in reducing the risk of developing myopia and slowing the progression in children. Each additional hour spent outdoors per week reduced the probability of myopia by 2% (Sherwin et al, 2012). Although the exact biological mechanism is unknown, the most credited hypothesis is that the protective effect of outdoor exposure comes from the light intensity. When the eyes are stimulated by bright light, the eyes increase the level of retinal dopamine and activation of dopamine receptors. Dopamine limits axial myopia development by inhibiting the eyes from elongating. The chromatic composition of sunlight is also hypothesised as slowing the progression of myopia. Sunlight has a high ratio of blue light, which is focused closer to the lens. The eyes develop hyperopic refraction, which counters myopia.The synthesis of vitamin D through sunlight is another contributing factor. Vitamin D adjusts the smooth ciliary muscle, which regulates the length and refractive degree of the eye (Zhang & Deng, 2020). Outdoor activity is important as sunlight reduces the progression of myopia through light intensity, chromatic composition, and vitamin D synthesis.

Although the recommendation is two hours of outdoor activity everyday, it is difficult to utilise the intervention in practice for young children. Primary children have less structured hours at school, which makes it difficult to accurately quantify the amount of time spent outdoors. It is also impossible to measure the time the children spent outdoors in between classes from recall alone. More objective measures must be used.

Several mobile apps use the GPS and cellular strength sensors in the mobile phone to detect if the person is indoors or outdoors. However, this is impracticable for young children who are likely to not have mobile phones. Even when the child carries the mobile phone, the accuracy decreases when the phone is put inside the bag or in the pocket. Privacy is also a concern. It is not safe for children to be tracked with a device that has wireless connection. Wearable detectors have been developed overseas, such as the HOBO Pendant and the Actiwatch. However, these solutions are not for the purpose of quantifying outdoor time. The HOBO Pendant is used for temperature and light monitoring applications, and the Actiwatch is used to monitor sleep patterns.

Specification and Design

Collecting Data with the Neox Sens and Syncing to the App



The Neox Sens consists of an Arduino Nano 33 IoT microcontroller, colour light sensor, UV sensor, inertial measurement unit (IMU), real time clock (RTC) and an EEPROM (electrically erasable programmable read-only memory) for storage. The colour light sensor detects the received light intensity and separates it into red, green, blue and clear values. These are then used to calculate the intensity of lux received. The UV sensor is used to detect the intensity of incident ultraviolet light. The IMU and RTC reside on the microcontroller itself. We utilise the accelerometer on the IMU to detect the device's acceleration of motion. These sensor readings are useful for the machine learning model to classify if the user is indoors versus outdoors. The RTC records timestamps in UNIX time each time the sensors are polled and assigns this timestamp to each set of sensor readings.

As it is designed to be a wearable device for children, we decided it must be compact enough so as not to hinder or cause discomfort. We have chosen our sensors and microcontroller based on size, so as to meet this requirement. We have also based our set up on this requirement. The device is mounted on a small breadboard connected with minimalistic wiring to further minimise size.

The clients specified the need for a large quantity of data so it is useful for research and future studies. Thus, the sensors are polled every minute so as to provide this useful quantity. The clients also specified the importance of having the raw data from the sensors saved, as these would be needed for their research. All raw data from the sensors and their associated timestamps are stored on the 32KB EEPROM. This data is then sent to the app via the microcontroller's built-in BLE module.

To transfer the sensor data to the phone app, we use Bluetooth Low Energy (BLE) to communicate between the Neox Sens and the phone. BLE is designed for minimal power consumption, making it suitable given the strict power requirements of a wearable device. BLE advertisements are broadcast every 100 milliseconds, allowing the app to easily connect whenever the Arduino is nearby.

However, BLE comes with privacy implications. We do not want it to be possible for arbitrary people to read the sensor data which contains timestamps that can be used to track the child's behaviour. Therefore, the Neox Sens uses a unique authentication protocol to verify incoming connections and to prove its own identity using a shared secret. This shared secret is a 10-digit code printed on a piece of paper shipped with the Neox Sens and is fixed at production time. After entering this code into the app, the parent can retrieve the data which will then be stored in a central SQLite database on the app. Currently, we limit each child to be associated with one device at a time. However, parents have the option to reassociate the device at any time.

Local Classification of Outdoor Environments

In the Neox App, the data received from Neox Sens is classified into indoor or outdoor environments. Machine learning techniques have been implemented for accurate and comprehensive classification. The machine learning model was trained locally through Scikit-learn in Python. The final model was then converted to Dart to be able to predictions on the local flutter app

The clients emphasised on having low false positive classification rate. The sample collected must be assumed to be indoors, unless proven to be outdoors. Although some outdoor time may be classified as indoors, the clients prioritised on having indoor time not misclassified as indoor time. Accommodating to the requirement, the Area Under the Curve (AUC) measure was used to identify the best model. The AUC measures how well the model correctly classified the outdoor samples, in comparison to how well the model does not classify indoor samples as outdoors. The AUC considers the true positive rate, compared to how many negatives are correctly labelled. The AUC compromises that outdoor samples can be incorrectly identified as negative, to favour all the indoor samples correctly classified as positive (specificity). The clients were happy to compromise how much outdoor time was detected over incorrectly classifying indoor time as outdoor time.

SIx different models were trained using 2974 outdoor samples and 1672 indoor samples. The samples were collected during May 2024. The samples were collected from a variety of environments. Particular attention was given to classifying semi-indoor and semi-outdoor environments. The indoor samples range from fully artificially lighted rooms to near bright windows with full sunshine. The outdoor samples range from open grass fields to under cover of thick shade.

The Arduino sensor collects uv, acceleration, light chromatic components, colour temperature and lux. Furthermore, 40 additional features were engineered to train the model. Each model was tuned so that the best hyperparameter combination was selected. Cross validation was implemented to avoid overfitting. A separate validation set and test set from unknown environments, which includes samples collected from a bright lit windowsill, were used to determine the final model. The LightGBM classifier was chosen.

Statistical Analysis and Engagement

The Neox App offers comprehensive statistical insights into the amount of time the child has spent outdoors, as well as gamification to engage usage by parents.

The home tab displays a simplified overview of the outdoor time. Animated graphs display the outdoor time spent today as well as the time over the past week and month. These graphs fill according to a customisable target which is set to 120 minutes per day by default. Once the target has been reached, the parent is rewarded with a gold graph.

On the statistics tab, we have two views to allow parents to view their child's outdoor time in varying levels of detail. The daily view displays a bar graph showing the number of minutes spent outside each day for the last seven days. The bars are clickable and will show a detailed hourly breakdown for the relevant day in another char graph below. Hovering on these bars will display a tooltip showing exact metrics, and it is also possible to swipe sideways to view data from previous weeks.

Similarly, the monthly view displays a bar graph showing the average time spent outdoors per day on a monthly basis over the last year. Again, the user can view previous years' data and hover over bars to view exact metrics. As a part of our gamification goal to encourage parents to continue using our app, we have included a calendar in this view which displays the days where the daily target was reached. For each day that the target is achieved, the user is rewarded with a stamp which we hope will incentivise the user.

Cloud Syncing

The cloud page on the navigation bar is for features related to posting data to external sources. Users will be prompted to sign in or create an account when clicking on the page. This will sign them in using AWS Cognito or registering an account with the Cognito User pool. Once signed in, users will be shown a few options, such as syncing their data to the cloud or joining a research study. If a parent chooses to sync the data to the cloud, the phone will take all samples from the database and send them to the server. The parent will also receive existing child profiles and their relevant data synced to Neox Cloud. Once parents decide to join a research study, their children's data can be used by researchers to study myopia.

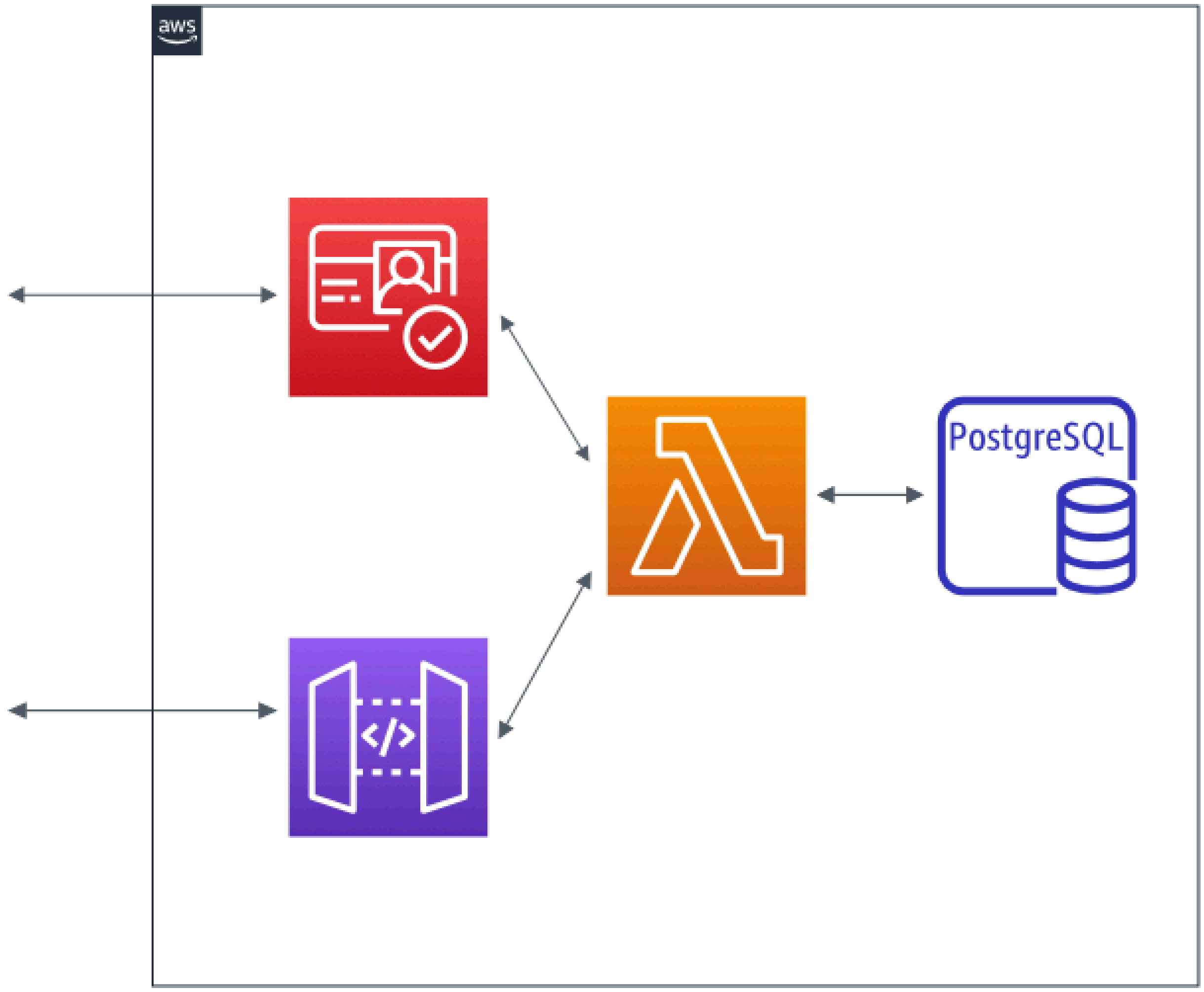
In addition to the app which helps parents see the amount of time that their children spend outdoors, our project requires relevant researchers to be able to download the samples collected by the Neox Sens using a web interface so they can analyse the data.

Both the app and the web interface must each send and receive samples from the cloud. From this, we can split users of this system into two main categories—parents and researchers—which must be recognised by our solution. It must also be easy to collaborate with other researchers who may be from the University of Auckland like our clients, or they may be from outside the university.

To provide for this, we define and implement an API for the app and the web interface to send and receive samples. It is REST-based so it has a uniform interface by reusing HTTP semantics which includes the HTTP methods GET, POST, PUT, PATCH, and DELETE; HTTP status codes; and the use of URIs as a representation of resources on the server. Each API action is orthogonal to ensure a clean separation between the server and API clients and thereby allow the server and clients to evolve independently. This helps reduce the amount of code needed on the server since the API actions only need to implement one task and to do it well. For example, the action to send samples to the server *only* sends samples and nothing else. This is simpler and therefore also faster.

We also use “serverless” architecture to reduce the maintenance burden during and after development because the service provider manages and provisions servers and we just write the code to glue services together. To do this, we use AWS since it is used widely in industry and thereby allows our “server” to have a large pool of developers and system administrators familiar with it after we finish development. We use:

- Cognito for authentication
- Lambda (NodeJS) for serverless code execution
- API Gateway to manage and execute API actions in response to HTTP requests
- RDS (PostgreSQL) for a database we do not have to manage



A good side effect of using these AWS services is that each individual component is production-grade and secure if configured correctly. Here we see clients interact with Cognito directly so we do not need to reimplement the interface for secure authentication and thereby better leverage the accumulated expertise of Cognito. Additionally, serverless components scale up much easier than monolithic applications since we can automatically run more instances of lambda functions and easily set up more database mirrors in response to spikes in requests.

As for collaboration with external researchers, one option is to send CSVs of the samples by email. This is simple, familiar to researchers, and requires no work on our end but puts the burden on them to periodically share the data rather than simply adding their collaborators to the web interface so they can download it when they need it. Therefore, we provide that web interface for researchers to add collaborators and download samples sent by parents via the API.

Myopia Research

Signing up for Studies

To protect children's privacy, we do not connect the Neox Sens directly to the API and instead have it communicate to the app which the parents have control over. The parents have full control of whether they choose to send their children's sample data or not, and whether they consent for that data to be used by researchers.

One option is for parents to consent to specific researchers having access to their children's sample data, but this has problems with scaling in which it would make collaboration harder as any new researchers would have to be specifically approved by every parent. By contrast, if parents instead consent to studies and not specific researchers, this scaling issue is resolved. This second option also mimics how real research is done in which participants consent to specific studies. A good side effect of this is that our solution could also be used for multiple different research projects simultaneously and on the same infrastructure because access to data in different studies is protected.

As a consequence, a simple split of users as parents and researchers is no longer possible. We now need administrators to ensure that only researchers authorised for specific studies have access to them. Since our clients are also researchers themselves, we can simply opt to have parents, researchers, and admin-researchers. But to allow the system to evolve to have pure admin users (non-researcher admins), we choose to have the three-role split of parents, researchers, and admins. In our project, "admin" users would simply have both the researcher and admin roles but this can be easily changed in the future.

Parents can sign up to participate in research after creating an account through the Neox app. Parents can choose to join a study via a study code and sync their data to the cloud for research purposes. On the website, only administrators have access to features for assigning researchers to studies and creating studies and researcher accounts, while non-administrators are restricted to accessing authorised studies and downloading data of the studies. The users can receive and send data by using features on the website which utilise API actions containing authentication information which is verified by Cognito and the cloud backend.

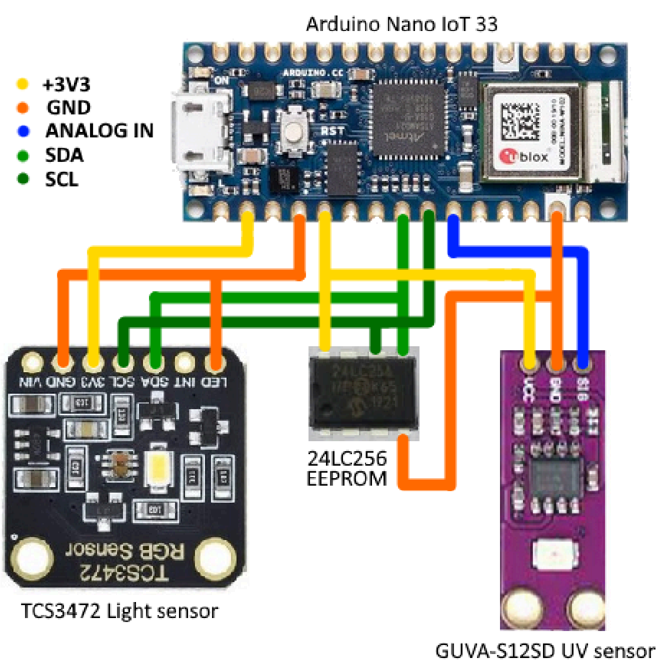
Project Implementation

Hardware

Arduino Development Environment

To develop for the Neox Sens, we used Arduino IDE due to its popularity and low learning overhead. Arduino IDE has an extensive variety of libraries available and setting them up for our project was trivial. Unfortunately, Arduino IDE does not have a standard way to list dependencies, so we had to list these in our readme file manually.

Hardware



We have chosen to use the TCS3472 digital light sensor to measure the environment's light. This sensor can detect varying red, green, and blue light levels, as well as an independent clear channel. From this information, we derive lux and colour temperature values to assist the indoor/outdoor classification. To support the colour light sensor, we use the GUVA-S12SD analogue UV sensor. Since UV-B cannot pass through windows, this UV sensor is used to prevent false positives in cases where bright light shines into indoor environments. This sensor is not perfect however, as it is sensitive to the 240-370nm light spectrum and responds to some of the UV-A spectrum. Additionally, we have the 24LC256 EEPROM to store sensor readings which are polled every minute, and miscellaneous settings. This EEPROM has a capacity of 32 kilobytes which is enough to store approximately one day's worth of data at 20 bytes per sample. The limited capacity is not a problem as this is a prototype and the capacity is trivial to increase in production.

Wire Routing

As a wearable device, it is important for the Neox Sens to be compact. As shown in the diagram above, we have three components connected to the Arduino each requiring their own +3.3V and ground connections. To simplify the routing, the A0 and A1 GPIO pins are used to provide +3.3V and ground sources and achieve a compact layout. The SCL and SDA wires form the multi-slave I2C bus for digital communication, and the UV sensor is connected directly to the A6 analogue input pin.

EEPROM

The storage on the EEPROM is designed to be efficient and robust. Since it is possible for a write to fail halfway through due to power loss, we have implemented the idea of an *atomic transaction*. Once an atomic transaction has started, the transaction must finish before any other reads or writes occur even between power cycles. This is implemented with three registers, address, value, and pending, which contain the information needed to resume a half-successful 4-byte write. To store the sensor samples, we use a circular buffer which allows us to have time complexity for appends as well as automatic overwriting over old data. The data layout is shown in the table below.

