



ClubHQ

Project Engineering

Year 4

Elaine Killalea

Bachelor of Engineering (Honours) in Software and

Electronic Engineering

Atlantic Technological University

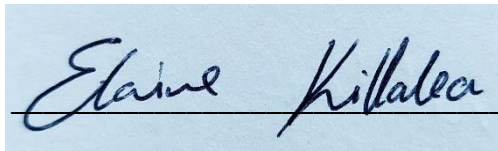
2022/2023

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Atlantic Technological University.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

I acknowledge the use of ChatGPT to explain and recommend technologies and coding techniques, as well as to identify and resolve errors in code. Examples of frequently used prompts are listed in the appendix of this report.

A handwritten signature in blue ink on a light blue background. The signature reads "Elaine Killalea" in a cursive script. The signature is written over a horizontal line.

Acknowledgements

I would like to thank all my final year lecturers for their help with this project. Especially my supervisor Brian O'Shea for his guidance.

I would also like to thank my classmates for all of their advice and help during this project.

Table of Contents

1	Summary	6
2	Poster	8
3	Introduction	9
4	Project Architecture.....	10
5	Project Plan – Teamwork.com	11
6	Hardware	12
6.1	ESP32 WiFi.....	12
6.2	Fingerprint Sensor	12
6.2.1	Enrolling members.....	13
6.2.2	Logging attendance.....	13
7	Amazon Web Services (AWS).....	15
7.1	Connection and Certificates	15
7.2	Lambda code	15
8	Mongo Atlas	17
8.1	Collections	17
9	Next.js Webapp.....	19
9.1	API folder	19
9.1.1	API database routes	19
9.1.2	NextAuth authentication	19
9.1.3	Google Provider	20
9.2	Pages and server-side props	20
9.3	Stale-While-Revalidate (SWR) research	21
9.4	Components	22

9.4.1	Calendar	22
9.4.2	Profile	22
9.4.3	Student List	22
10	Vercel Deployment	24
11	Ethics	26
12	Conclusion	27
13	Appendix	28
13.1	Source code	28
13.2	Frequently used websites	28
13.3	ChatGPT	28

1 Summary

ClubHQ is a full-stack web application designed to simplify the management of club members and their class attendance.

This project uses the Next.js framework for both its frontend and backend, making use of its API folder and routing system. It connects to an online database, MongoDB Atlas, which stores member and attendance data that can be viewed using the ClubHQ web app. This webapp can be accessed from a PC or mobile device for ease of use. The fingerprint scanner is used to enroll members and log their attendance on arrival at class. The ESP32 board sends the members id to the database through AWS IoT Core and AWS Lambda where it is immediately available to be viewed on the app.

Once logged in with their Google account, the app allows members to view and edit their profile. They can also view a heatmap calendar tracking their attendance. An admin account can view and edit their profile and calendar. They also have access to a list of members, where they can view, edit and delete members.


This project incorporates many technologies such as React.js, Next.js, Vercel, MongoDB Atlas, AWS IoT Core, AWS Lambda, Node.js, Javascript and the C language. This project was managed using Git and Github, and for tracking the project timeline I used Teamwork to create Gantt charts of my tasks.

2 Poster

Elaine Killalea – G00370088
 BEng Software and Electronic Engineering
 Final Year Project

Ulster
 Technological
 University

ClubHQ



ClubHQ is a full-stack web application to simplify the management of club members and their class attendance. The aim was to centralise a club's data and reduce paper records that can be lost or damaged.

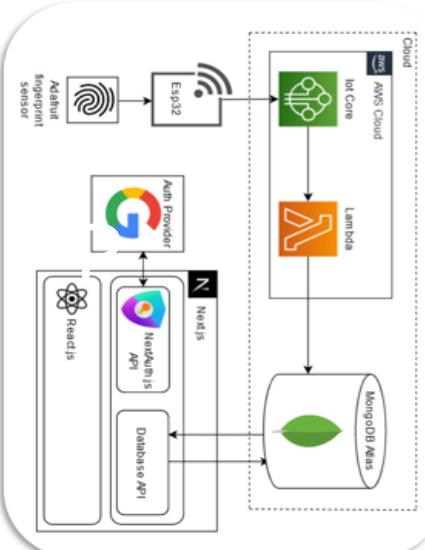
Technical Overview

Authentication
 Members that have been set up by the admin can log in using the NextAuth.js Google provider.

Web App
 The Next.js web app allows members to view their attendance, progression and profile. Admin can also manage members and their profiles.