EEPROM Address Space (0x0000-0x7FFF)		
Offset	Size	Function
0x0000	0x0004	Atomic transaction address
0x0004	0x0004	Atomic transaction value
0x0008	0x0004	Atomic transaction pending
0x000C	0x4000	Sample buffer
0x400C	0x0004	Sample buffer tail index
0x4010	0x0004	Sample buffer length
0x4014	0x0020	Authentication key
0x4034	0x0004	RTC time
0x4038	0x3FC8	Unused

BLE Authentication Protocol

Authentication is a key part of the BLE protocol. Before any sensitive data is transmitted from the Neox Sens , it must verify the identity of the incoming connection. Normally, this is handled by BLE with 6-digit pairing codes, but since the Neox Sens does not have a display or any buttons, the only option is to use an out of band key distribution method. This is why we require parents to enter a code into the app. The authentication protocol works by verifying that the Neox Sens and the app both share a common key. To do this, both sides challenge each other by requesting the hash of the key combined with a securely generated random number. This means that the key itself is never transmitted, and the random number ensures protection against replay attacks. Clients are also required to authenticate within 10 seconds to mitigate the possibility of a denial-of-service attack.

BLE Structure

The BLE protocol has a total of five data buffers. All five data buffers are structured in the same way. The first 4 bytes are occupied by the timestamp, followed by 2 bytes for UV sensor data, then 6 bytes reserved for acceleration on the x, y and z axis. The next 8 bytes are allocated to colour light sensor data, with 2 bytes reserved for red, green, blue and clear values. The bytes are sent in little endian byte order.

When a sync is initiated by the app, these buffers are filled with sensor data from the EEPROM. They are then assigned to one of five BLE characteristics which are read by the app. A characteristic is a GUID-byte array key-value pair that BLE uses for communication. BLE does not use streams. Once the app has received the data, it can write to the update characteristic. This notifies the Neox Sens that the app has received all five data buffers and it is ready to receive more. When this characteristic is written to, the Neox Sens knows to read the EEPROM from the next index and fill the buffers with new samples. This process is repeated until all required data is sent.

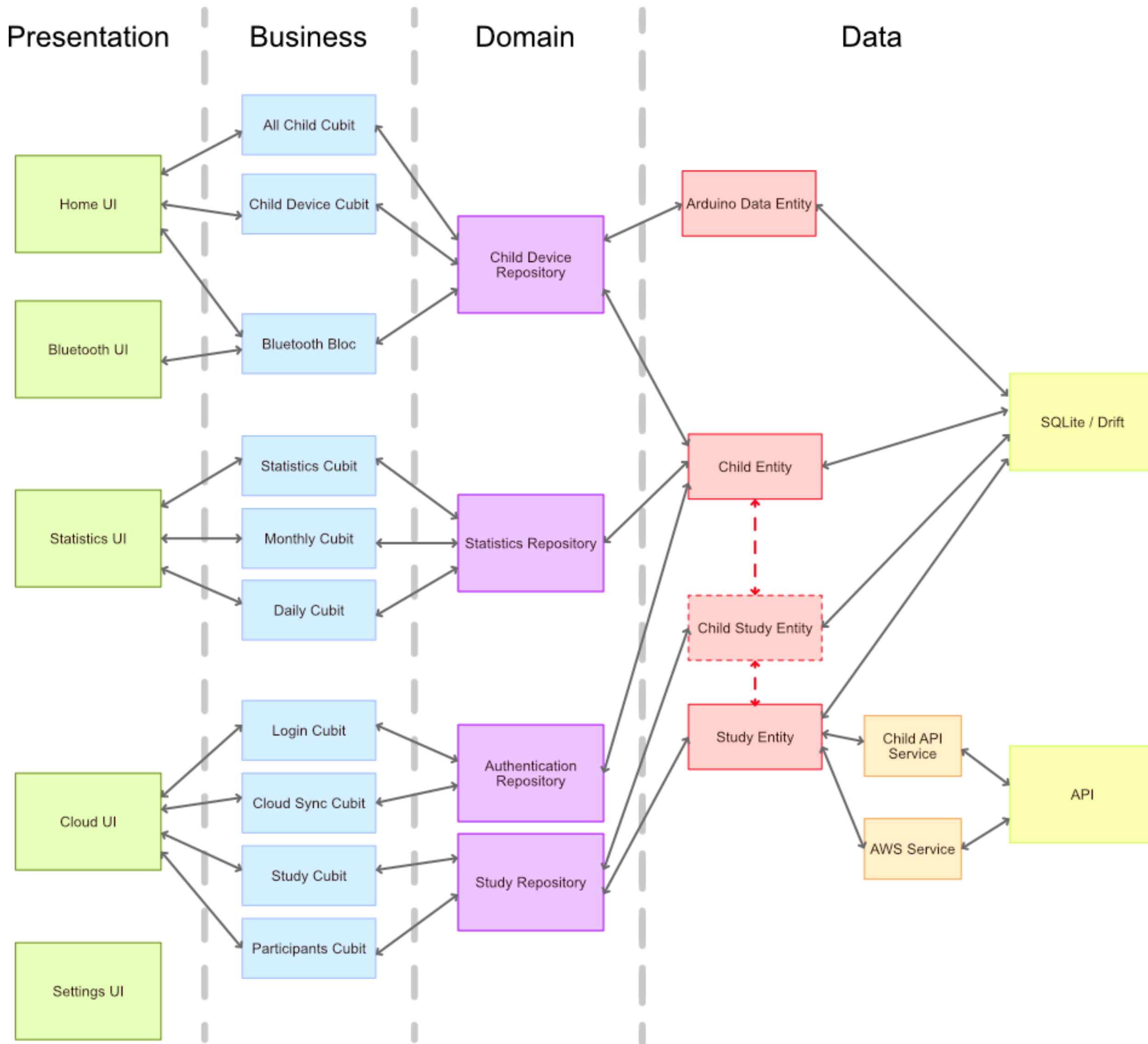
There is also a timestamp characteristic. This is written to by the app to communicate the last received timestamp, allowing the Neox Sens to strictly send samples that the app has not received yet. The Neox Sens performs a binary search for this timestamp in the EEPROM and only sends samples after the timestamp. This is done to minimise data transmission time and storage space usage. However, if the timestamp is not found, all samples are sent. Lastly, there is a progress characteristic, which contains the total number of samples to be sent for this sync. This will be utilised by the app to display a progress bar for the user.

App

App Architecture

Flutter was our chosen framework for the project. Flutter offered the most robust support for syncing with Bluetooth Low Energy. Flutter also supports cross-platform development, allowing us to build applications for IOS and Android from a single codebase. Additionally Flutter’s use of widgets allowed us to create a visually appealing and intuitive interface. In addition, Flutter is backed by Google, which ensures regular updates and long-term support, making it a reliable option for the future. The hot reload feature on Flutter greatly enhanced the app's development time as changes could be seen and tested in real-time.

The app is separated into four layers: Presentation, Business, Domain, and Data. The clean architecture software design pattern was used. The SOLID design principles were implemented, allowing a structured architecture which allowed for test-driven development from the outside-in. The architecture also allowed multiple members working on the app independently. Each component of the Neox App is decoupled, allowing them to be isolated for durability and maintainability.



Frontend

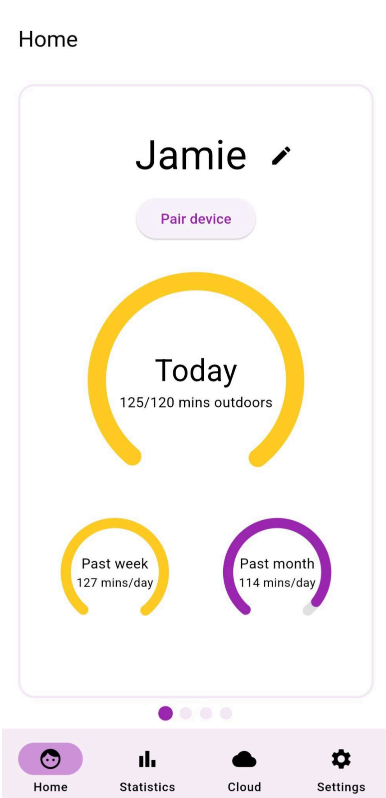
Colour Scheme

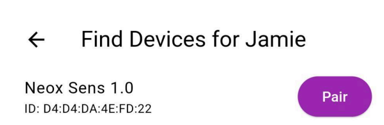
The colour palette characterised by the vibrant violet colour is an embodiment of our values we want to offer for families with myopia. Recent research in myopia reveals that violet light exposure reduces the progression of myopia in children. We have chosen the colour schematic based on dark violet to communicate our vision for Neox. Not only do we want to achieve positive influence on the families, but also facilitate long term engagement in myopia research. The colour palette is a reflection of our commitment to fostering the younger generation.

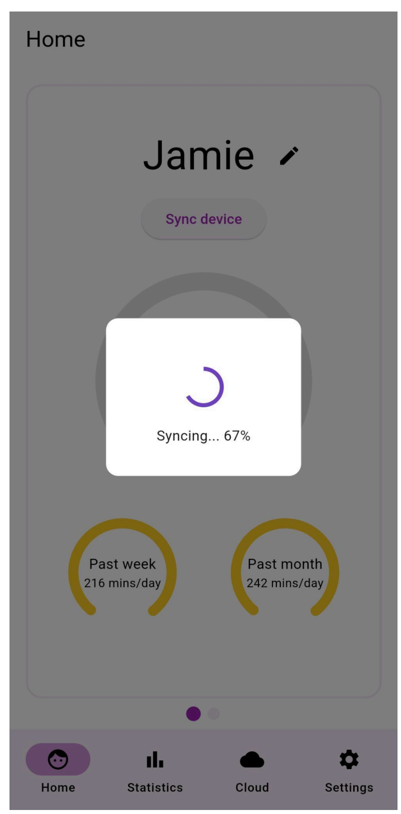
Syncing with Neox Sens

The sync functionalities were implemented in the home page for easy access. When the "Pair device" button is pressed, the parent is navigated to a scanning screen. Only the Neox Sens device is recognised by the app. The parent is prompted to enter an authentication code.

When the child has a paired device, a "Sync device" button is shown. A popup is displayed to notify the user about the syncing progress.

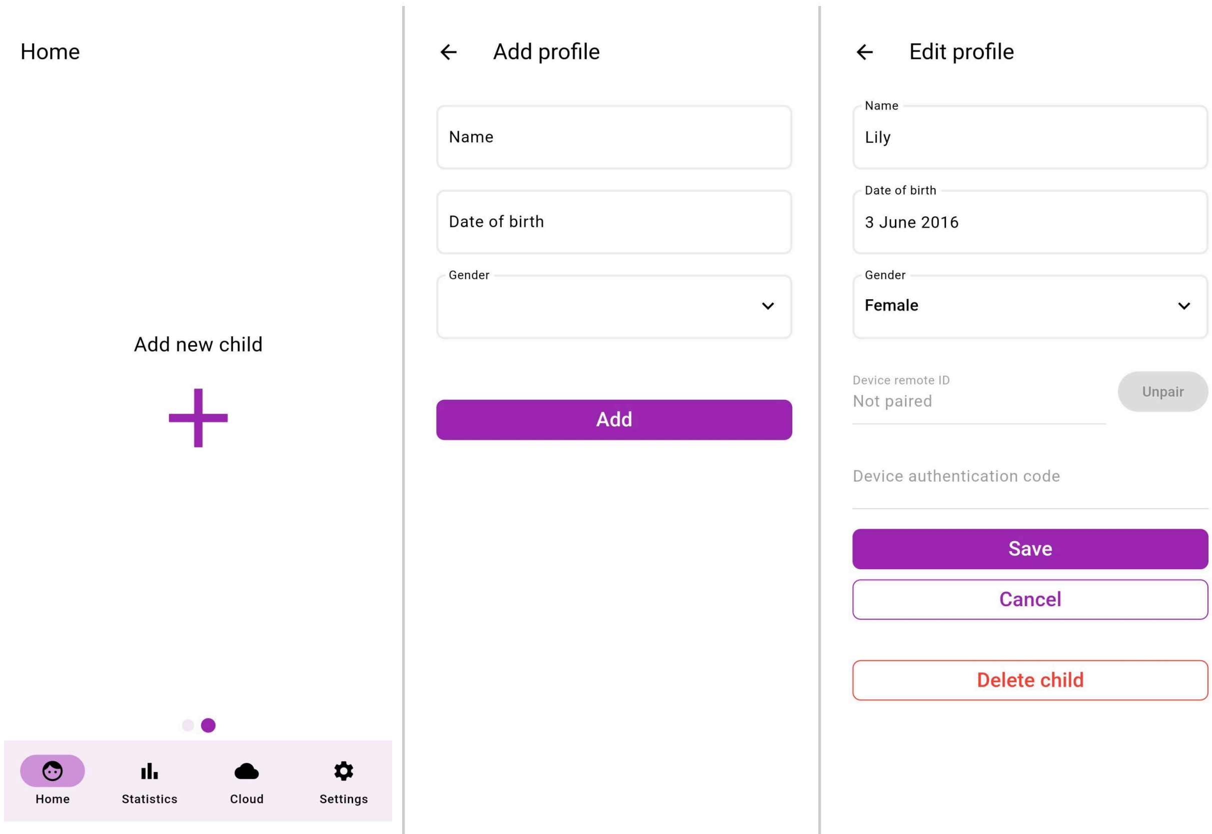






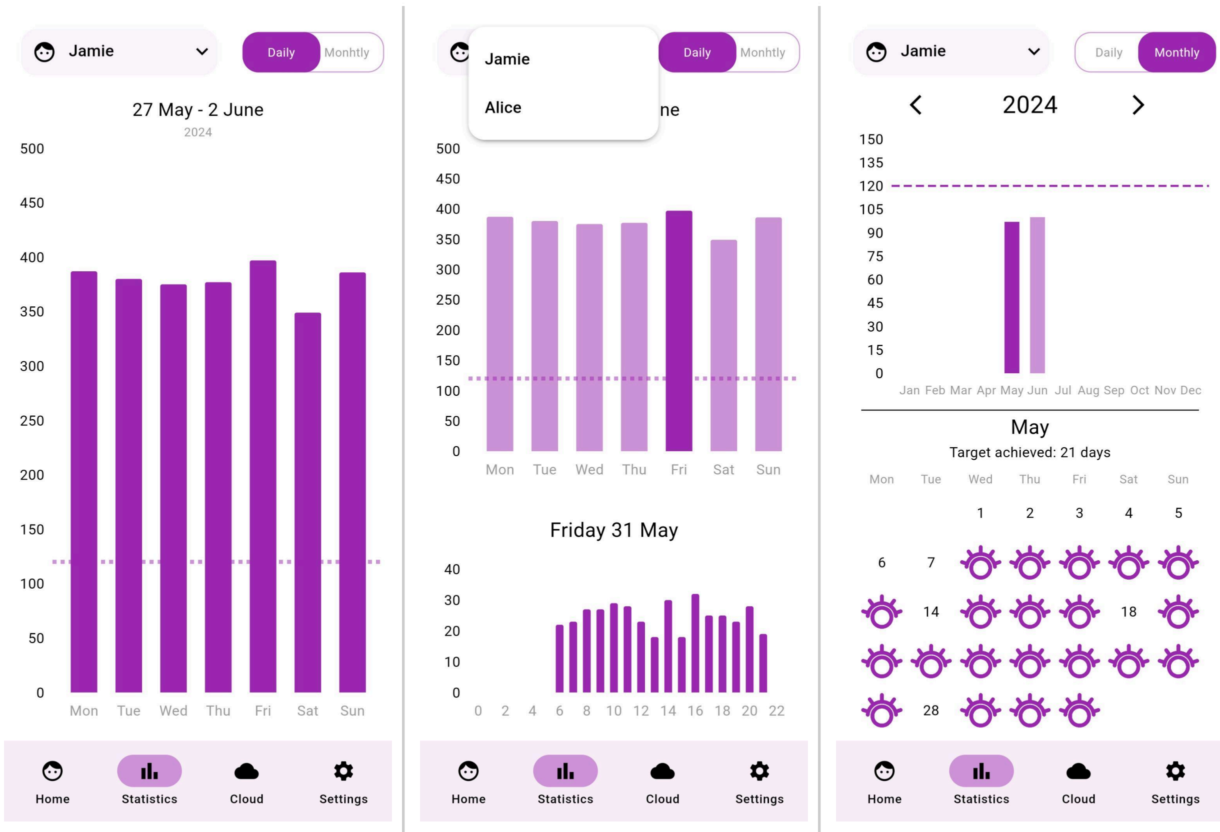
Managing Children

In the home screen, the parents can scroll through the children or add a new child. An edit button is implemented so that the parents can edit the child details, unpair the device, or change the device authentication code.



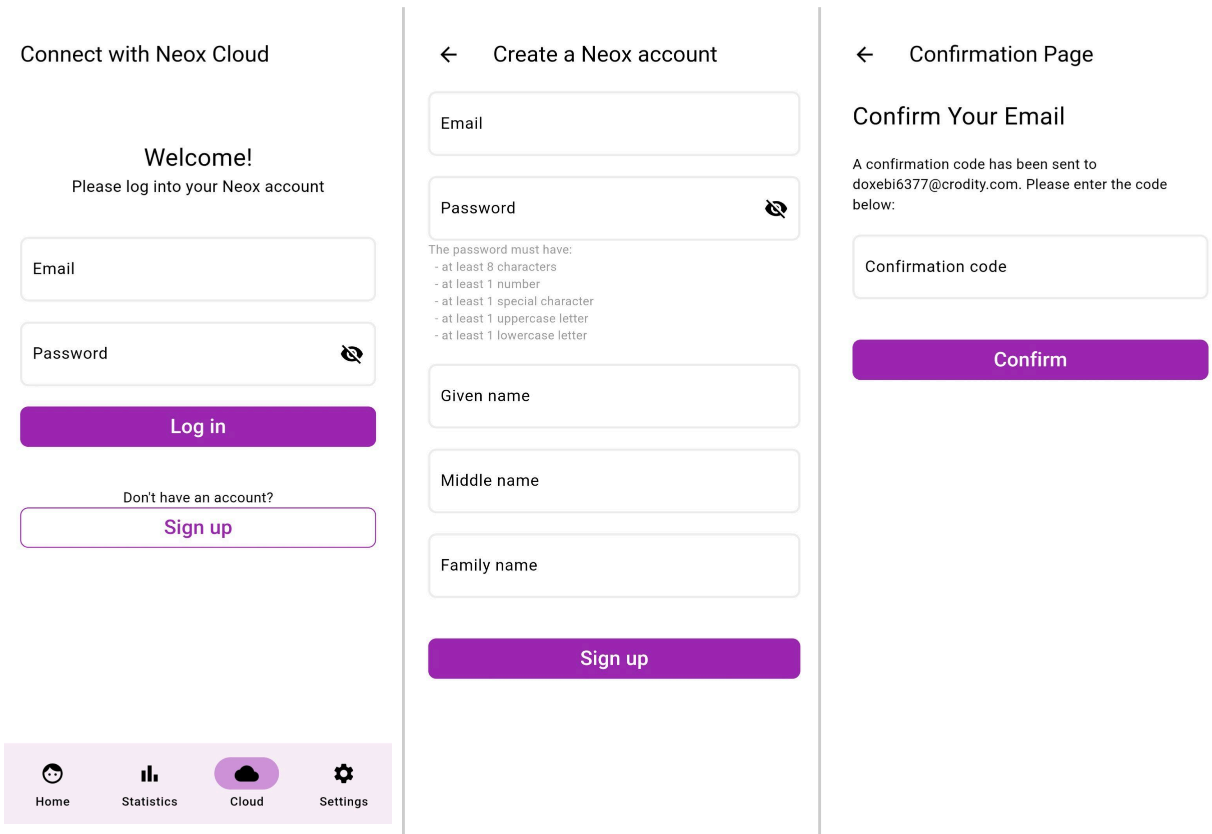
Statistical Analysis

An overview is provided in the home page. In the Statistics page, particular attention was given to the user interaction such as scrolling and tapping. The hourly graph is displayed depending on whether the user wants to show or hide the graph. Infinite scrolling has been implemented to show more data. The monthly bar chart and calendar are interconnected. The UI is rendered depending on the press of the bar chart or scroll of the calendar. On hold, a tooltip is displayed to show the exact minutes. The parent can change the child through a drop down menu. A toggle was implemented to change views.



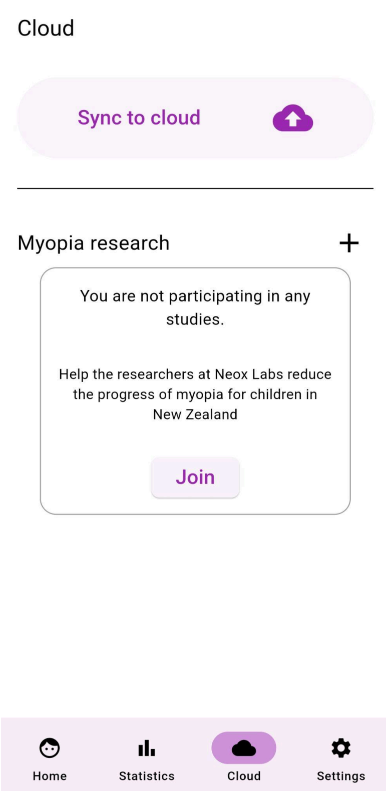
Authentication

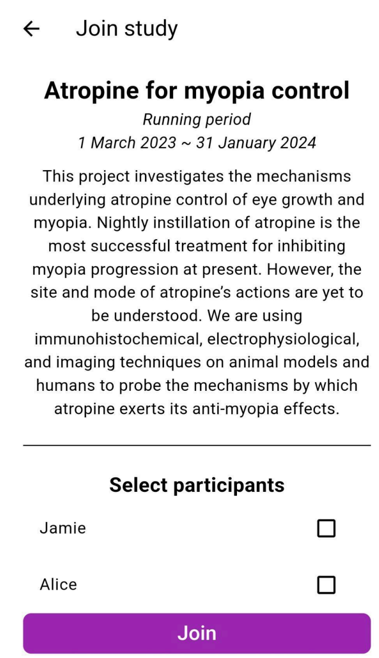
Attention was given to reduce friction in the sign up, confirmation, and login process. When the parent signs up for an account, a confirmation code is immediately sent to the user. After successful confirmation, the parent is automatically logged in. The log in state is maintained throughout the session until the app is closed. The AWS tokens are handled in the backend. Consistent styling was applied for the sign up, confirmation, and login pages.



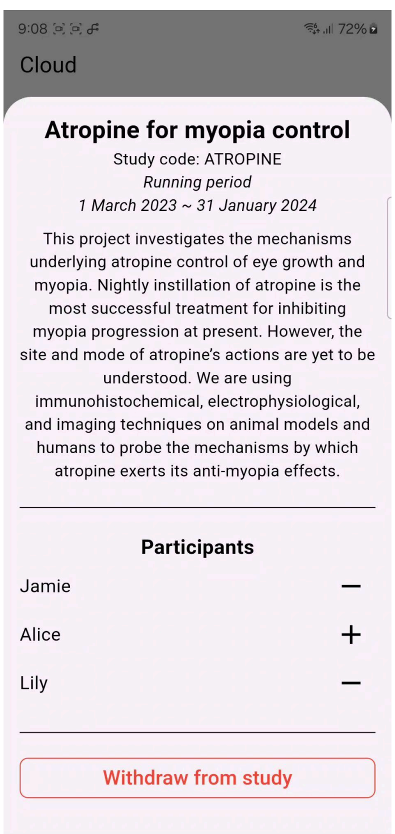
Cloud and Research

In the cloud page, the parent can sync the samples. If there are children synced to the cloud, the profiles will load to the local database. The samples downloaded from Neox Cloud are reclassified according to the local classification model. The parents can also participate in research. When the plus or button is pressed, a popup is shown for the parents to enter the study code. The parent can select which children they want to participate in. The parent can also manage studies by clicking on the study cards. A bottom sheet is shown displaying the study details, options to opt in or out of the study, or withdraw from research.


Cloud
Sync to cloud
Myopia research
You are not participating in any studies.
Help the researchers at Neox Labs reduce the progress of myopia for children in New Zealand
Join
Home
Statistics
Cloud
Settings

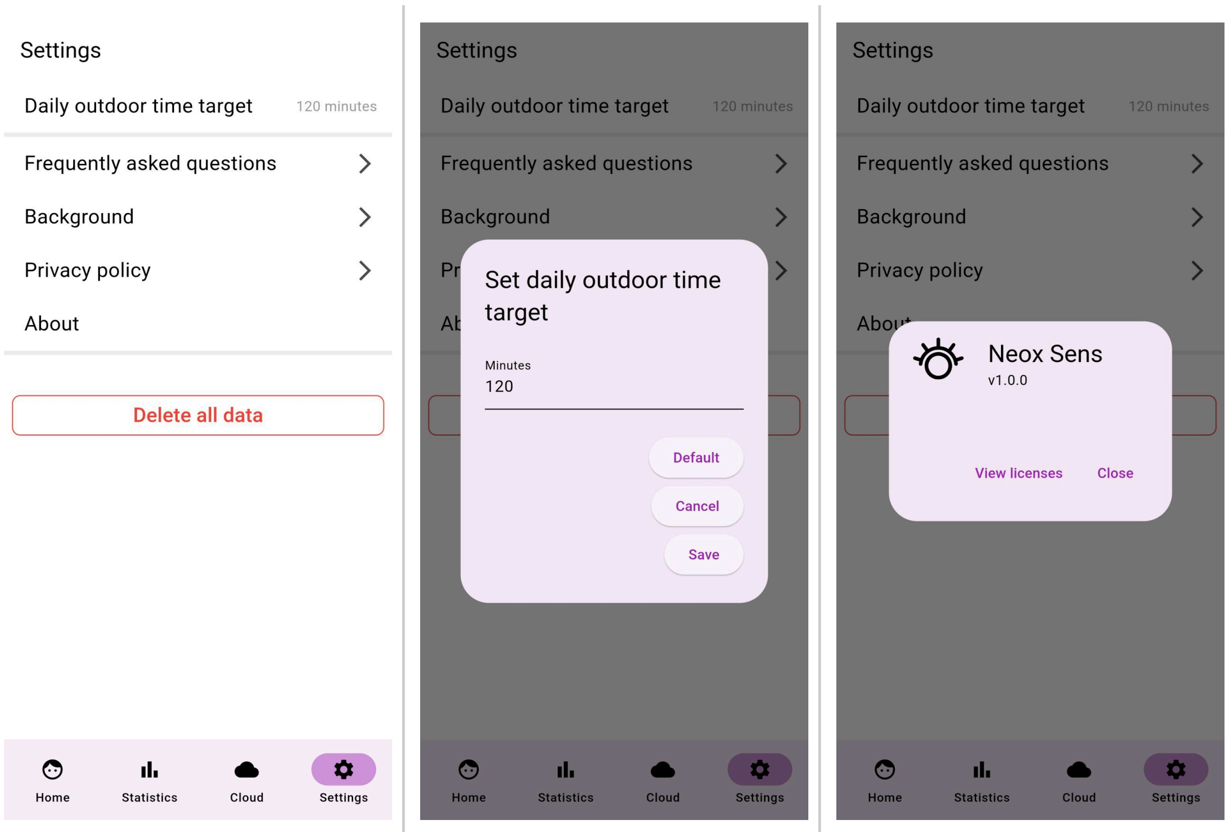

Join study
Atropine for myopia control
Running period
1 March 2023 ~ 31 January 2024
This project investigates the mechanisms underlying atropine control of eye growth and myopia. Nightly instillation of atropine is the most successful treatment for inhibiting myopia progression at present. However, the site and mode of atropine's actions are yet to be understood. We are using immunohistochemical, electrophysiological, and imaging techniques on animal models and humans to probe the mechanisms by which atropine exerts its anti-myopia effects.
Select participants
Jamie
Alice
Join


Cloud
Sync to cloud
Myopia research
Atropine for myopia control
Running period
1 March 2023 ~ 31 January 2024
This project investigates the mechanisms underlying atropine control of eye growth and myopia. Nightly instillation of atropine is the most successful treatment for
Choroid and myopia
Running period
1 January 2023 ~ 31 December 2026
Research suggests that the choroid plays a major role in guiding axial eye growth.
Home
Statistics
Cloud
Settings


9:08
72%
Cloud
Atropine for myopia control
Study code: ATROPINE
Running period
1 March 2023 ~ 31 January 2024
This project investigates the mechanisms underlying atropine control of eye growth and myopia. Nightly instillation of atropine is the most successful treatment for inhibiting myopia progression at present. However, the site and mode of atropine's actions are yet to be understood. We are using immunohistochemical, electrophysiological, and imaging techniques on animal models and humans to probe the mechanisms by which atropine exerts its anti-myopia effects.
Participants
Jamie
Alice
Lily
Withdraw from study

Settings

In the Settings page, the parent can set the daily outdoor time. The statistical insights are globally affected. The daily outdoor time is used as the threshold for the home screen, bar charts, and calendar stamps. Common pages are also implemented, such as Frequently Asked Questions and Privacy Policy. A button to delete all data has been implemented.

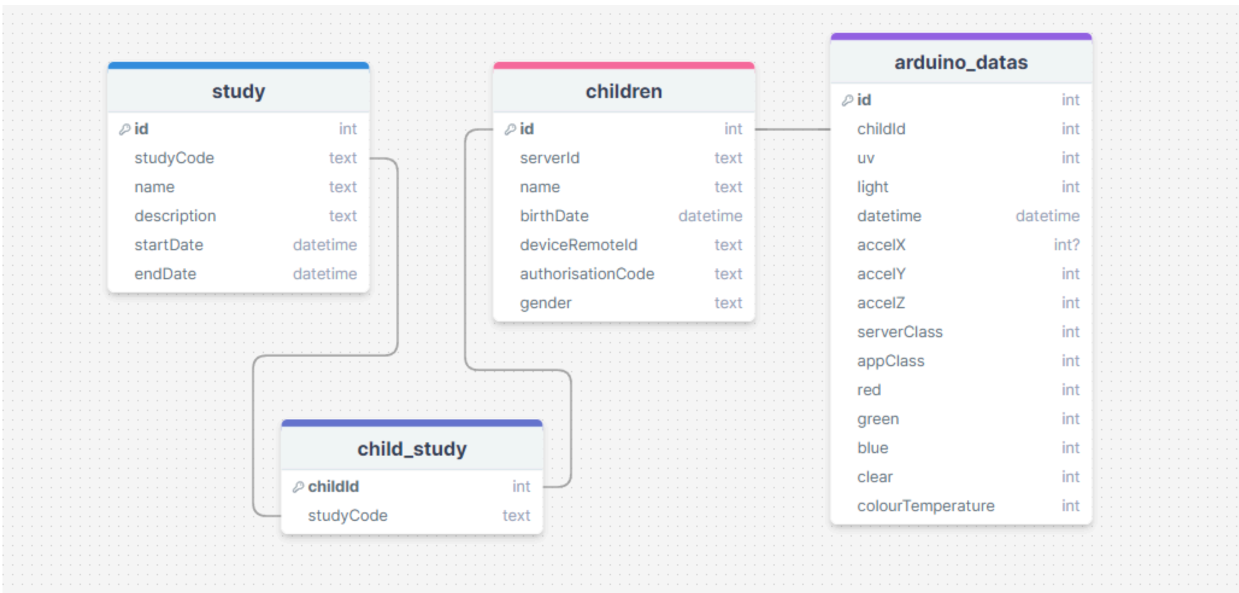


Backend

Database

SQLite is bundled into the Flutter app by default. While SQLite is very stable, it lacks features for optimisation, which affects the performance. Therefore, an object relational mapper (ORM) is used on top of the database to make data handling quicker and easier. Out of the many options, Drift (formerly Moor) was selected as it is the most widely supported by the Flutter community. It also serves our purposes, and offers cross platform support.

Drift has the risk of not being maintained as it is built by one developer. However, it has a 3 year record and is widely supported by the community. Even if Drift is not maintained, the SQLite database is still present underneath. The app can easily implement another ORM that supports SQLite.



We also have development only features for the backend

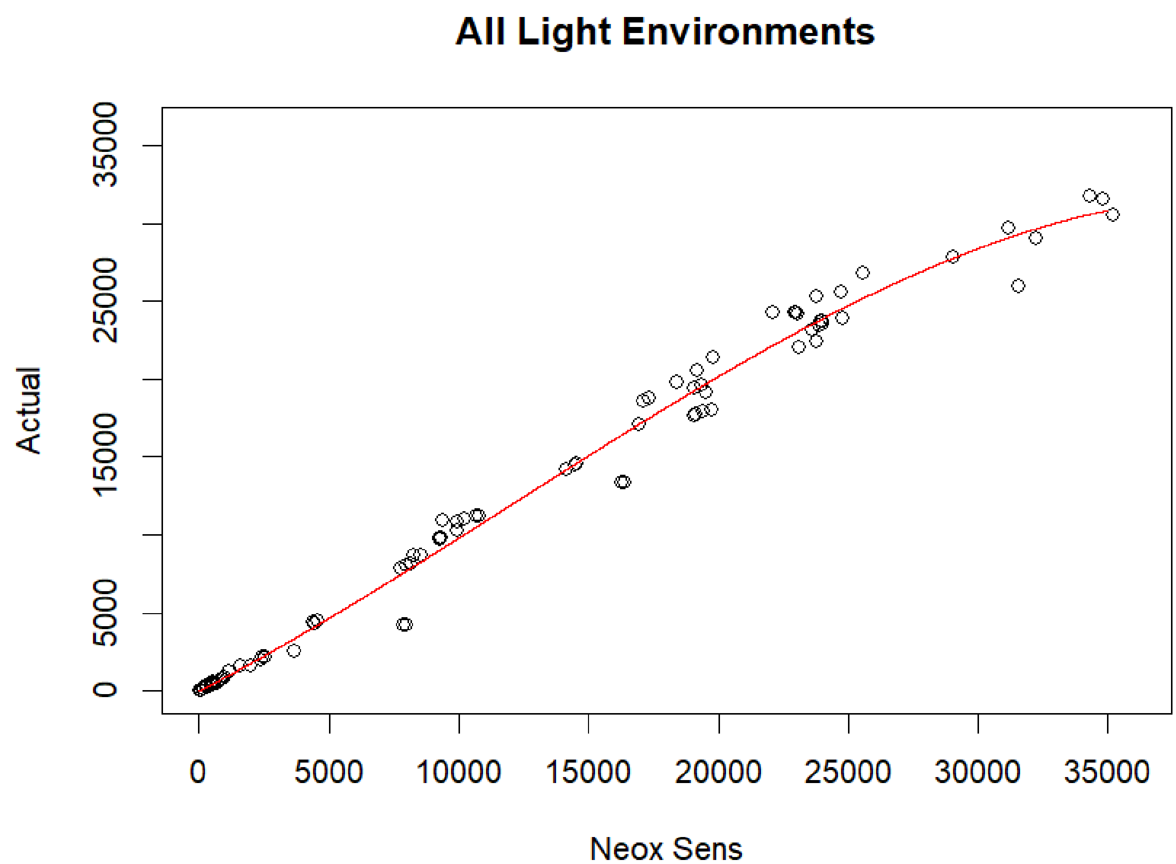
1. Data generation
2. Database viewer

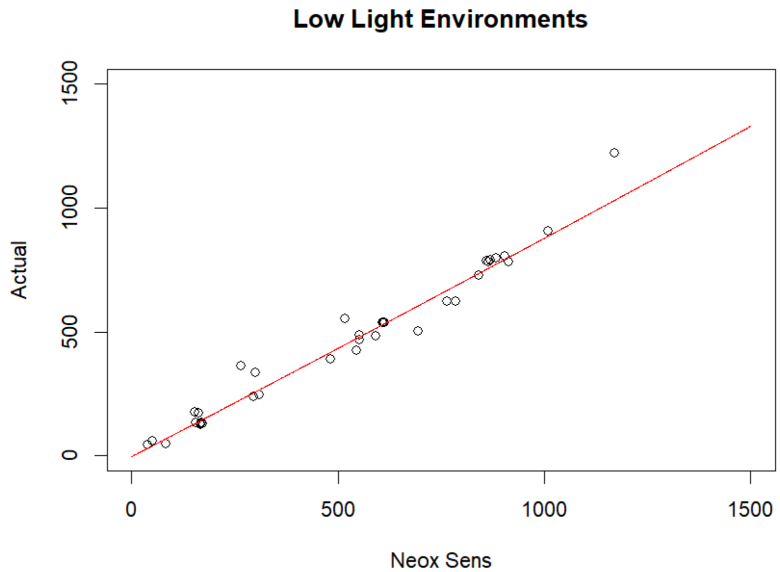
API Endpoints and Usage

Flutter comes with its own Dart HttpClient bundles in the Flutter SDK, providing a simple API for making HTTP requests. However, the built-in HTTP package falls short when handling complex network scenarios. In our project, we decided to utilise the Dio package for our API integration. Dio is a powerful HTTP client for Dart, which provides various features that simplify the process of making HTTP requests and handling responses. Dio was chosen for ease of use, asynchronous operations, interceptors and error handling. This choice ensures that the app remains maintainable, scalable, and fluidly handles complex network scenarios.