MongoDB
 The database has two collections. One for classes and student IDs that attended and one for members' personal information.

AWS
 Using AWS security certificates, the ESP32 securely publishes messages to IoT Core which triggers a Lambda function to send the data to MongoDB Atlas using Node.js.




Hardware

The Adafruit fingerprint scanner stores and compares students' fingerprints. The ESP32 Wi-Fi module connects to AWS and sends the student id using the configured Wi-Fi network.

Results




To send the data from the ESP32 to Mongo Atlas, an AWS Lambda function was needed. AWS IoT Core subscribes to messages from the ESP32 and using Node.js, transfers these messages to the database in the correct JSON format.






Deployment

The web app has been deployed on Vercel instead of AWS as it does not require an EC2 instance to be set up and running. When code is committed to the main branch on Github a Vercel build is triggered. If successfully built the app is deployed to the designated URL.

Admin can


-  Enrol member fingerprints
-  Add/delete members
-  Manage members' attendance, profile and progress

Members can

-  Log attendance with fingerprint
-  Edit profile
-  View attendance, profile and progress

See for yourself

<https://clubh.vercel.app.com>



3 Introduction

ClubHQ is a full stack web application project that aims to simplify the management of members for a martial arts club. The aim is to create an application that is accessible by admin and members which features attendance tracking, storing personal details and progress monitoring. Many small clubs still use paper records that can be easily misplaced or damaged and pose a risk to members' privacy. All information is stored securely in the cloud, accessible only to authorized club members. This solution can reduce paper waste and printing costs for the club. It also means that members and admin can view their information from any device.

This project also has a fingerprint scanner which provides a convenient and reliable way for members to log their attendance. Members scan their fingerprint when they arrive at class and their attendance is automatically sent to the database. This eliminates the need for admins to manually mark attendance, reducing errors and saving time.

A webapp was chosen rather than a mobile app as it gives members access to their data from anywhere on mobile devices and admin can use the app on larger screens such as computers for completing longer tasks such as enrolling students. To log attendance I considered NFC on mobile phones but this means members would be unable to do this if they forget their devices or if the member is a child without a phone. By using a fingerprint sensor, members do not have to worry about forgetting devices for training and each one is unique to that member.

This project includes learning and implementing knowledge about databases, Next.js frontend and backend, serverless functions, Amazon Web Services(AWS) cloud services and connecting hardware to cloud services using WiFi.

4 Project Architecture

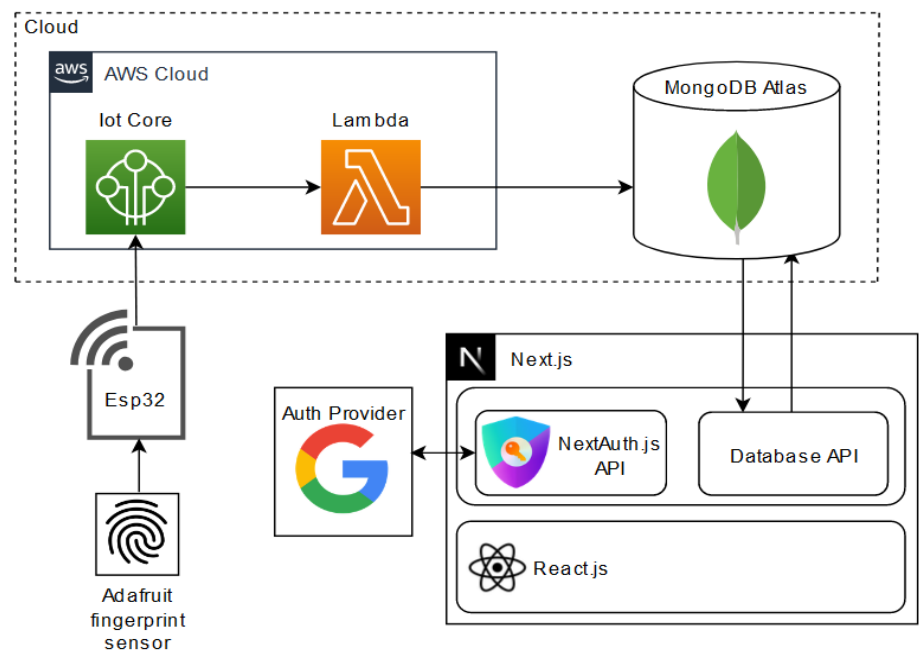
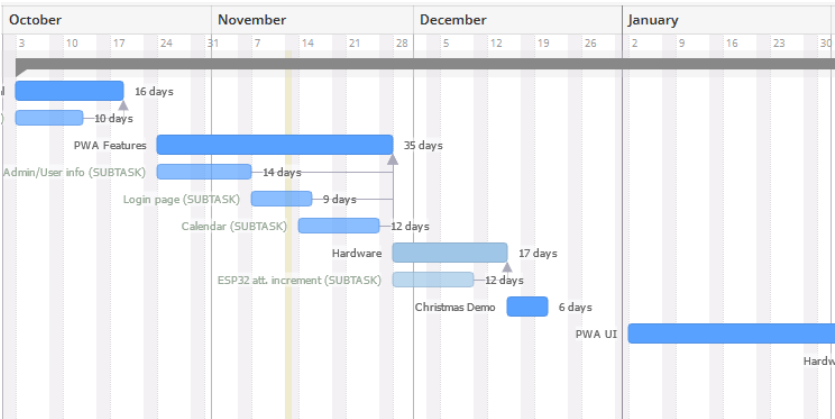


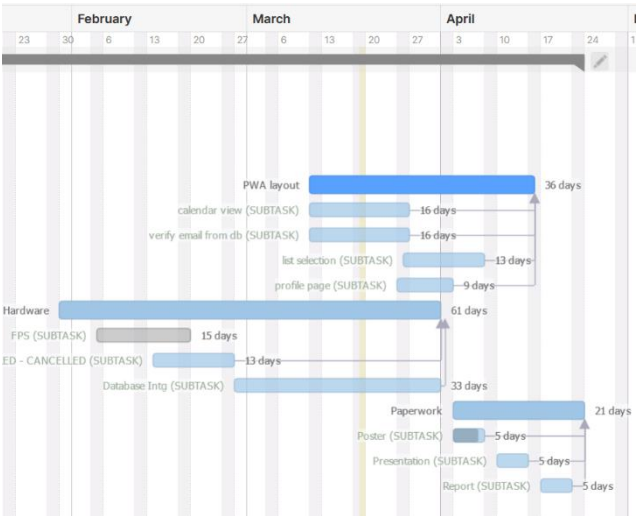
Figure 4-1 Architecture Diagram

5 Project Plan – Teamwork.com

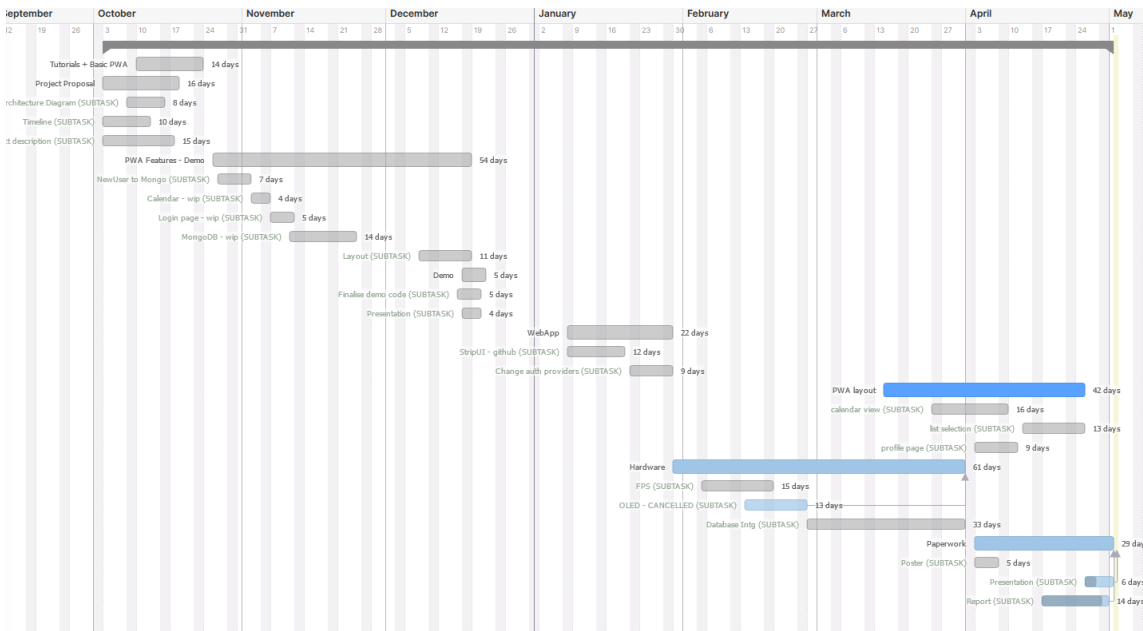
Starting timeline



Second semester



Final timeline



6 Hardware

This project uses an Adafruit fingerprint sensor to allow club members to log their attendance. It is connected to an ESP32 Wi-Fi module which sends the fingerprint sensor data to the clubs cloud database on Mongo Atlas.