Lux Calibration

The digital lux reading was calibrated to a luxmeter. The Chauvin Arnoux CA1110 Light Meter was used. Although the luxmeter reads between 0.1 to 200,000 lux, the digital lux reading from Neox Sens is only able to read up to 35,000 lux due to the size of the bits that store the reading. However, higher lux levels of over 35,000 lux are not important for indoor and outdoor classification as the level of lux cannot be achieved indoors. Therefore, the digital lux sensor was only calibrated in this range. 100 samples from various environments were used to calibrate the sensor.





The lux calibration curve between the Neox Sens and the CA1110 luxmeter follows a cubic relationship. There is a strong constant scatter between the Neox Sens lux reading and the actual lux value. The calibration is reflective of the actual lux reading for lower lux values which are used for indoor classification.

The model explains 98.9% of the variability in the lux readings.

The final calibration model is:

$$Actual_i = -2.290 + 0.8646 \times NeoxLux_i^{2} - 1.627 \times 10^{-5} \times NeoxLux_i^{2}$$

$$- 4.515 \times 10^{-10} \times NeoxLux_i^{2} + \epsilon_I \text{ , where } \epsilon_i \sim iid\, N(0, \sigma^2)$$

Server

To implement our serverless architecture, we used AWS Serverless Application Model (SAM) which provides a file format in which to declaratively specify our cloud infrastructure as well as a command-line program to build, validate, and deploy such infrastructures.

API

From the start of the project, we made a central document for the API which all teams could program against rather than having to constantly ask how to use it. It was therefore easier to work asynchronously.

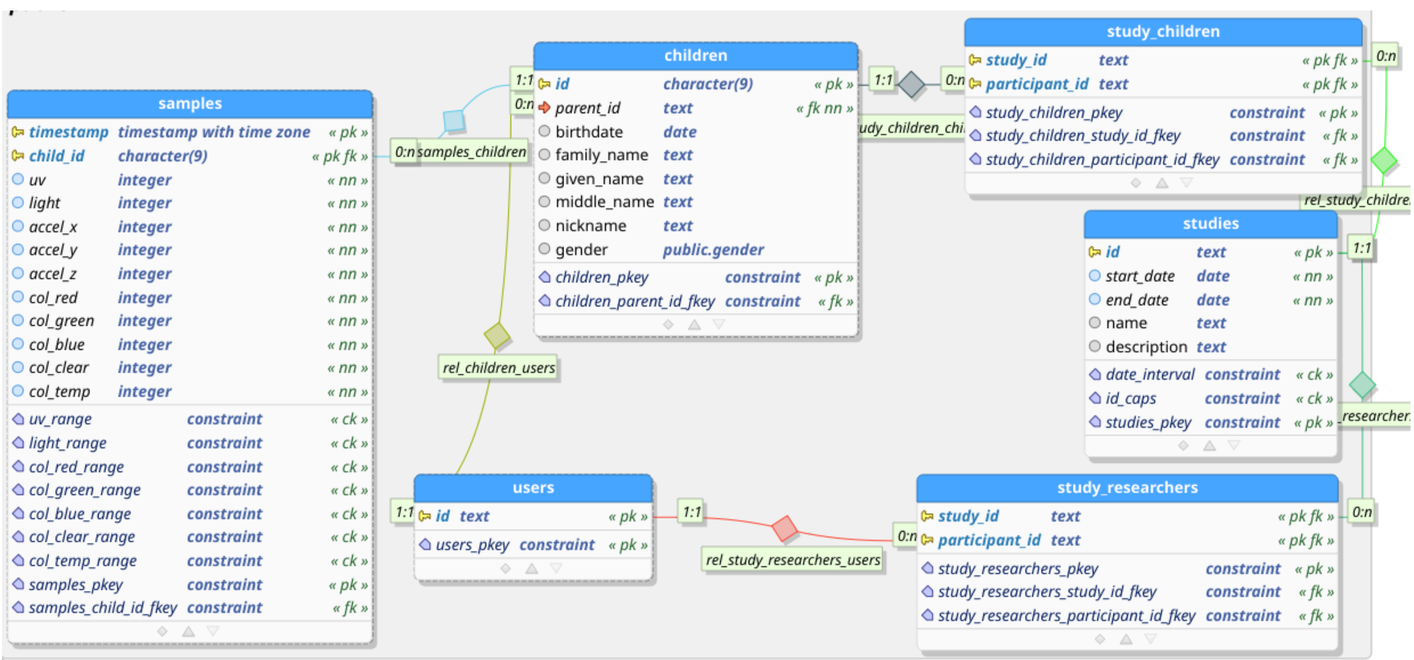
Since the API is required to have studies in addition to sending and receiving sample data, many more API actions were needed, which can be broadly grouped as:

- registering *X* with the server,
- viewing and editing the details about *X*,
- viewing the associations of *X*, and
- sending and retrieving samples from *X*

where *X* can be a child, parent, researcher, or study. The API document goes into more detail about each API action.

Since we use API gateway and Lambda, different lambda functions are executed in response to API calls. Each such handler for API calls can thus be small and simple but since there's no monolithic OS process linking the pieces together, it was more difficult to ensure data was consistent. To help with this, we made the database model as rich as possible in order to enforce the constraints and relationships between entities in our API.

Database



In order to implement this, we opted to use PostgreSQL because it is a mature and open source SQL database with support for rich data models. In addition, it is easy to add extensions to PostgreSQL to add support for auditing, optimisations for time series data, and other features. We use Amazon Relational Database Service (RDS) since it manages administrative tasks such as security upgrades, OS upgrades, backups, and mirroring for high availability in case one DB instance or data centre goes down. To manage all those tasks ourselves would defeat the purpose of a serverless architecture because it distracts us from our main purpose which is to provide an API for researchers to send and receive samples from the hardware device.

Implementation of API Actions

In addition, there was a concern that the implementations of similar API actions would go out of sync if they were written in completely separate files, so we instead implemented multiple related API actions in one file and then that source file branched depending on what API action was being run. For example, the API actions to create studies as well as fetching and modifying their details was implemented in one file since they all dealt with study details. A monolithic application would not have had to do this since it would be easier to maintain abstractions across the whole application.

As for the language used to write the handlers for API actions, we used Javascript on the Node runtime because Javascript is widely used in industry and has many libraries available.

URI Scheme

To get a uniform interface in the URIs, we defined a simple scheme for representing all the entities available:

- */samples/{childID}*: for samples from a child identified by *childID*
- */children/{childID}/**
- */parents/{userID}/**
- */researchers/{userID}/**
- */admins/{userID}/**
- */studies/{studyID}/**

We put samples under the special */samples* resource because sending and fetching samples is the primary purpose of the API and it should therefore be easy for consumers of the API to access them. The more verbose alternative would be */children/{childID}/samples*.

Security

We configured API gateway to serve the API actions but left clients to authenticate with the API using a token retrieved directly from the authentication service from AWS (Cognito). We use only ID tokens from Cognito which provides only the identity and group membership of the caller. Before the API actions are executed, we have configured API Gateway to validate the ID tokens received from the caller and reject the request with a status code 401 if it fails. We use only ID tokens and not access tokens which provides more features because we have a custom permissions scheme which depends only on the identity of the caller and (for researchers) also the studies that the caller is part of. For example, researcher accounts can fetch the samples of a specific child (GET */samples/{childID}*) but only if the caller is part of at least one study that the child is also part of.

We could do this with access tokens and the OAUTH scopes it provides but there is a risk of the scopes in a token becoming out of sync with the studies tables in the database and thereby introducing a security flaw since a researcher would be authorised for a study even after being removed from it. This contrasts with the user groups feature provided in both ID and access tokens which we use to store whether the user has the role of parent, researcher, or admin. This does not have a potential security flaw from stale tokens because the group membership of a user never changes after creation.

We also hardened the security of our cloud backend by layering multiple security measures such as:

- Placing the database inside a private subnet within our virtual private cloud (VPC)
- Enforcing security groups so that only authorised applications (such as the lambda functions) can access the database and the AWS service to securely store credentials for our database (Secrets Manager)
- Enforcing network access-control lists
- Setting up a NAT gateway so that the lambda functions can access the Cognito API but without exposing each lambda to unsolicited connections from the public internet. We do this because the Cognito API cannot be accessed from within a private subnet

This makes our solution closer to being ready to deploy in production for which our declarative infrastructure specification has some initial support.

Website

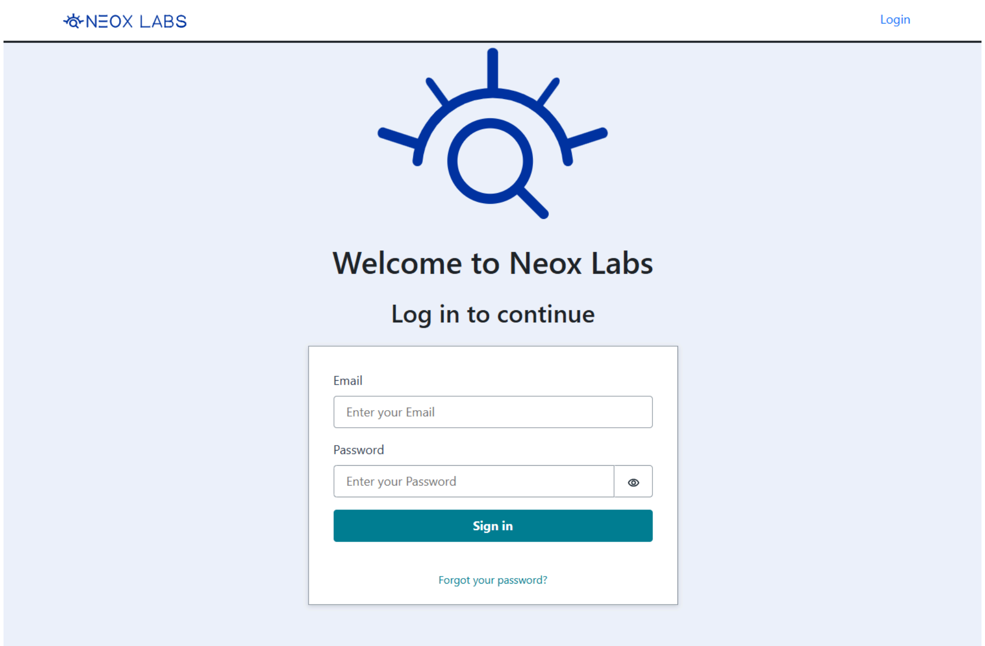
The website utilises Amazon Web Services' Cognito to ensure security and stability for the authentication system. It relies on API actions to authenticate users and manage data securely. For authentication, the website verifies the user's token through AWS Cognito, to ensure that only authorised users can access the authorised features. Additionally, API actions are used to manipulate details of studies and enable downloading raw data. The website requests data from the server using the user's token, then the server verifies it to check if the user is authorised for access. Subsequently, the server returns the raw data and then the built-in function of the website converts the raw data into a CSV file to send it to the user.

While designing the website UI, we followed various design principles and Nielsen Heuristics while taking into account accessibility guidelines and Gestalt principles to ensure a good user experience. For the overall look of the website, we follow the minimalist design Nielsen Heuristic. We ensured there was no irrelevant information on the screen, and in doing this, we were also able to display the emphasis design principle. The user is always able to recognise what part of the interface they should focus on. We intentionally set up the study cards to follow the common fate Gestalt principle. As the cards all follow the same path, the user will perceive them as related, making our interface easy to understand at first glance. We ensure the size of the Neox Labs logo on the left side of the navbar has the same weight as the links on the right side, which is in line with the design principle of balance. This improves the overall look and feel of our website.

An important aspect of our design is accessibility, particularly in regards to colour. We ensure all coloured text has a contrast ratio that is above 4.5:1. This is important to ensure all text on the interface is clear and readable, even to those with eye conditions. We also ensured to choose colours that users who may be colour-blind would be able to see, regardless of which type they may have. We mainly use hues of blue and solid black against a white background. This further enhances the accessibility of our website.

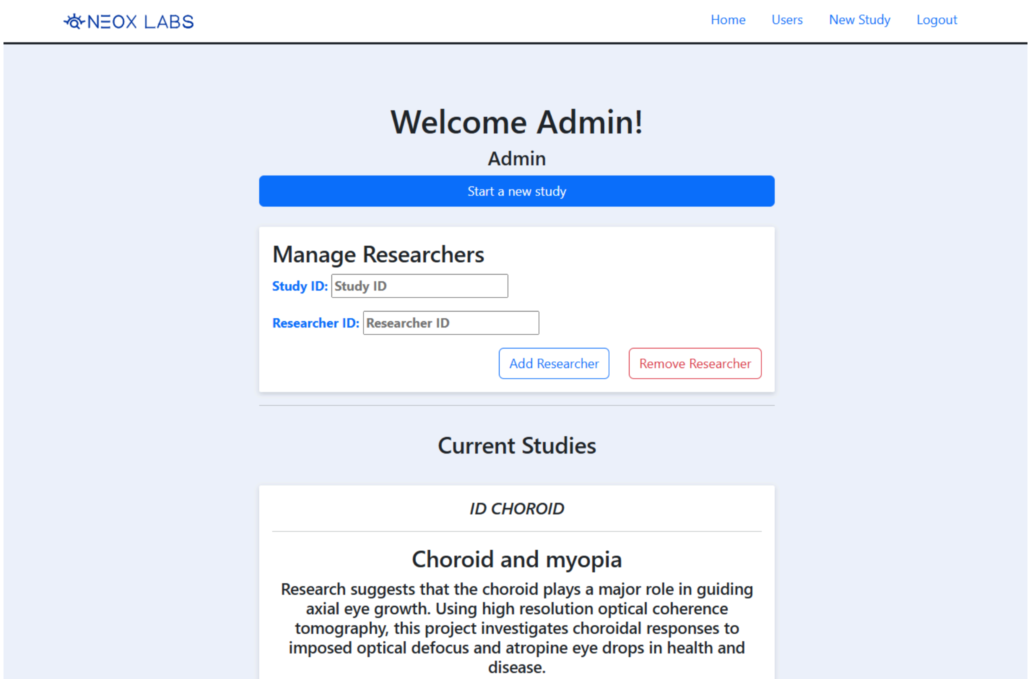
Login

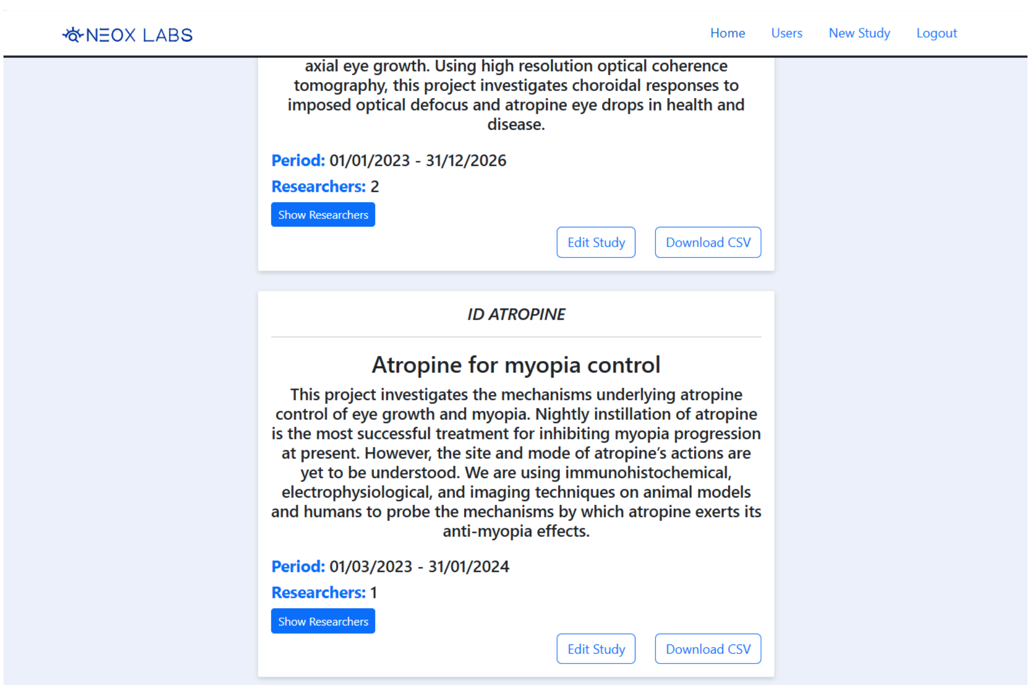
Role-based authentication depending on the account. Allows password reset for first time users.



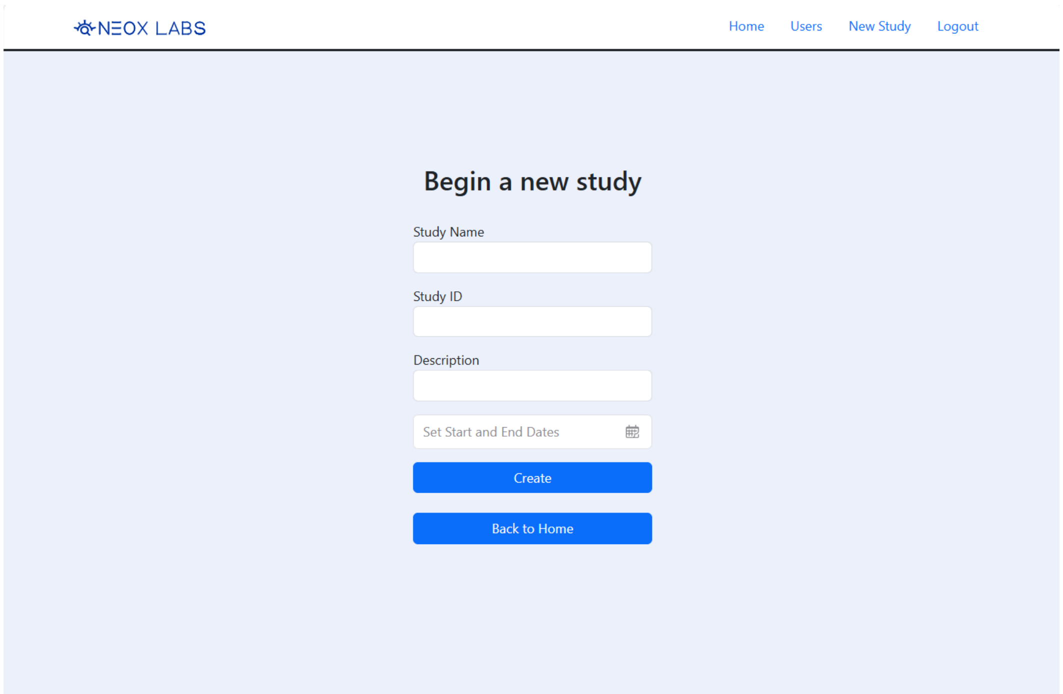
Admin View

Button to start a new study, and a panel to manage researchers. The current studies display the study details, as well as the admin-only option to view the number of researchers and edit the study. The admin can also download the samples as a CSV.