6.1 ESP32 WiFi

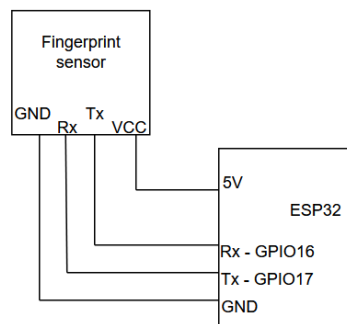


Figure 6-1 Sensor and ESP32 wiring diagram.

It is important to note that pins 16 and 17 are designated as Tx2 and Rx2 meaning they use the name Serial2 in the code. This caused a blocker during the project where I didn't realise there was no Serial1 port.

6.2 Fingerprint Sensor

The fingerprint sensor can store up to 162 fingerprints in the onboard FLASH memory [1]. It has a supply voltage of 3.6-6 VDC and operating current of 120mA max meaning it can be run on the ESP32 5V pin .

The Adafruit provide an interfacing library for their sensor with the complete code for enrolling and searching for fingerprints [2]

6.2.1 Enrolling members

The ESP32 board is uploaded with the enrolment code. Admin must enter a number for the finger ID they wish to assign to the person.

```
Ready to enroll a fingerprint!
Please type in the ID # (from 1 to 127) you want to save this finger as...
Enrolling ID #6
Waiting for valid finger to enroll as #6
.....
Image taken
Image converted
Remove finger
ID 6
Place same finger again
.....
Image converted
Creating model for #6
Prints matched!
ID 6
Stored!
```

Code 6-3 Serial monitor output

6.2.2 Logging attendance

When the student arrives at class they scan their fingerprint. If the scan was successful they are shown the below message of success along with the confidence rating or the match.

```
Connecting to AWS IOT
AWS IoT Connected!
Waiting for valid finger...
Sensor contains 5 templates
Date: 30-April-2023

Found ID #1 with confidence of 118
Date: 30-April-2023
```

Code 6-4 Serial monitor output

When a fingerprint is scanned it returns a finger object. Finger id is passed to a function that will publish it as a JSON document to the database. The date is also displayed when a fingerprint is successful using the “time.h” library[3]. The date is not logged with the fingerprint until the Lambda code to keep the communication shorter for the ESP32

INIT_START Runtime Version: nodejs:16.v12 Runtime Version ARN: arn:aws:lambda:eu-west-1::runtime:734d712d91ebc245bd01c52ae85b4025a7885efd1b8c87c0ce49f3e47d2116a4			
START RequestId: f8a17d28-8501-4093-8e10-8b1e36ac92ca Version: \$LATEST			
2023-04-30T19:48:31.701Z	f8a17d28-8501-4093-8e10-8b1e36ac92ca	INFO	Connected to MongoDB cluster
2023-04-30T19:48:31.862Z	f8a17d28-8501-4093-8e10-8b1e36ac92ca	INFO	Connection to MongoDB cluster closed
END RequestId: f8a17d28-8501-4093-8e10-8b1e36ac92ca			
REPORT RequestId: f8a17d28-8501-4093-8e10-8b1e36ac92ca Duration: 1269.17 ms Billed Duration: 1270 ms Memory Size: 128 MB Max Memory Used: 81 MB Init Duration: 361.73 ms			

Figure 6-4 CloudWatch Log AWS

_id: ObjectId('644d8a54aebf521fd636ff3b')
date: "2023-04-28"
studentIDs: Array

_id: ObjectId('644ec60f1fd5d5b838884eb2')
date: "2023-04-30"
studentIDs: Array
0: 1

Figure 6-4 MongoDB document

7 Amazon Web Services (AWS)

On the Arduino IDE, the following libraries were imported.

PubSubClient, Nick O’Leary [4], publish/subscribe messaging with a MQTT server.

ArduinoJson, Benoit Blanchon[5], JSON serialisation/deserialization for Arduino.

7.1 Connection and Certificates

AWS certificates are used to make a secure connection between the code and AWS IoT Core.