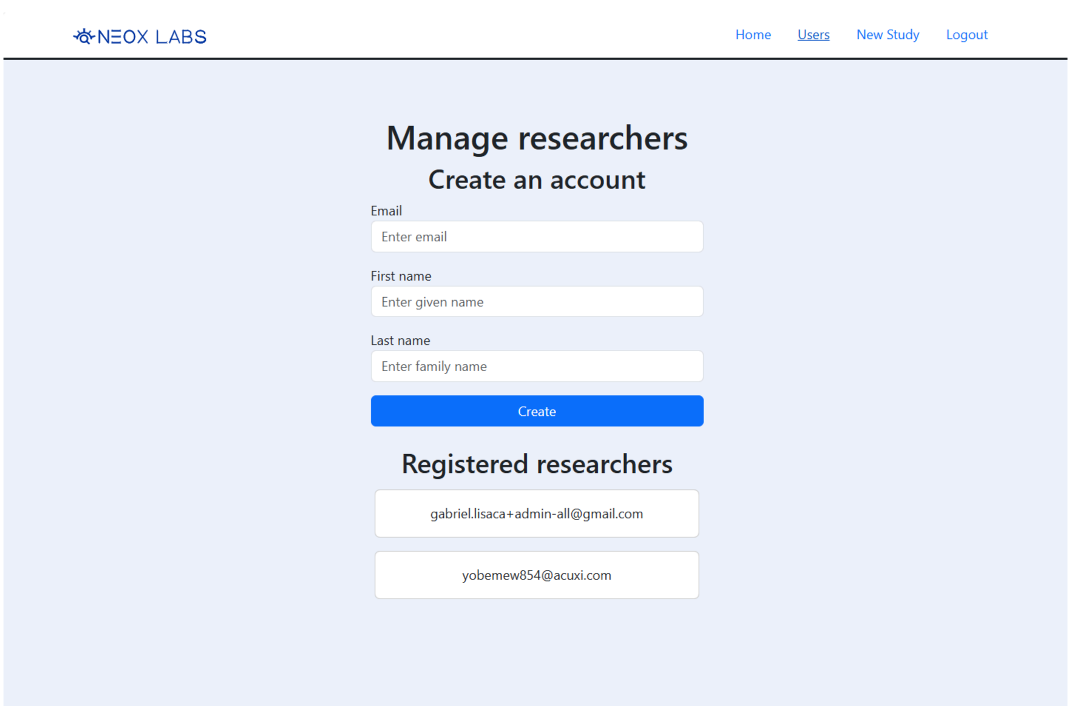




Create a study by providing the name, study id, description, and start time and end time. The study details, except for the study id, are editable.

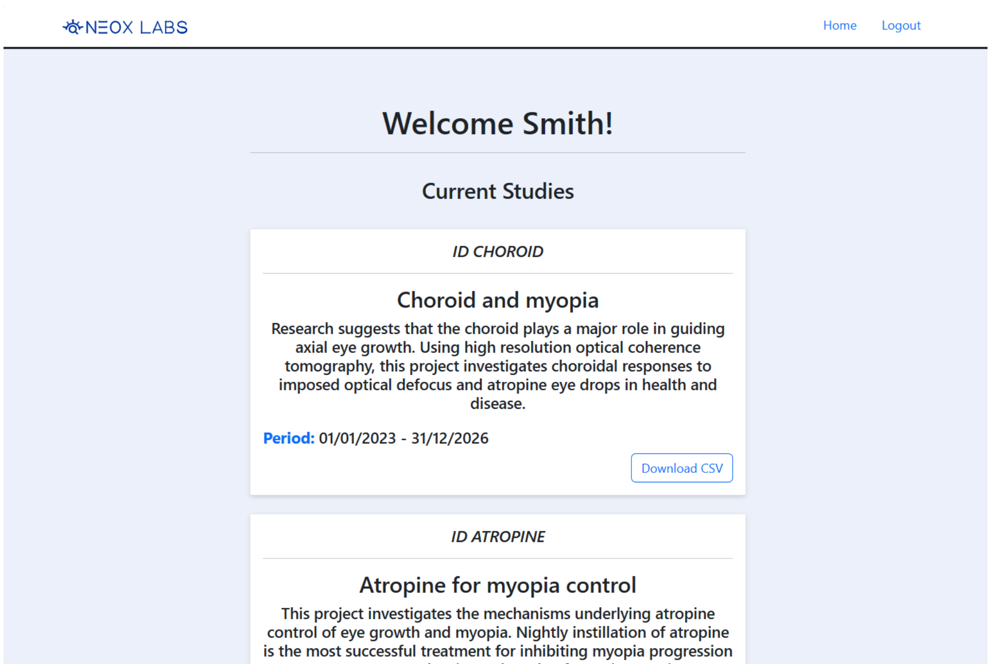


Create accounts for researchers by providing an email address and name. The admin can also view registered researchers.



Researcher View

View study details and download samples as a CSV file



Classification

Features

For the classification, eight features are collected from the Arduino hardware. These are uv, the x, y, z axis of acceleration, red, green, blue, and clear components of colour. Two features, colour temperature and lux, are calculated from the Arduino default library using the red, green, blue, clear chromatic components of light. Features were engineered for the classification. In total, 50 features were used for the classification.

19 features are related to light intensity and uv. 10 features were engineered from the calculated digital lux and analogue uv reading, in relation to the blue colour component of light.

Feature	Formulae	Total created
Light intensity	Lux	1
Analogue uv reading	UV	1

Colour temperature	$ColourTemp$	1
The difference in clear and blue colour light, in terms of uv	$\frac{C - B}{UV + 1}$	1
Ratio between light intensity or blue colour light and UV	$\frac{\{Lux, B\}}{UV + 1}$	2
Ratio between logged light intensity or blue colour light and logged UV	$\frac{\log(\{Lux,\ B\})}{\log(UV) + 1}$	2
Ration between square root of light intensity or blue colour light and logged UV	$\frac{\sqrt{(Lux, B)}}{UV + 1}$	2

Five features are for acceleration. Two features were engineered from the accelerometer data.

Feature	**Formulae**	**No. of Features**
Acceleration on each axis	$X,\ Y,\ Z$	3
Magnitude	$X^2\ +\ Y^2\ +\ Z^2$	1
Multiplication of lateral movement	$\|(X\ \times Y)\|$	1

The relationship between red, green, blue, and clear colour light sensor channels was investigated. From the four colour channels, 21 features

Feature	Formulae	No. of Features
Red, green, blue, clear colour component readings	$R,\ G,\ B,\ C$	4
The ratio between red, green, blue and clear	$\frac{\{R, G, B\}}{C + 1}$	3
The ratio between each colour	$\frac{R}{G + 1}, \frac{B}{R + 1}, \frac{B}{G + 1}$	3
The difference between the clear light and each colour, in terms of each chromatic component	$\frac{C - \{R, G, B\}}{\{R, G, B, C\} + 1}$	12

The ratio between logged red, green, blue and logged clear	$\frac{\log(\{R, G, B\})}{\log(C)+1}$	3
Ratio between logged colour	$\frac{\log(R)}{\log(G)+1}$, $\frac{\log(B)}{\log(R)+1}$, $\frac{\log(B)}{\log(G)+1}$	3
Ratio between square root of red, green, blue and square root of clear	$\frac{\{\sqrt{R}, \sqrt{G}, \sqrt{B}\}}{\sqrt{C+1}}$	3
Ratio between square root of each colour	$\frac{\sqrt{R}}{\sqrt{G}+1}$, $\frac{\sqrt{B}}{\sqrt{R}+1}$, $\frac{\sqrt{B}}{\sqrt{G}+1}$	3

Classifiers

XGBoost, Adaboost, Random Forest, Support Vector Machines and Light GBM were used to classify the samples. The models were trained to achieve the highest Area Under the Curve score (AUC). The AUC reflects the ability to be selective on classifying the unknown samples as outdoors, in comparison to the false positive rate. To prevent overfitting, 5-fold cross validation was used. Grid search was used to find the best hyperparameter. 243 Xgboost models, 36 Adaboost models, 24 Random Forest models, 432 Extra Tree models, 18 Support Vector Machine models, and 480 Light GBM models were trained.

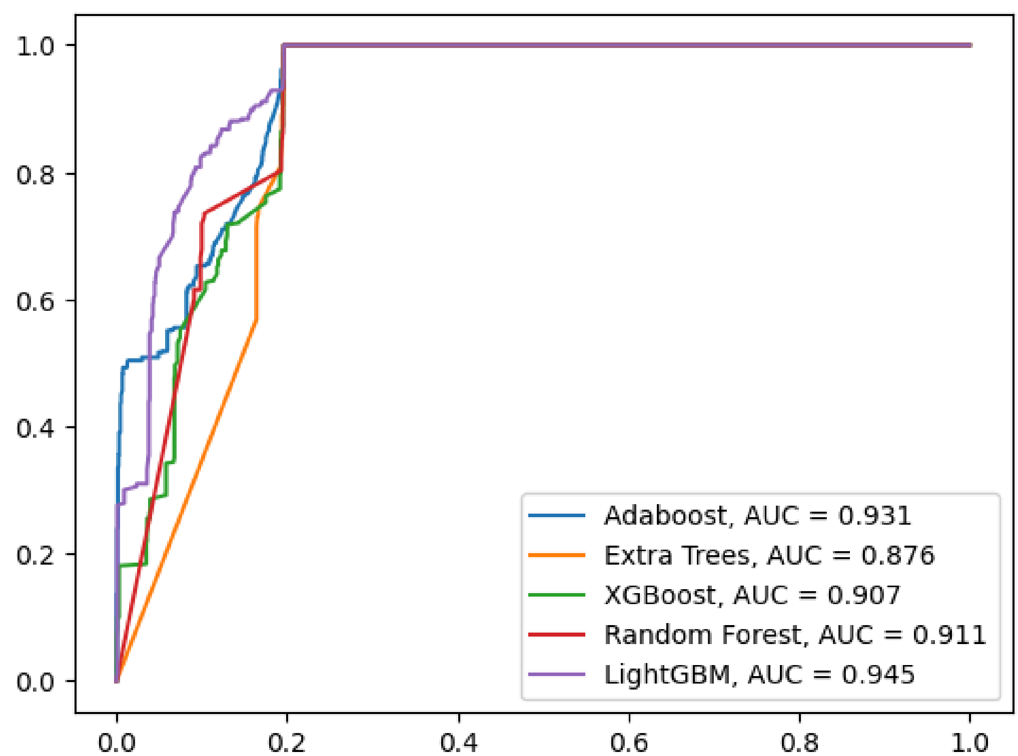
The best hyperparameter combinations for each classifier are outlined below.

Model	Best hyperparameters	AUC
XGBoost	'alpha': 0, 'gamma': 0, 'learning_rate': 0.1, 'max_depth': 5, 'subsample': 1	0.9429
Adaboost	'estimator__max_depth': 4, 'estimator__min_samples_leaf': 10, 'n_estimators': 50	0.9936
Random Forest	'bootstrap': True, 'max_depth': 80, 'max_features': 'sqrt', 'min_samples_leaf': 3, 'min_samples_split': 10, 'n_estimators': 100	0.9885
Extra Trees	'criterion': 'gini', 'max_depth': 16, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 6, 'n_estimators': 50	0.9300

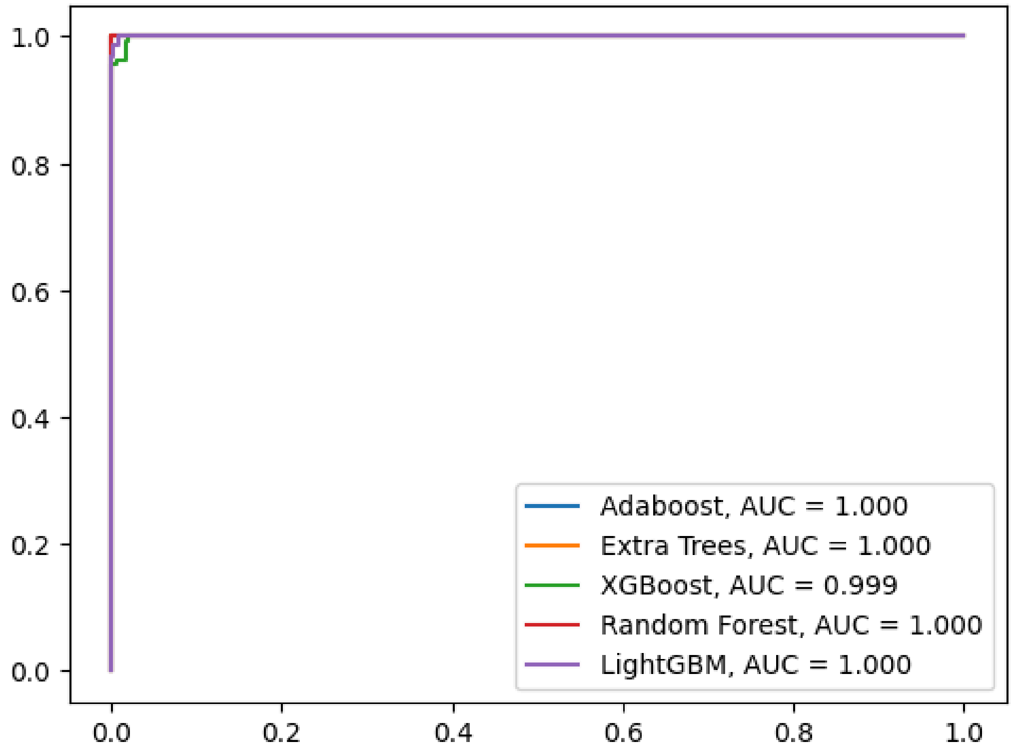
Support Vector Machines	Selected features: "(clear-red)/blue", "log(blue)/log(clear)", "log(blue)/log(red)", "sqrt(blue)/sqrt(red)" ''C': 10, 'gamma': 10, 'kernel': 'rbf'	0.9546
Light GBM	'boosting_type': 'gbdt', 'min_child_samples': 20, 'num_leaves': 80, 'reg_alpha': 0, 'reg_lambda': 0	0.9796

As we are not able to normalise the data or remove collinearity, tree-based methods performed the best. The best AUC score was achieved with Adaboost, Random Forest, and Light GBM. The worst AUC was from Support Vector Machines. This is understandable as no scaling was done to the data, and collinearity was not removed completely.

A second validation set was used to estimate the performance on unseen environments. The samples collected are independent from the training set. To introduce difficulty, environments that are difficult to classify were particularly selected. These include bright light indoor areas, such as 50 cm from a window with full sunshine, and a retail store which has strong light sources. Low-light outdoor environments include sunset time and in between large corporate buildings, where the lux is lower than 100.



All models performed well in defined indoor and outdoor environments.

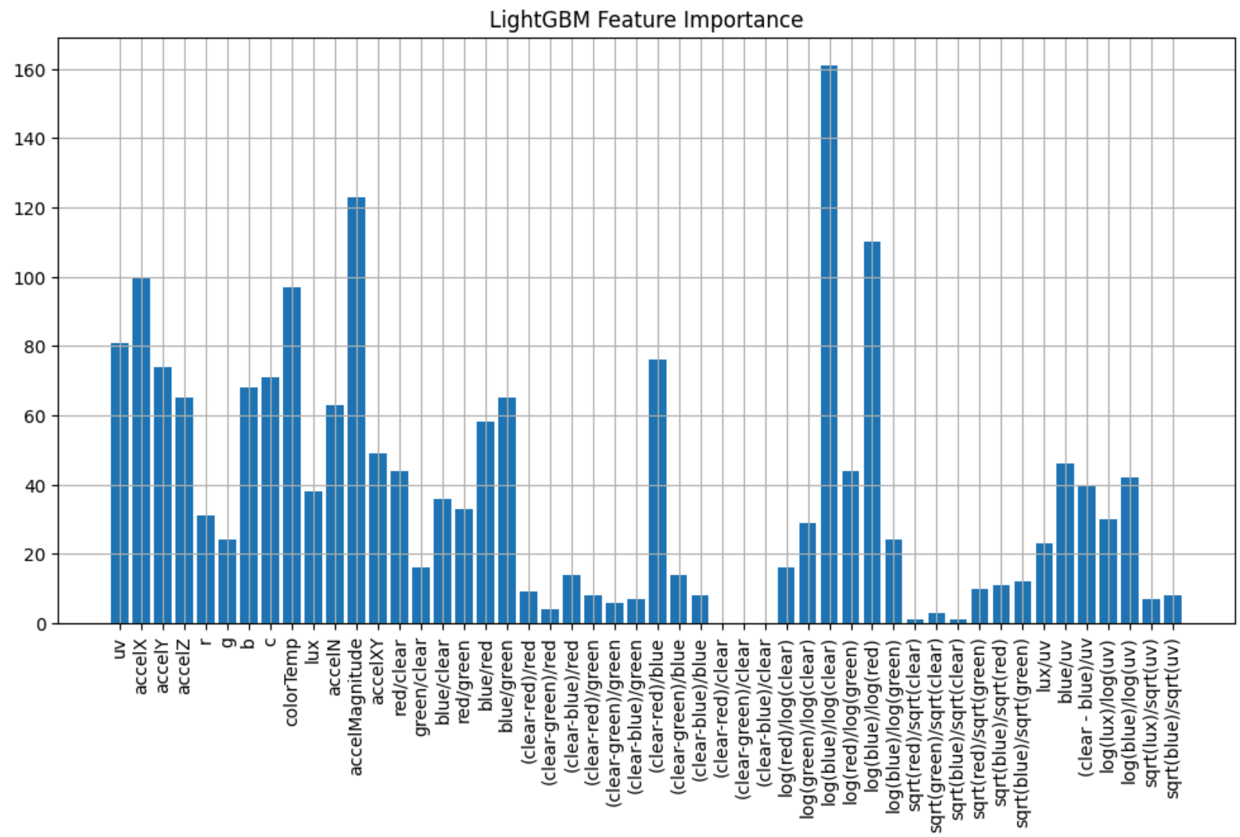


The highest AUC was obtained by the Light GBM model. The Light GBM model was most robust to difficult situations. Therefore, it was selected for the final model implemented in the Neox App.