Here an MQTT client is configured to connect to an AWS IoT endpoint securely.

```
WiFiClientSecure net = WiFiClientSecure();
PubSubClient client(net);

// Configure WiFiClientSecure to use the AWS IoT device credentials
net.setCACert(AWS_CERT_CA);
net.setCertificate(AWS_CERT_CRT);
net.setPrivateKey(AWS_CERT_PRIVATE);
```

- The root CA (Certificate Authority) is a digital certificate used to verify the identity of a website. When you connect to a website using SSL/TLS, your browser checks the root CA certificate to ensure that communication is secure.
- The client certificate is a digital certificate that used to authenticate a client device. When a client connects to a server, the server checks that the client is authorized to access the server using the client certificate.
- The private key is a secret key that decrypts data that has been encrypted with a public key. It ensures that communication is secure.

7.2 Lambda code

AWS Lambda is serverless service that allows users to create functions and upload them to AWS Lambda which executes the function. ClubHQ uses this service to execute Node.js code that makes a connection to our MongoDB database. It uses the MongoDB Node.js driver installed in

the node_modules folder. The code is edited in VSCode and a zipped version is uploaded to Lambda.

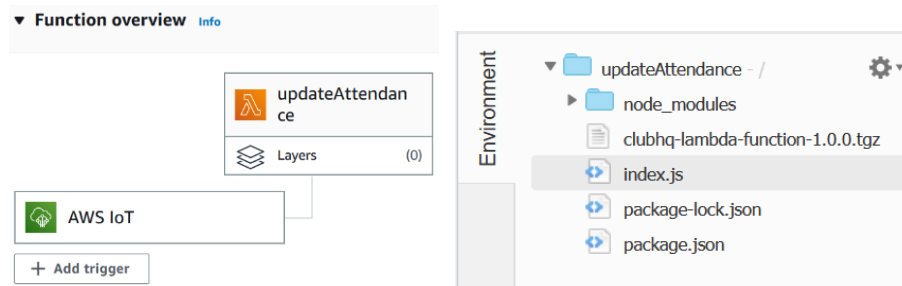


Figure 7-1 The function overview and code environment on for ClubHQ

8 Mongo Atlas

“MongoDB Atlas is a multi-cloud database service by the same people that build MongoDB” [6]. A MongoDB Atlas cluster makes use of the MongoDB servers to store and manage data on the cloud without having to maintain a database server. A NoSQL database was chosen for this project as it allows for flexible document schemas and the document-based data model makes retrieving data faster than SQL databases.

In the code there is a file `/lib/mongodb.js` that exports a function `connectToDatabase()`. This file creates a `MongoClient` object and the MongoDB Node.js driver makes a connection to our database using the connection string from the ClubHQ cluster.

```
// the client and db objects are returned as a promise
let client = new MongoClient(MONGODB_URI, opts);
await client.connect();
let db = client.db(MONGODB_DB);
```

The client and db objects are created unless there are cached client and db objects. New objects are cached to improve performance for future connections.

To make a connection from the API routes to the database the `connectToDatabase()` is imported at the top of the file and is used when making a connection.

```
const { connectToDatabase } = require("../lib/mongodb");
```

```
let { db } = await connectToDatabase();
```

8.1 Collections

The ClubHQ cluster contains two collections, Students and Attendance. The Students database stores every member as a document containing their personal information and is edited through the webapp. The Attendance collection stores every date that someone logged attendance for. The documents each have the date and an array of student ids, of members that logged attendance at that days class.

```
_id: ObjectId('644d8a54aebf521fd636ff3b')
date: "2023-04-28"
▶ studentIDs: Array

_id: ObjectId('644ec60f1fd5d5b838884eb2')
date: "2023-04-30"
▼ studentIDs: Array
  0: 1

_id: ObjectId('63778cf9464a19d7ffdf0a93')
name: "Sally"
birthday: "2022-10-01"
▼ address: Object
  address1: "ATU"
  address2: "Dublin rd"
  county: "Galway"
  phone: "0879988777"
  emergencyContact: "0877778899"
  grade: "Blue"
  email: "sally@gmail.com"
  studentID: 4
  dateGraded: "2023-02-02"
```

Figure 8-1 Attendance and student documents from database

9 Next.js Webapp

Next.js is a backend framework based on React, with some additional features[7]. Next.js allows server-side rendering, which improves the performance of the web app by generating content on the server and sending it to the client. Next.js also provides API routes for fetching data from database without the need for a separate server.