When the best hyperparameters were used to train the model, the final model resulted in over 30,000 lines of code in dart. The app could not compile on build time. A smaller version of the Light GBM model was trained by limiting the maximum depth of the trees. The best hyperparameter combination is:

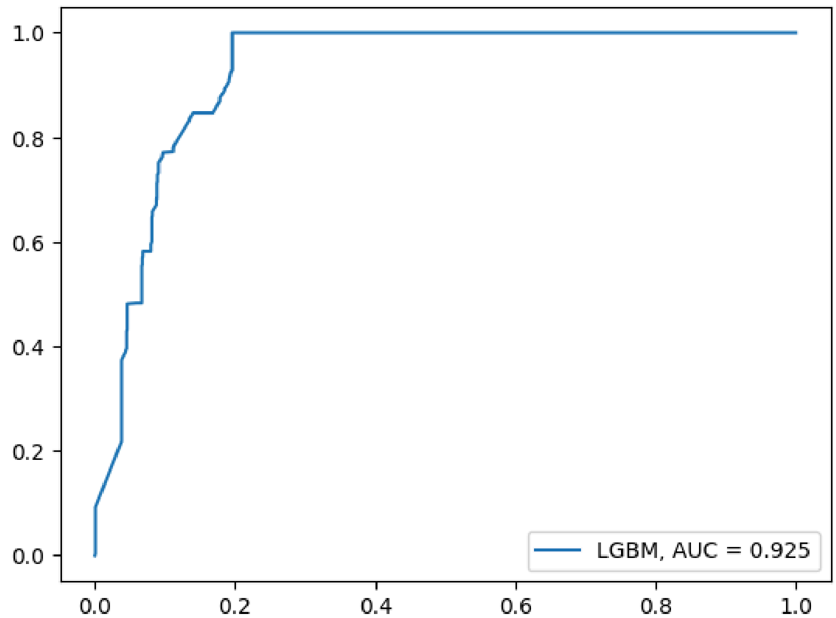
LightGBM	'boosting_type': 'gbdt', 'max_depth': 50, 'min_child_samples': 20, 'num_leaves': 20	0.9812

The two features that were most important for the classification are the ratio between the logged blue and logged clear components of light, and the ratio between the logged blue and logged red components of light.

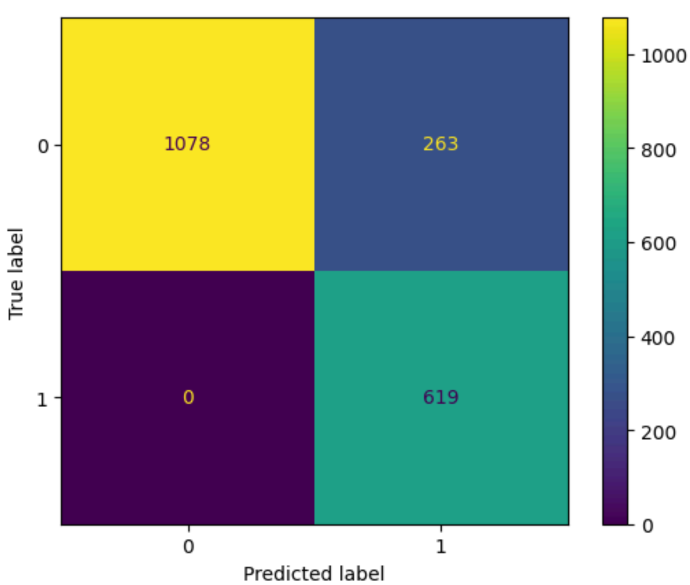


Contrary to our initial expectations, the chromatic composition of light was the most effective in determining whether the device was located indoors or outdoors. The lux and uv had lower reliance. This is because outdoor light levels can be as low as 200 lux when sunlight is blocked by large buildings or thick tree shade. Retail stores with bright lighting can reach over 4000 lux. The uv levels of indoor environments near windows in summer are comparable to outdoor uv levels in winter. Unless the relative difference can be observed, it is difficult to accurately classify indoor and outdoor environments with only lux and uv.

The final model has a AUC score of 0.925 in difficult semi-indoor and semi-outdoor environments. The score is comparable to the best LightGBM model trained. The Adaboost model cannot be implemented as Dart code. Therefore, the LightGBM model was chosen.



The threshold which results in the best performance was selected. The threshold that resulted in the largest Youden's J statistic and the threshold resulting in the top-left most corner of the above ROC plot was calculated. Both the thresholds were 1.0. That is, the model predicts outdoors only if the probability of the sample being classified is 100% certain to be outdoors.



A third validation test was collected in unknown environments. The final LightGBM model correctly classifies most indoor, semi-indoor, outdoor and semi-outdoor environments. The model failed to classify 263 samples that were collected on a windowsill that is facing full sunshine. However, the model correctly classified indoor samples that were 50 cm away from windows.

The final model is implemented to the app as compiled Dart code.

Unforeseen problems

Unsuitable/Faulty Sensors

We encountered problems with several of the hardware components and had to repurchase many of them. We purchased our initial set of components from DigiKey, an electronics wholesaler based in the US. However, once the components arrived, we realised that the light sensor, UV sensor, and EEPROM were industrial components designed to be soldered onto a PCB by machine. This meant that we had no way of utilising them. To avoid more delays from shipping, we purchased more suitable parts from a local shop. However, we encountered yet another problem where the UV sensor was not responding to different UV levels. We took this back to the shop where it was confirmed to be faulty, and we were forced to purchase a third UV sensor. The problems with the sensors caused significant delays, and this meant that our data analysis was completed much later than we had planned.

Bluetooth LE Bonding

We experienced significant problems with the Bluetooth bonding technology, which caused unstable user experiences during the data sync process. This was not a problem of our implementation, but of the underlying Android platform.

We implemented built-in encryption on the characteristics of the Arduino device. Android tries to automatically bond with the device. However, we had found that the bluetooth bonding offered by android keeps a record of the previous arduino bonding state. This prevented the Arduino from connecting to the app. and also stock phones after the manufacturer packages were removed.

The sync was unstable, where only the next sync attempt after an unsuccessful attempt would initiate the syncing. We had a consistent pattern when the bluetooth syncing failed. When the syncing fails by the underlying Android system, we suspect that the bonding information is cleared, which resets the state. The hypothesis was confirmed when we observed the bonding pattern. When the syncing was initiated the second time, the bonding state was initiated as “bonding”. We removed the built-in encryption to find that the error was removed. When there is no built-in encryption, the bonding state is initiated as “null”, indicating that the bonding information is reset. No syncing problems were observed after this. However, the unexpected problem delayed the hardware development process. We finished the hardware development four weeks later than planned.

AWS Email Limit

Our AWS sending quota for emails was 50 per day. We had assumed this would be enough for us, but as we were developing account creation for the website and app at the same time, we found ourselves hitting this quota. This slowed down development time, as we were not able to create accounts until this quota reset the next day.

CORS Errors

During the implementation process for the website, we encountered several unforeseen challenges. The most significant issue was CORS errors when the website performed API actions. This occurred due to the security restrictions enforced by the browser, which prevented the website from making requests to a different origin. To resolve this, we configured the server to include the necessary CORS headers in its responses.

Restrictive AWS Account Policies

As we have a website rendered entirely on the client side, we opted to use S3 static website hosting but found that our AWS account had a rule which automatically applied a policy to all S3 buckets to prevent unencrypted connections. The problem with this is that S3 static sites do not provide an HTTPS endpoint so we were unable to access the site over the internet. We wanted to disable this so we got approval from the AWS team on the Slack channel, provided that we re-enable it after the course ends.

Results & Evaluation

Achievement of project goals

We have achieved beyond the minimum viable product that the clients requested. We were able to develop the hardware, app, website and server. The clients initially said that barebones UI was sufficient. However, we were able to create a clean, modern and intuitive design for both our app and website.

We also offered proactive solutions to the problem that the researchers were facing. We actively implemented the study and research pipeline. We delivered further value than sending the hardware data to the server. We set up communication between the end users and academia to facilitate myopia research.

We believe that our achievements will have a positive impact on families with myopia.

System testing

Variety of Android Devices and Android Versions

To ensure correctness and consistency, we tested the app on a wide range of Android phones, including Samsung, Oppo, and emulators, covering Android versions 11 to 14. Some of these phones had a much lower resolution screen than others and this allowed us to catch unexpected overflow errors on several occasions. We also discovered varying behaviour for the BLE syncing process. We noted that on some phones, the pairing prompt would show repeatedly, while some did not show it at all. In the end, we disabled encryption which seemed to fix many of these issues. We planned to reimplement encryption at the application layer without relying on encryption from the BLE protocol, but we did not have time to implement this.

API Action Testing

We used API action testing to ensure our application interacts correctly and reliably with our API endpoints. The primary APIs our application interacts with include AWS Cognito and Custom API endpoints. The endpoints were tested for functionality by ensuring the correct data was sent or retrieved based on input parameters. Security was also tested by checking status codes on responses sent by API endpoints when sending requests from authorised and unauthorised users. APIs were tested with different sizes of data to ensure appropriate time duration.

Lambda Timeout

In our project, we use AWS Lambda to handle data backend processing. This introduces considerations around function timeouts and managing tasks that might exceed this duration. One such instance where this situation would occur is when large amounts of samples are posted to the server. To workaround this problem, we break down our data sets into smaller, more manageable chunks that can be processed separately, enhancing performance and reliability. When testing the server, we found that the optimal maximum number of samples to post to the server at one would be 1000.

BLE Throughput

Due to the constrained nature of the Arduino which forced us to use a small 23 byte MTU, and the low throughput design of BLE, we needed to know whether it was feasible to achieve satisfactory speeds with BLE. At first, we were concerned that the app was reading data at only 200 bytes per second, taking an unacceptable 80 seconds to transfer the whole 16KB in the worst case. However, performing a throughput test between the Arduino and a computer gave us a speed of 1400 bytes per second and 11.5 seconds for 16KB. This allowed us to reduce the problem to the app. We discovered that the problem was Android increasing the latency to improve battery life and the fix was to request a high connection priority from Android.

Create Researcher Accounts and Studies

We tested the website on multiple operating systems and web browsers, including Windows 10, Linux, Firefox, Chrome, Brave, and Edge, to ensure compatibility. Creating researcher accounts and studies requires an administrator's security token. The website uses API actions to check the user's token and passes it to the server along with the details of the new researcher or study to create. The test was conducted using multiple systems and accounts with different intervals and inputs to find any malfunction. We have not discovered any errors or limitations during testing other than the limit on the number of account creations due to the AWS Cognito.

Download Raw Data

We tested the download feature using different types of accounts and multiple studies with various participating researchers and data sizes. The raw data with 1,000 entries required approximately 1 second to receive, convert and start downloading the data, while data with 10,000 entries required approximately 4 seconds. We confirmed that non-participating researchers can not access unauthorised studies or download the raw data.

Evaluation

Strengths

Variety of Sensors

The Neox Sens uses a variety of sensors which ensures we are able to get useful data in multiple different environments. We are also less likely to get false positives, as multiple sensors will have to give appropriate readings for us to classify the sample as outdoor.

Near Production Ready

We designed the Neox Sens to be compact and low power. Therefore, if this device were to go into production, the process of creating a custom PCB would not require large changes. The code also runs on bare metal and does not rely on a bloated operating system, allowing for high control over tweaks and optimisations.

The Neox App shows an intuitive layout with robust state management for enhanced user interaction. The app has implemented animations, and user interaction artefacts such as toggle, bottom sheet, and scrolling.

Joining studies is easy and can be done by entering a unique code into the app. Similarly, leaving studies is just as easy and can be done with the press of a button. Everything is handled by asynchronous API calls in the backend, ensuring a seamless and efficient user experience. When a user joins a study, the app communicates with the server to register the child's participation, ensuring all data is synchronised and up-to-date. Similarly, when leaving a study, the API handles the deletion of the child from that study and your phone, maintaining data integrity.

Static Website

Our website utilises client-side rendering, meaning we do not need to use complex server logic to render web pages. This lower server complexity leads to a lower cost for the clients to keep our website running.

Website Usability and Accessibility

Our website design is in line with accessibility standards such as WCAG contrast ratio requirements as well as various design principles. This ensures a usable and accessible website and an improved user experience.

Design-first API Development

From the beginning of the project, we specified the API in a central document which details how API clients authenticate; the expected inputs, outputs, and errors from each API action; and other useful information about the API and its design. This encouraged the server team to specify the API exhaustively and in a clear way so that the app and the website developers can program against the API without having to consult the API developers all the time. This allowed us to work more efficiently.

Multiple Independent Instances of Backend and Website

Since we specify our infrastructure in a text file rather than configuring it through menus on AWS, we could easily run multiple instances of the API and the website so that separate teams could work independently without fear of getting in each other's way. This benefit would be inherited by later developers.

Hosting Multiple Studies

The project description given to us at the start of the semester did not require the studies feature we provide, but we found it was an elegant way to model the consenting process. Furthermore, the same instance of the cloud backend can serve multiple studies at the same time and thereby reduce the costs of operating in such an environment.

Weaknesses

API Actions to Delete Accounts, Children, Samples, and Studies

At the start of the project, we wanted to implement deletion so that researchers could fulfil their obligations under various privacy laws around the world. We currently have lambda functions available to system administrators to clear entire database tables but not any one specific entity.

Exposed hardware

The Neox Sens lacks any kind of case or cover. This means it is exposed to both external conditions that may damage it (e.g. rain) and potentially harm the user wearing it. To fix this issue, we could build a simple case that covers the hardware but leaves spaces where the sensors can still get readings.

Analog Sensor Readings Prone to Error

Since we are currently measuring the voltage exactly once per sample stored, the reading is prone to high variation from noise. To fix this, we should take several readings in a short time and take the average before storing the value.

Classification

The classification of indoor and outdoor time is based on limited training data collected during May 2024. The classification does not take into account seasonal differences in uv, chromatic composition of sunlight, and light intensity. Classification will decrease for data that is beyond the scope of the data collected during the project duration. The second weakness is that the model has low performance when the Neox Sens is in direct sunshine behind a window. The hardware uses an analogue uv sensor which does not differentiate between UVA and UVB. 75% of UVA passes through ordinary glass. The lack of UVB detected by the sensor is not large enough to impact classification. Furthermore, seasonal differences in uv radiation makes classification difficult. A digital uv sensor can be implemented on a specialised printed circuit board. A UVA filter could also be utilised to block UVA from being detected by the sensor.

Unfiltered and Unpaged Sample Fetching from API

Currently, the API returns all samples associated with a child or with a study for child sample fetching and study sample fetching, respectively. As there is a 29s hard cap on lambda function timeouts, there is a risk of not being able to retrieve anything once there are too many samples for a child or a study. We could allow API callers to retrieve only samples within a certain timestamp range. An alternative solution is to implement “pagination” in the sample fetching action and require callers to call the action again with a special token to retrieve the next “page”. The latter solution requires more state on the server which we would prefer not to have, but it can be less work on the caller since pagination would return a predictable number of samples per call.

Potentially Confusing Admin Privileges

Admins are just users part of both the “researchers” and “admins” groups. As mentioned in the specification and design section, we do not allow admins access to all studies. We do this so that the system can easily change to allow researchers to enlist non-researcher admin staff to manage studies. Instead, admins must add themselves to each study for which they wish to see the data. This weakness is easily dealt with, however, by just changing a flag in the authentication section of the API action to download samples.

API Actions Cannot Handle Spikes

A serverless architecture is characterised by easily scalable components but we currently have a bottleneck in our database which has a limit on the number of concurrent connections. This weakness can be mitigated by pooling connections using another AWS service called RDS Proxy.

Future Work

Further features

Discarding Samples While Stationary

As an optimisation to decrease the amount of data sent to Neox App, we could stop recording samples during long periods of inactivity as detected from IMU. We would expect the IMU to be stationary when the Neox Sens is not being worn, and this would also prevent it from recording inaccurate data. Lowering the amount of data collected would also significantly speed up the BLE syncing.

Resettable Authentication Code

If a parent loses the authentication code for the BLE syncing, the Neox Sens will become unusable. It would be great to have a backup method to view or change the authentication code. Since we do not want to have a display on the Neox Sens, the interface for this would have to be in the app. In this case, we could have a button on the Neox Sens that temporarily transitions it into a special code resetting mode which communicates with the app over BLE.

Gamification and Reward System

The purpose of gamification and rewards would be to encourage children to keep using the app and spend more time outdoors. A system could be put in place that treats time spent outdoors as something equivalent to experience for levelling up or virtual currency. The child would then unlock different rewards that could be something like a virtual pet. This would increase the retention rate and help collect more data.

Visualisation for Data on Website

Interactive charts and graphs could be used to improve the usability of the collected data. These visualisations would assist researchers to analyse the relationship between light exposure patterns, UV levels, time spent outdoors, and myopia development by outlining patterns and anomalies.

Comment Feature

Implementing a comment or discussion forum feature for each study could improve research efficiency. This would allow researchers to share findings and ideas related to the collected data and its implications for myopia research. This approach is more straightforward and efficient for providing feedback and keeping records compared to collaborating via email.

System Enhancements

Better Timekeeping

The Neox Sens currently relies on the internal real time clock to count time and uses the EEPROM to continue approximately where it left off after losing power. This means that time will not pass while the Neox Sens is not powered. Another problem is that the internal RTC is not accurate and can accumulate up to 20 minutes of error per day. A functioning external RTC powered by a separate coin battery would solve these issues.

Multiple Devices

A possible improvement that could be made is allowing children to use multiple Neox Sens devices. This would lead to increased accuracy in data readings by taking the average across the devices. The database would need to be changed to accommodate for the additional devices per child.

Time Series Database

The project could be optimised to use a time-series database since we are collecting time-stamped sensor data. When dealing with a vast amount of data samples, we could see performance enhancements in speed and efficiency. For example, if a child were to wear the device during school hours of 9 am to 5 pm, the device would collect 540 samples. For a school week, there would be 2700 samples. Over a month, a single child would generate approximately 10800 samples. In reality, multiple children would wear devices, so this number would be multiplied by many times. As the user base grows, the need to switch to a time series database will be amplified.

Sign-up Options

Additional login methods via Google and Facebook could be implemented to make sign-up easier and more convenient.

Conclusion

We develop the Neox Sens, a wearable light monitor prototype based on an Arduino. We aim to assist in understanding the correlation between light exposure patterns and myopia development in children. The collected data can be synchronised with a cloud database through a dedicated mobile phone application. This application allows parents to monitor and analyse their child's light exposure patterns. Additionally, researchers can create studies that parents can opt into, allowing researchers to access and analyse their data.

Our hardware Neox Sens has the ability to measure different types and qualities of light, such as UV, lux, colour and colour temperature. It also has an in-built accelerometer to help us classify whether the child is outdoors or not.

The mobile application Neox App is developed to complement our hardware and features visual representations of outdoor time. The app provides parents with clear insights into their child's light exposure without being there to watch over them. Additionally, the parents can sync their data to Neox Cloud. Furthermore, for the benefit of researchers, parents can choose to join a study and share anonymised data to help with ongoing research into myopia.

Researchers can access data through a web interface Neox Lab, where they can choose to download data in a CSV format. Using the web interface, researchers are able to create and monitor ongoing studies. However, because we never collect or send any unique and personal fields from the users, researchers are never able to access anything that would cause ethical issues.

Overall our project has met the requirements set out by the client and in some cases exceeded them. We have developed a cross-function platform spanning multiple domains from the hardware to the cloud. We offer a comprehensive system for monitoring and analysing light patterns. Our project, Neox, aims to contribute to the efforts to slow the progression of myopia in children.

References

Ashby, B. (2023). *Myopia: Aotearoa's inaugural report card*. NZ Optics. https://www.nzoptics.co.nz/live-articles/myopia-aotearoa-s-inaugural-report-card

Holden, B. A., Fricke, T. R., Wilson, D. A., Jong, M., Naidoo, K. S., Sankaridurg, P., Wong, T. Y., Naduvilath, T. J., & Resnikoff, S. (2016). Global Prevalence of Myopia and High Myopia and Temporal Trends from 2000 through 2050. Ophthalmology, 123(5), 1036–1042. https://doi.org/10.1016/j.ophtha.2016.01.006

Holden, B., Sankaridurg, P., Smith, E., Aller, T., Jong, M., & He, M. (2013). Myopia, an underrated global challenge to vision: where the current data takes us on myopia control. Eye, 28(2), 142–146. https://doi.org/10.1038/eye.2013.256

Lewis, R. (2024). *Myopia.* Encyclopedia Britannica. https://www.britannica.com/science/myopia

Recko, M., & Stahl, E. D. (2015). Childhood myopia: epidemiology, risk factors, and prevention. Missouri Medicine, 112(2), 116–121.

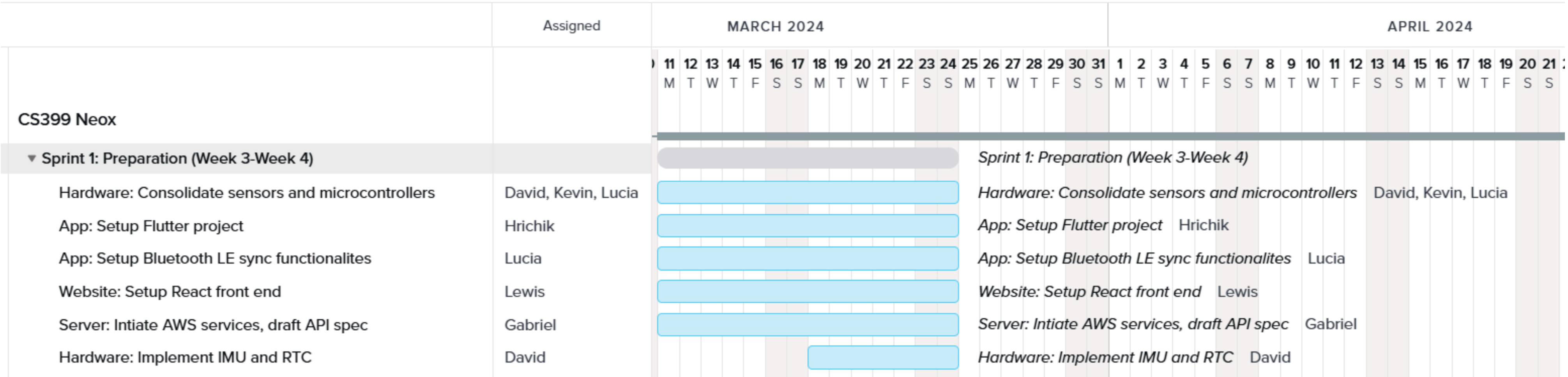
Sherwin, J. C., Reacher, M. H., Keogh, R. H., Khawaja, A. P., Mackey, D. A., & Foster, P. J. (2012). The Association between Time Spent Outdoors and Myopia in Children and Adolescents. Ophthalmology, 119(10), 2141–2151. https://doi.org/10.1016/j.ophtha.2012.04.020

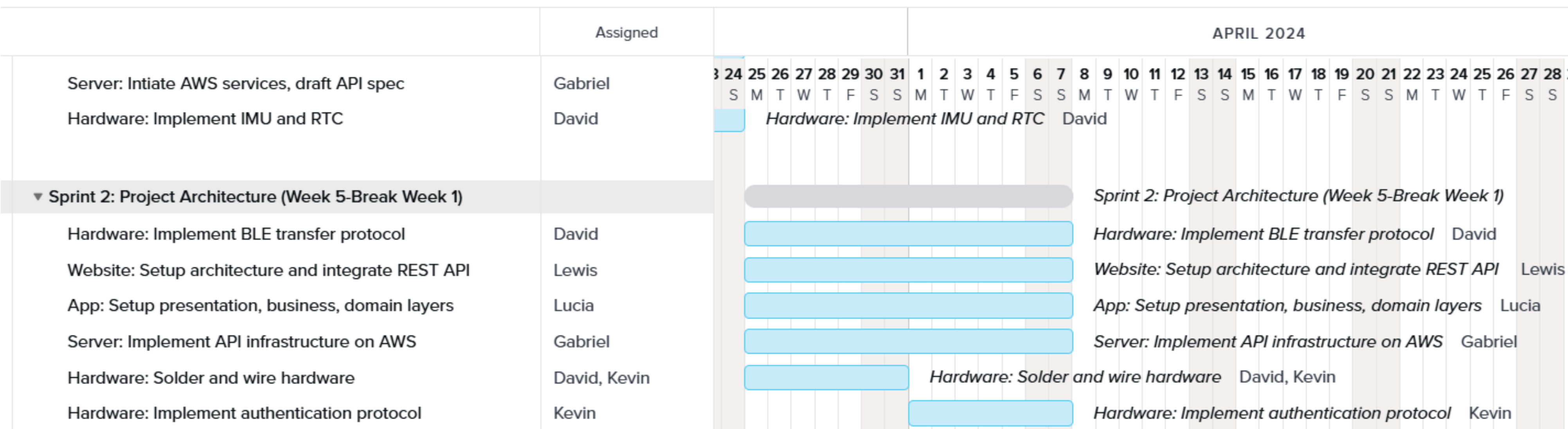
Williams, K., & Hammond, C. (2019). High myopia and its risks. Community Eye Health, 32(105), 5–6. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6688422/

Zhang, J., & Deng, G. (2019). Protective effects of increased outdoor time against myopia: a review. Journal of International Medical Research, 48(3), 030006051989386. https://doi.org/10.1177/0300060519893866

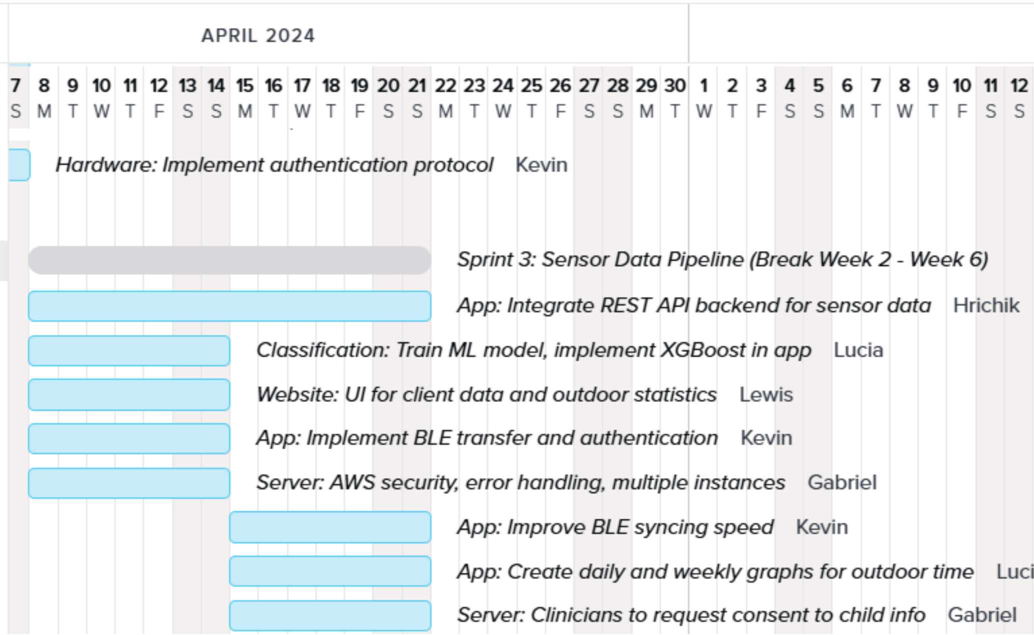
Appendices

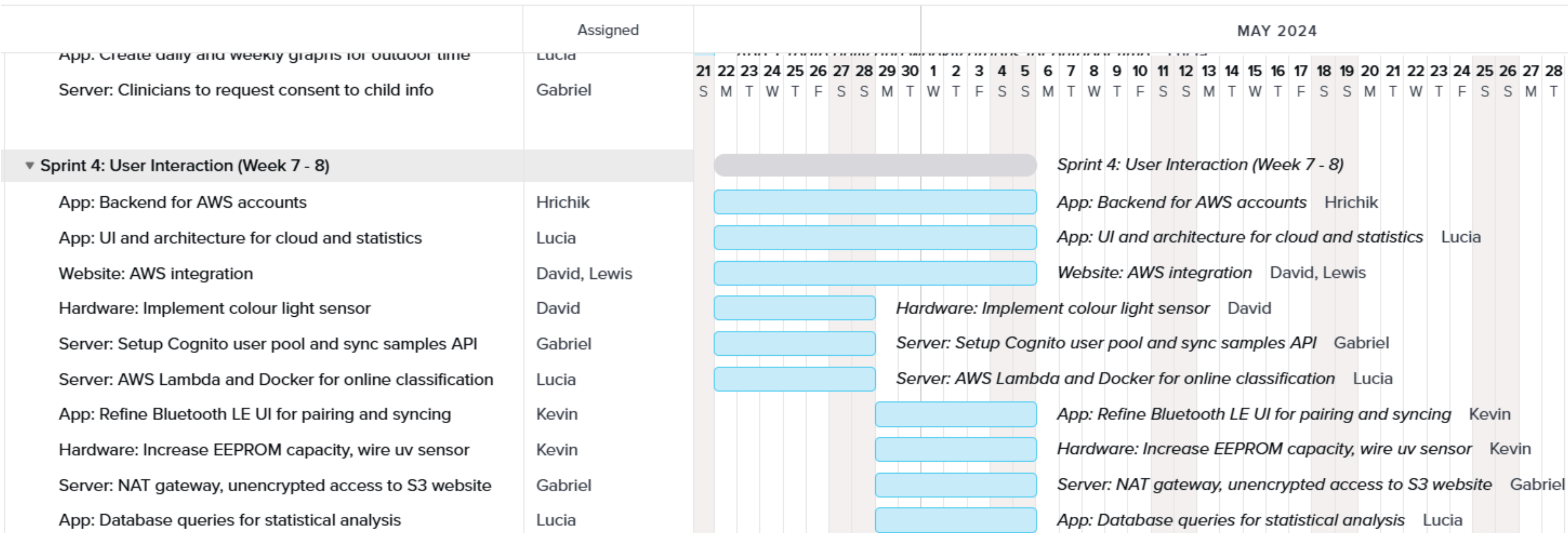
Gantt Chart

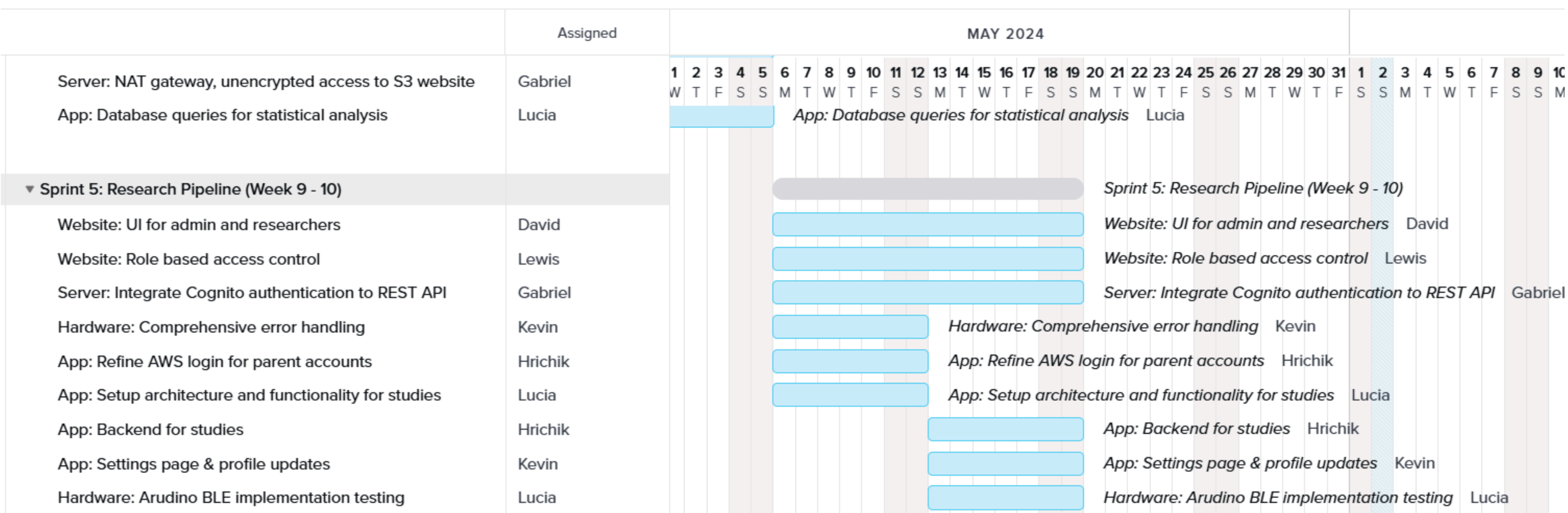


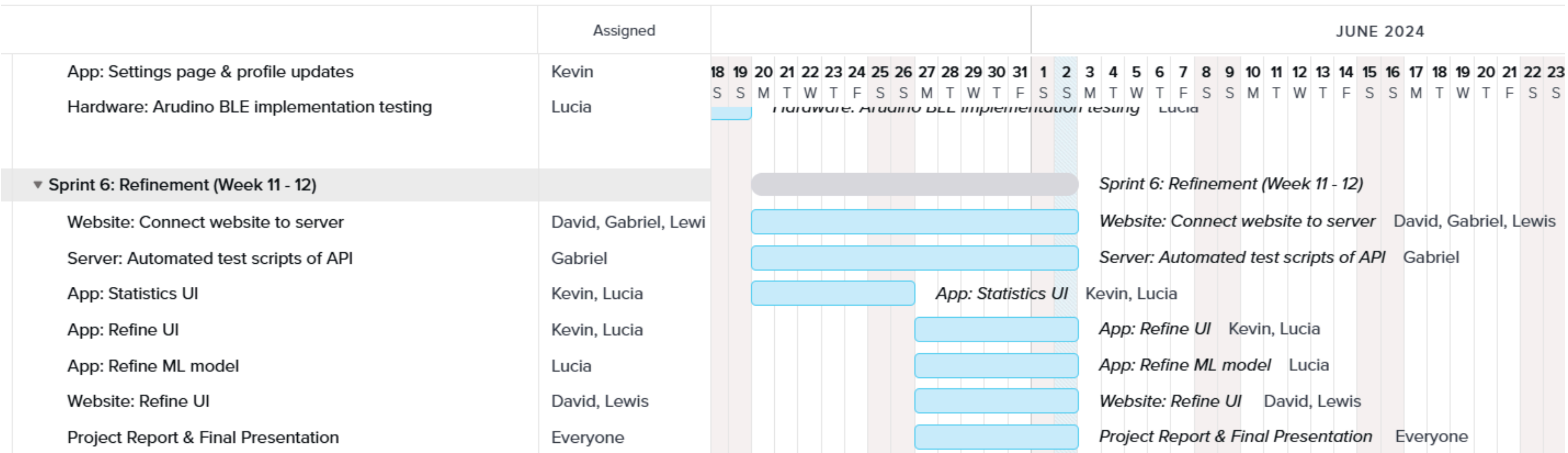


	Assigned
Server: Implement API infrastructure on AWS	Gabriel
Hardware: Solder and wire hardware	David, Kevin
Hardware: Implement authentication protocol	Kevin
▾ **Sprint 3: Sensor Data Pipeline (Break Week 2 - Week 6)**	
App: Integrate REST API backend for sensor data	Hrichik
Classification: Train ML model, implement XGBoost in app	Lucia
Website: UI for client data and outdoor statistics	Lewis
App: Implement BLE transfer and authentication	Kevin
Server: AWS security, error handling, multiple instances	Gabriel
App: Improve BLE syncing speed	Kevin
App: Create daily and weekly graphs for outdoor time	Lucia
Server: Clinicians to request consent to child info	Gabriel




Assigned
MAY 2024
Server: Clinicians to request consent to child info
Gabriel
Sprint 4: User Interaction (Week 7 - 8)
App: Backend for AWS accounts
Hrichik
App: UI and architecture for cloud and statistics
Lucia
Website: AWS integration
David, Lewis
Hardware: Implement colour light sensor
David
Server: Setup Cognito user pool and sync samples API
Gabriel
Server: AWS Lambda and Docker for online classification
Lucia
App: Refine Bluetooth LE UI for pairing and syncing
Kevin
Hardware: Increase EEPROM capacity, wire uv sensor
Kevin
Server: NAT gateway, unencrypted access to S3 website
Gabriel
App: Database queries for statistical analysis
Lucia


Assigned
MAY 2024
Server: NAT gateway, unencrypted access to S3 website
Gabriel
App: Database queries for statistical analysis
Lucia
Sprint 5: Research Pipeline (Week 9 - 10)
Website: UI for admin and researchers
David
Website: Role based access control
Lewis
Server: Integrate Cognito authentication to REST API
Gabriel
Hardware: Comprehensive error handling
Kevin
App: Refine AWS login for parent accounts
Hrichik
App: Setup architecture and functionality for studies
Lucia
App: Backend for studies
Hrichik
App: Settings page & profile updates
Kevin
Hardware: Arudino BLE implementation testing
Lucia