9.1 API folder

9.1.1 API database routes

Next.js has an API folder that means a separate server does not need to be maintained to handle API requests [8]. This allows the server-side logic of the API endpoints to be handled by the applications hosting platform, in this case Vercel. The code has two API endpoints, students, and attendance. These endpoints handle all requests to the database using the URLs 'api/students' and 'api/attendance'. Both routes use a switch statement to check for the HTTP method of the request.

```
switch (req.method) {
  case "GET": {
    if (req.query.email) {
      return getSingleStudent(req, res);
    } else {
      return getStudents(req, res);
    }
  }
  case "POST": { return addStudent(req, res); }
  case "PUT": { return updateStudent(req, res); }
  case "DELETE": { return deleteStudent(req, res); }
}
```

9.1.2 NextAuth authentication

This webapp uses NextAuth to handle authentication. It uses the Google provider to allow users of the webapp to log in with their Gmail account. This means that for members who use the webapp on their personal devices, if they are logged in to their Google account, they will not have to log in every time they access the webapp.

The webapp code checks the database for the current signed in account in the list of students and if it is they can access their resources. If the user is not a registered member in the database, they are shown a generic 'please log in' message. Admin must always add a members email to a new profile.

9.1.3 Google Provider

A Google Cloud project was set up for ClubHQ to handle authentication requests for NextAuth. In APIs and Services – Credentials, the authorised URIs for the web app need to be added to the project. The authorised redirect URIs must also be configured with the callback URIs for the project.

The Google provider is configured with the client ID and client secret properties for the Google OAuth 2.0 API. These properties are stored as environment variables in the project.

```
providers: [
  GoogleProvider({
    clientId: process.env.GOOGLE_CLIENT_ID,
    clientSecret: process.env.GOOGLE_CLIENT_SECRET,
  }),
],
```

9.2 Pages and server-side props

The ClubHQ web app consists of four pages contained in the API folder, Add-student, Calendar-page, Profile-page, and Student-list. Most of these pages return a component from elsewhere in the code, with the exception of the Student-list page. This page uses the `getServerSideProps()` function, a Next.js feature, to return an object of students as props to the page component. This function fetches data on the server side before the page is rendered, allowing the page to be pre-populated. This means that the page can display data at a much faster time than pages that implement `useEffect()` hooks, which run after the component renders.

```
export async function getServerSideProps(ctx) {
  const { PROD_URL } = process.env;
  const response = await fetch(PROD_URL + '/api/students');
  const data = await response.json();
```

```
return {
  props: {
    students: data['message'],
  },
};
}
```

However, a limitation of `getServerSideProps()` is that the route URL for the 'students' API could not be modified to include an email variable based on the member currently logged in. This means for pages returning data based on the current session user, a `useEffect()` hook in the component itself is more efficient.

```
let emailURL = "/api/students?email=" + currentUser;
useEffect(() => {
  const fetchData = async () => {
    const response = await fetch(emailURL);
    const resdata = await response.json();
    setUser(resdata["message"]);
  };
  fetchData();
}, []);
```

9.3 Stale-While-Revalidate (SWR) research

SWR is a React hook library for client-side data fetching that allows revalidation and caching [9]. It allows for a page to be populated with data from a cache so that users can immediately see information on the page. The function then revalidates and fetches the new data and the page gets updated with the actual data.

On discussion with my project supervisor it was decided this method was over complicating a process that can be achieved with a `useEffect()` within the component and that because the components need to receive the 'session.user' before fetching the data it would be just as efficient to fetch the data without SWR

```
const { data } = useSWR('/', () => fetcher())
const fetcher = async () => {
  const response = await fetch(PROD_URL + '/api/students');
  const data = await response.json();
```

```
return data
}
```

9.4 Components

9.4.1 Calendar

A heatmap calendar is a visual representation of all a member's attendance in a year. It is easier to view overall progress in a visual way rather than a monthly view of text. Each date is clickable making it possible to view data related to that date. This project uses the react-calendar-heatmap component [10], installed using 'npm install' as a dependency.[11]

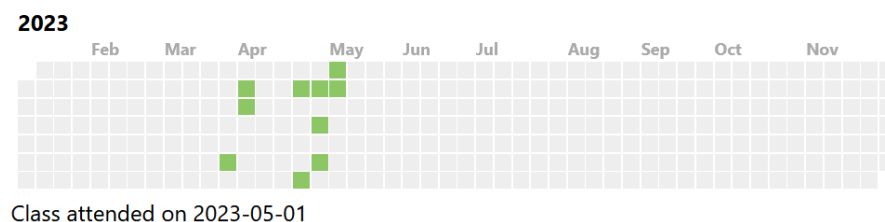


Figure 9-1 Heatmap from app

9.4.2 Profile

The profile page for each student uses the current session user email to find the current logged-in users' details from the database. A useEffect() hook is used to fetch data from the database any time the page renders.