Assigned
JUNE 2024
App: Settings page & profile updates
Kevin
Hardware: Arudino BLE implementation testing
Lucia
18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S
Sprint 6: Refinement (Week 11 - 12)
Website: Connect website to server
David, Gabriel, Lewi
Server: Automated test scripts of API
Gabriel
App: Statistics UI
Kevin, Lucia
App: Refine UI
Kevin, Lucia
App: Refine ML model
Lucia
Website: Refine UI
David, Lewis
Project Report & Final Presentation
Everyone
Sprint 6: Refinement (Week 11 - 12)
Website: Connect website to server
David, Gabriel, Lewis
Server: Automated test scripts of API
Gabriel
App: Statistics UI
Kevin, Lucia
App: Refine UI
Kevin, Lucia
App: Refine ML model
Lucia
Website: Refine UI
David, Lewis
Project Report & Final Presentation
Everyone

Link to Final Build

App: Github APK

Release Neox App v1.0.3 · uoa-compsci399-s1-2024/capstone-project-2024-s1-team-14-neox

Website: AWS link
http://neox-frontend-all-dev.s3-website-ap-southeast-2.amazonaws.com

Link to Video

https://drive.google.com/file/d/1kAvbrHDLL6tkFGh1Y7se45FLSYbGaR_o/view?usp=sharing

Classification

Training Samples

Outdoors

Semi outdoors: Walk in Auckland CBD
Collection start: 17:29 6 May 2024

Outdoors: Walk to primary school
Collection start: 8:46, 7 May 2024

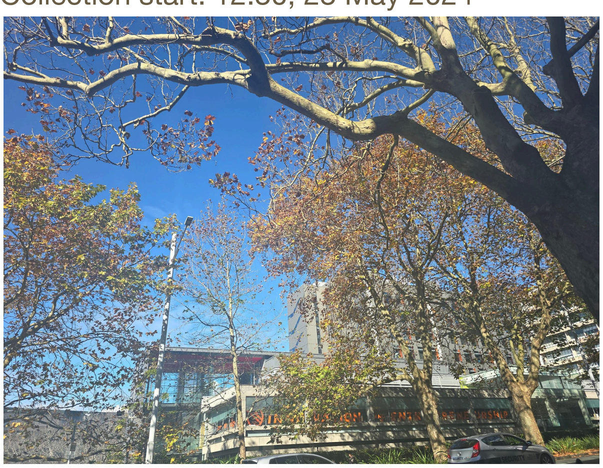
Outdoors: Grass field
Collection start: 14:53, 7 May 2024

Semi outdoors: Under thick tree shade
Collection start: 14:58 7 May 2024

Outdoors: Walk home from primary
Collection start: 15:04, 7 May 2024

Semi-outdoors: Between tall buildings
Collection start: 16:19, 9 May 2024

Outdoors: Walk in Auckland CBD
Collection start: 12:30, 23 May 2024



Outdoors: Under partial shade Albert Park
Collection start: 14:21, 23 May 2024

Outdoors: Walk in Auckland CBD, drizzly
Collection start: 12:58, 27 May 2024

Semi-outdoors: Partial translucent roofing
Collection start: 13:08, 27 May 2024

Outdoors: Walk home from primary
Collection start: 15:40, 28 May 2024

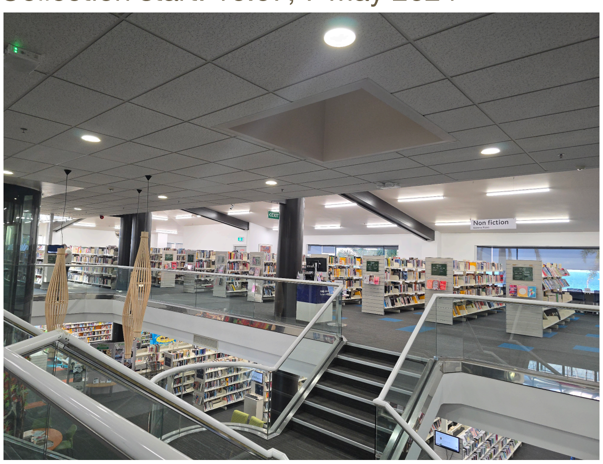
Outdoors: Sunrise
Collection start: 7:40, 31 May 2024

Indoors

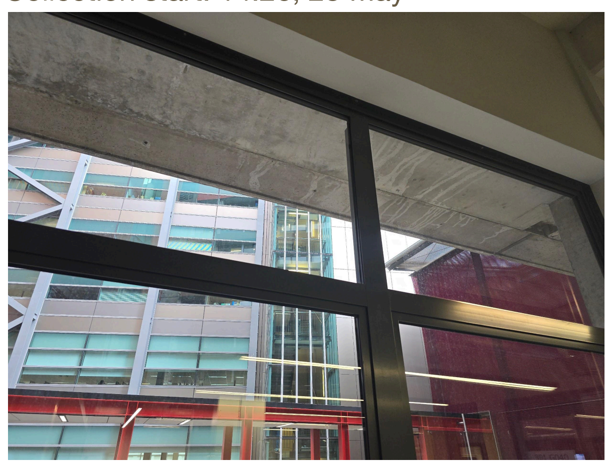
Indoors: Bus at sunset
Collection start: 17:29, 6 May 2024

Indoors: Library children's area (3000+ lux)
Collection start: 15:58, 7 May 2024

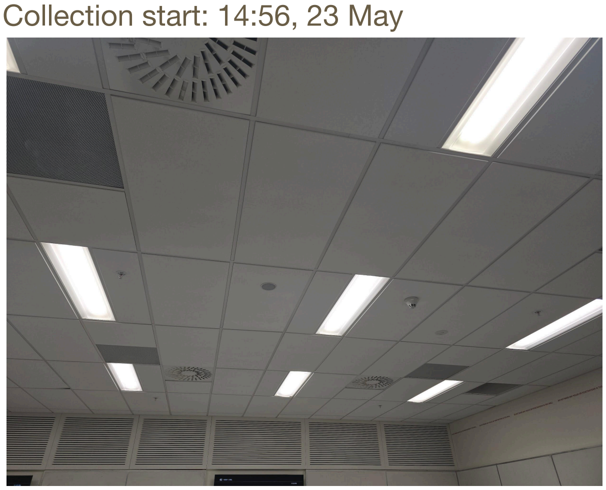
Indoors: Library general area
Collection start: 16:07, 7 May 2024



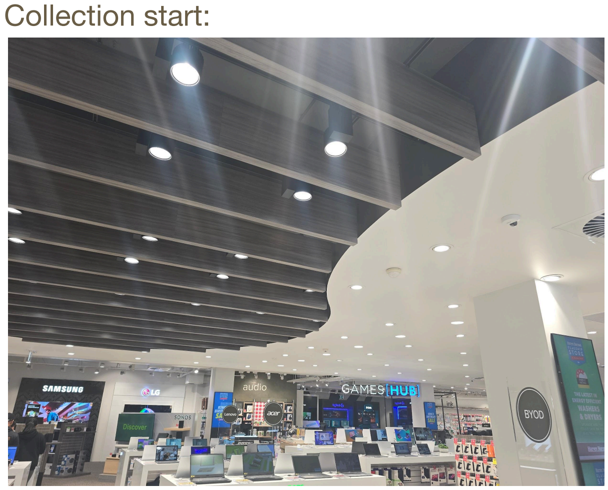
Semi-indoors: 1m from large window
Collection start: 14:26, 23 May



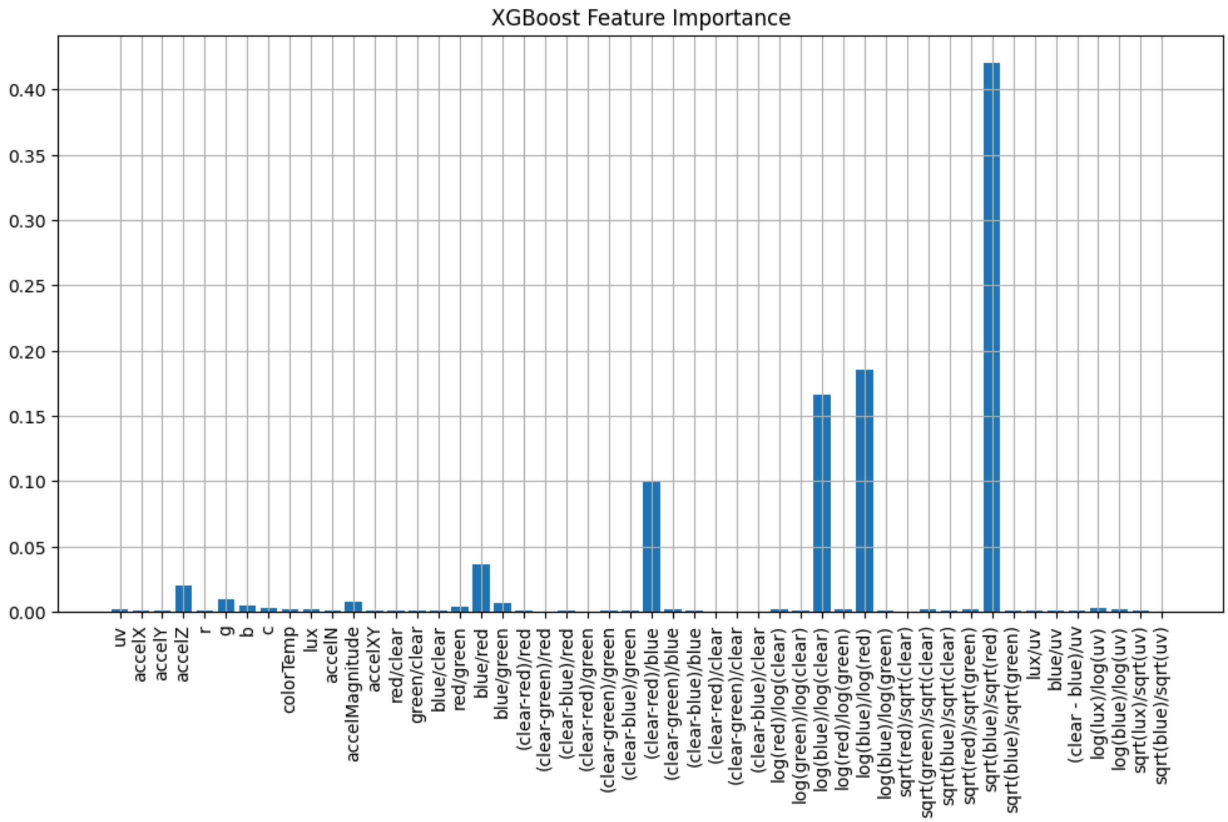
Indoors: Classroom
Collection start: 14:56, 23 May

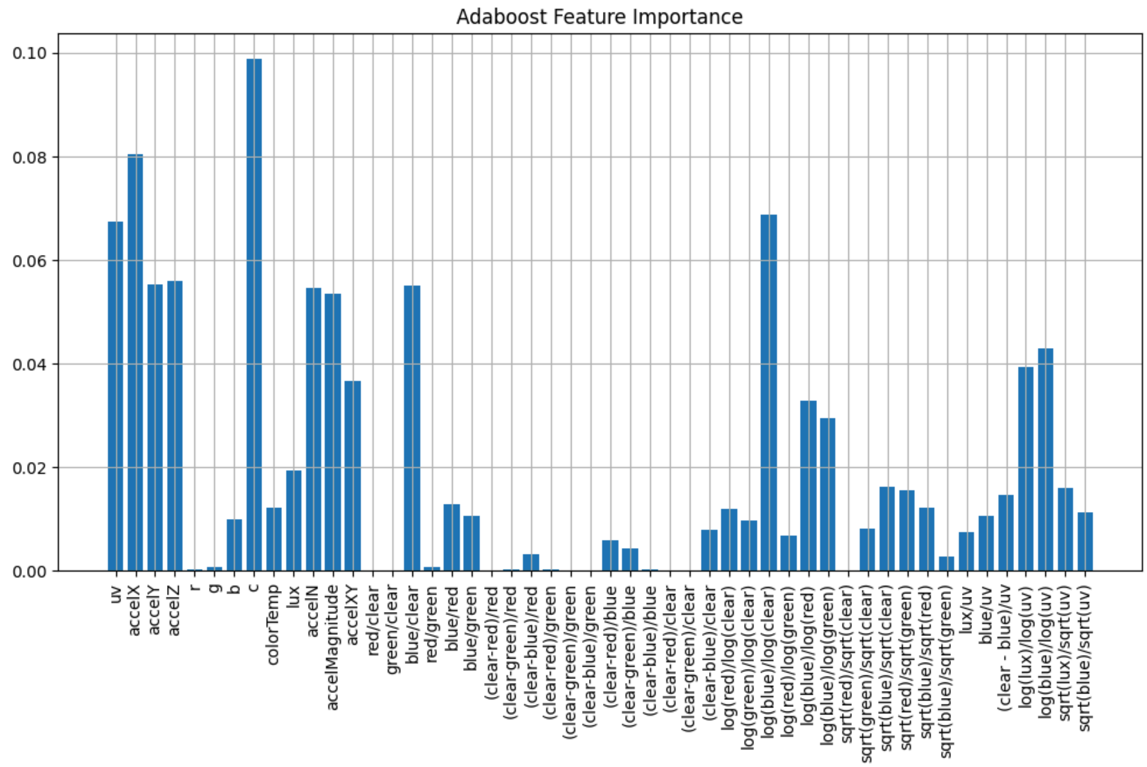


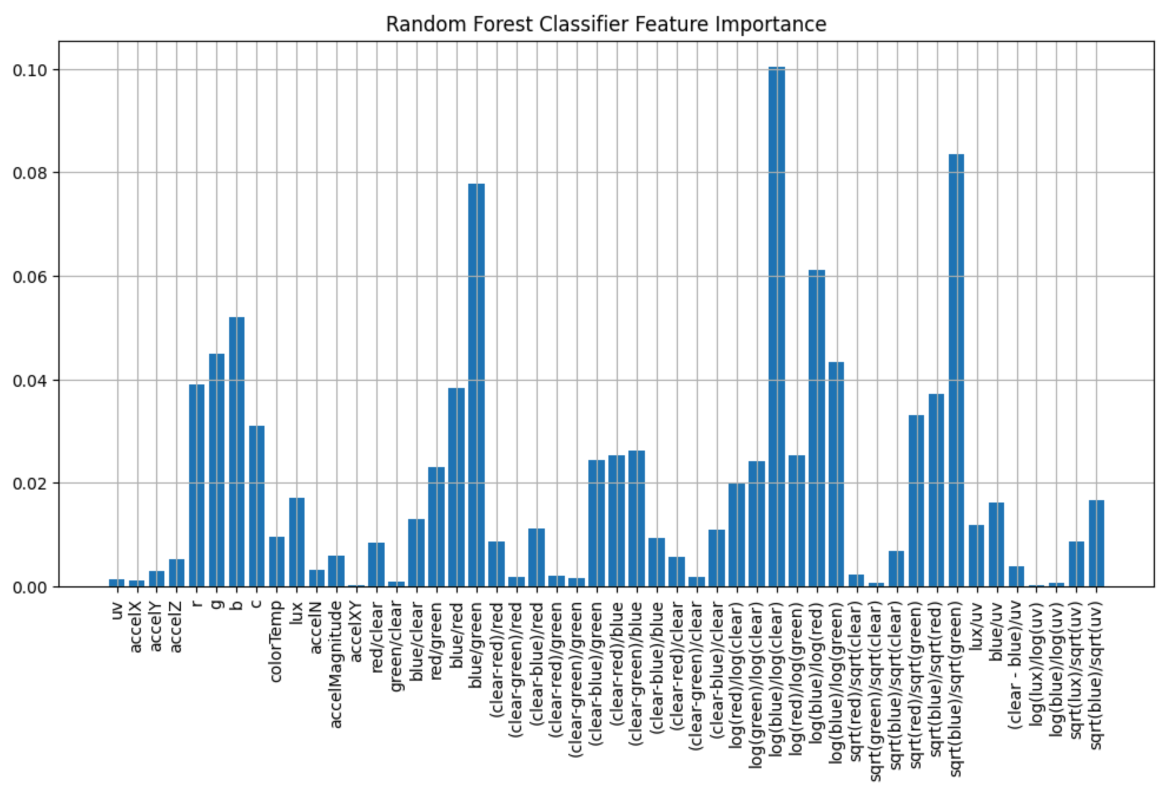
Indoors: Bright retail store (4000+ lux)
Collection start:



Feature Importance for Different Models




Adaboost Feature Importance
0.10
0.08
0.06
0.04
0.02
0.00
uv
accelX
accelY
accelZ
r
g
b
c
colorTemp
lux
accelN
accelMagnitude
accelXY
red/clear
green/clear
blue/clear
red/green
blue/red
blue/green
(clear-red)/red
(clear-green)/red
(clear-blue)/red
(clear-red)/green
(clear-green)/green
(clear-blue)/green
(clear-red)/blue
(clear-green)/blue
(clear-blue)/blue
(clear-red)/clear
(clear-green)/clear
(clear-blue)/clear
log(red)/log(clear)
log(green)/log(clear)
log(blue)/log(clear)
log(red)/log(green)
log(blue)/log(red)
log(blue)/log(green)
sqrt(red)/sqrt(clear)
sqrt(green)/sqrt(clear)
sqrt(blue)/sqrt(clear)
sqrt(red)/sqrt(green)
sqrt(blue)/sqrt(red)
sqrt(blue)/sqrt(green)
lux/uv
blue/uv
(clear - blue)/uv
log(lux)/log(uv)
log(blue)/log(uv)
sqrt(lux)/sqrt(uv)
sqrt(blue)/sqrt(uv)


Random Forest Classifier Feature Importance
0.10
0.08
0.06
0.04
0.02
0.00
uv
accelX
accelY
accelZ
r
g
b
c
colorTemp
lux
accelN
accelMagnitude
accelXY
red/clear
green/clear
blue/clear
red/green
blue/red
blue/green
(clear-red)/red
(clear-green)/red
(clear-blue)/red
(clear-red)/green
(clear-green)/green
(clear-blue)/green
(clear-red)/blue
(clear-green)/blue
(clear-blue)/blue
(clear-red)/clear
(clear-green)/clear
(clear-blue)/clear
log(red)/log(clear)
log(green)/log(clear)
log(blue)/log(clear)
log(red)/log(green)
log(blue)/log(red)
log(blue)/log(green)
sqrt(red)/sqrt(clear)
sqrt(green)/sqrt(clear)
sqrt(blue)/sqrt(clear)
sqrt(red)/sqrt(green)
sqrt(blue)/sqrt(red)
sqrt(blue)/sqrt(green)
lux/uv
blue/uv
(clear - blue)/uv
log(lux)/log(uv)
log(blue)/log(uv)
sqrt(lux)/sqrt(uv)
sqrt(blue)/sqrt(uv)