9.4.3 Student List

This page is only visible to a designated admin user. Here, an array of all students is displayed with their grade and student id by importing a student list component and passing it 'students' from the getServerSideProps() function. The admin user can use this page to add, edit and delete students from the database.

```
return (
  <div className={classes.content}>
    <StudentL students={students} />
    <div className={classes.spacer}></div>
    <Card>
      <Link href="/add-student">Add Student</Link>
    </Card>
  </div>
);
```


10 Vercel Deployment

ClubHQ has been deployed on Vercel, a platform developed by the creators of Next.js. This means that the webapp can be accessed from a URL rather than an IP address.

The ClubHQ repository on Github is imported on Vercel and any environment variables such as Google secrets and MongoDB Atlas connection string are added when creating the Vercel deployment project. An automatic deployment begins when a pull request is made to the main branch.

For each pull request a preview build is run to check for errors. Below is an example of a preview that failed, a fix was made and committed to the same branch and the next preview build was successful. The branch can then be merged to the main branch and a production build can begin. All successes and failures are automatically emailed to the project owner.

clubhq-dp1u55eir-elainekillalea.vercel.app Production	● Ready 28s	🔗 main ↪ 96868d3 Merge pull request #18 from el...
clubhq-5648tob5n-elainekillalea.vercel.app Preview	● Ready 36s	🔗 postbuildchanges ↪ 2e84bf1 Added checks for session in pr...
clubhq-n69pirpwq-elainekillalea.vercel.app Preview	● Error 17s	🔗 postbuildchanges ↪ c71eb73 changed getServerSideProps t...

Figure 10-1 Deployments

Vercel also provides logs on the current build, to show what requests have been made from the app. Here we can see requests to the API folder, a log in through the Auth folder and a server-side request.










MAY 03 09:53:17.64	304	clubhq.ve...	 [GET] /api/students	
MAY 03 09:53:15.97	304	clubhq.ve...	 [GET] /api/attend...	Get attendance
MAY 03 09:53:15.78	304	clubhq.ve...	 [GET] /api/attend...	Get attendance
MAY 03 09:53:12.35	200	clubhq.ve...	 [GET] /api/students	
MAY 03 09:53:10.59	200	clubhq.ve...	 [GET] /_next/data/qRa0CVyJoYonhRww168	
MAY 03 09:53:08.83	304	clubhq.ve...	 [GET] /api/attend...	Get attendance
MAY 03 09:53:08.61	200	clubhq.ve...	 [GET] /api/attend...	Get attendance
MAY 03 09:53:05.48	200	clubhq.ve...	 [GET] /api/students	
MAY 03 09:53:01.61	200	clubhq.ve...	 [GET] /api/auth/session	

Figure 10-2 Production logs

11 Ethics

An ethical consideration I had to make during this project was the use of AI tools. With the sudden rise of use of ChatGPT to generate code it was a very appealing idea. However as this tool has limited knowledge of events after September 2021 it could not be relied upon to know of recent coding practices and technologies. For this reason I did not use it for generating code for the project, rather I used it for finding and explaining errors and bugs in my code as VSCode is quite vague when reporting errors.

12 Conclusion

ClubHQ has been successfully deployed on Vercel with the URL clubhq.vercel.app. When the hardware is connected to a laptop/PC, students can be enrolled or can log their attendance at a class. The logged student id is then immediately published to IoT Core and triggers the Lambda function to add it to the database. The webapp allows admin to manage their members and allows members to view their data and attendance.

As it got postponed from the plan due to blockers with hardware, I would like to continue with the project and add an OLED display to the hardware to replace the serial monitor for the user.

I also plan to implement more features to the calendar to allow members to sign in to specific classes by time.

13 Appendix

13.1 Source code

[ClubHQ Github](#) – the web app code. The main branch here is the deployed app.

[ClubHQ Hardware Github](#) – clubESP_AWS C code folder and AWS Lambda code

13.2 Frequently used websites

<https://www.w3schools.com/html/default.asp>

<https://nextjs.org/docs>

<https://next-auth.js.org/getting-started/introduction>

<https://learn.adafruit.com/adafruit-optical-fingerprint-sensor>

13.3 ChatGPT

4 examples of my use of ChatGPT throughout this project. Answers provided where code was included.

[1] ChatGPT. “Can you explain is it better to use axios or fetch for Nextjs”, ChatGPT, OpenAI, knowledge cutoff: 2021-09, 2023.

[2] ChatGPT. “can i send data from esp32 to mongodb atlas cluster using aws iot core and aws lambda”, ChatGPT, OpenAI, knowledge cutoff: 2021-09, 2023.

[3] ChatGPT. “What is causing this error 'SyntaxError: JSON.parse: unexpected character at line 1 column 1 of the JSON data' in my code? <code snippet>”, ChatGPT, OpenAI, knowledge cutoff: 2021-09, 2023.

“The error "SyntaxError: JSON.parse: unexpected character at line 1 column 1 of the JSON data" indicates that there is an issue with the format of the JSON data being received from the API endpoint. This error typically occurs when the JSON data is not properly formatted, which could be due to a syntax error, missing or extra characters, or other issues.


In your code, the error could be caused by several factors. Here are some steps you can take to troubleshoot:

1. Check the response from the API endpoint: You can use `console.log()` to output the response from the API endpoint and check if it is in the correct format. You can also try to access the endpoint URL directly from your browser to see if it returns a valid JSON response.
2. Check if the JSON data is valid: You can use a JSON validator to check if the data returned by the API endpoint is valid JSON. There are many online tools available that can help with this... Etc.”

[4] ChatGPT. “Why I am getting ‘TypeError: network error when attempting to fetch resource’ when using SWR. <code snippet>”, ChatGPT, OpenAI, knowledge cutoff: 2021-09, 2023.

“The issue might be related to the API endpoint URL you are using. Specifically, you are using the `process.env.DEV_URL` environment variable to construct the URL. You should make sure that this environment variable is defined and pointing to the correct URL. Also, make sure that the API endpoint is properly implemented and returns the expected data format. You can try accessing the API endpoint directly using a tool like `curl` or `Postman` to confirm that it's working correctly.

If the issue persists, you might want to try using the `fetch` function directly instead of the `useSWR` hook to see if that helps in debugging the issue. You can also check the network tab in your browser's developer tools to see if there are any errors or warnings related to the network request.”

- [1] "Overview | Adafruit Optical Fingerprint Sensor | Adafruit Learning System."
<https://learn.adafruit.com/adafruit-optical-fingerprint-sensor> (accessed May 01, 2023).
- [2] "adafruit/Adafruit-Fingerprint-Sensor-Library: Arduino library for interfacing to the fingerprint sensor in the Adafruit shop." <https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library> (accessed May 03, 2023).
- [3] "ESP32 NTP Client-Server: Get Date and Time (Arduino IDE) | Random Nerd Tutorials."
<https://randomnerdtutorials.com/esp32-date-time-ntp-client-server-arduino/> (accessed Apr. 30, 2023).
- [4] "knolleary/pubsubclient: A client library for the Arduino Ethernet Shield that provides support for MQTT." <https://github.com/knolleary/pubsubclient> (accessed May 03, 2023).
- [5] "bblanchon/ArduinoJson:  JSON library for Arduino and embedded C++. Simple and efficient." <https://github.com/bblanchon/ArduinoJson> (accessed May 03, 2023).
- [6] "What is MongoDB Atlas? — MongoDB Atlas." <https://www.mongodb.com/docs/atlas/> (accessed Apr. 30, 2023).
- [7] "Next.js vs React – What are the Differences?"
<https://www.freecodecamp.org/news/next-vs-react/> (accessed Apr. 30, 2023).
- [8] "API Routes: Introduction | Next.js." <https://nextjs.org/docs/api-routes/introduction> (accessed Apr. 30, 2023).
- [9] "Data Fetching: Client side | Next.js." <https://nextjs.org/docs/basic-features/data-fetching/client-side> (accessed May 03, 2023).
- [10] "kevinsqi/react-calendar-heatmap: An svg calendar heatmap inspired by github's contribution graph." <https://github.com/kevinsqi/react-calendar-heatmap#readme> (accessed Apr. 30, 2023).
- [11] "react-calendar-heatmap - npm." <https://www.npmjs.com/package/react-calendar-heatmap> (accessed Apr. 30, 2023).