National College of Ireland

BSc in Computing

Cloud Computing

2022/2023

Kevin O’Rourke

X15042782

X15042782@student.ncirl.ie

Streamlining Sports Club Management: A Next.js Application with Strapi CMS and Supabase

Technical Report

Contents

Executive Summary

Max 300 words. Summarise the key points of the report. Restate the purpose of the report, highlight the major points of the report, and describe any results, conclusions, or recommendations from the report.

This report encapsulates the development process and functionality of a comprehensive website developed for a local sports club. The website, assembled using an array of modern technologies including Next.js, Tailwind CSS, Supabase DB, Strapi CMS, and Stripe, addresses diverse needs for different user types including members, potential members and team managers.
The main purpose of the platform is to simplify the membership registration and payment process, allowing a single account holder to enrol multiple family members. The secure payment processing for memberships and other purchases, like lotto tickets, is facilitated through an integration with Stripe. The website also provides a crucial news hub, an informational about section, and a downloads section for important resources.

One notable feature of this platform is its capacity to enable non-authenticated users to purchase lotto tickets, expanding its functionality beyond the membership management. Furthermore, team managers have the ability to view members in their respective teams, promoting better team management and communication.

The aim of this report is to explain the technology stack, discuss the design and architectural decisions, and to outline how the website meets both its functional and user requirements. The report concludes that the website effectively achieves its goals, presenting an efficient, intuitive, and multipurpose platform for the sports club. Future considerations for development involve the potential of a messaging platform for team managers and administrator to message members.

1.0 .Introduction

1.1. Background

Why did you undertake this project?

As the treasurer of my local sports club, I frequently encountered challenges while managing the membership payments. I was tasked with manually tracking each member's payment status, updating records, and ensuring timely fee collection. The process was labour intensive, time consuming, and often led to inconsistencies.

In addition, the club's team managers were reliant on weekly excel spreadsheets to keep track of their team members. These spreadsheets, while functional, were not the most efficient or reliable method to manage and communicate membership status or updates.

The motivation to undertake this project stemmed from these operational difficulties and the recognition of the potential for digital transformation within the club. The aim was to develop a web-based solution that could automate the membership management process, enable online payments, provide real-time updates to team managers, and offer other useful features to the club's members and potential members.

1.2. Aims

What does the project aim to achieve?

This project aims to achieve several key goals:

- Improve Membership Management: The primary aim is to automate and digitize the membership management process. The new system allows users to join as members and make payments online, thereby minimizing manual tracking and administrative tasks.
- Facilitate Team Management: By providing team managers with the ability to view and manage their team members in real-time, the project aims to streamline the communication between managers and members, thereby improving team organization and coordination.
- Enable Non-Authenticated Transactions: In addition to member-specific features, the system also allows non-authenticated users to purchase lotto tickets. This broadens the user base and potential revenue streams for the club.
- Enhance Communication: With a dedicated news section managed through Strapi CMS, the project aims to create a centralized platform for sharing updates and important information, thereby improving communication within the club.
- Provide Resources: The platform also serves as a resource hub, providing members with access to downloadable materials and information related to the club.
- Increase Efficiency: By automating manual processes and creating a one-stop platform for members and team managers, the project ultimately aims to increase the overall operational efficiency of the club.

Overall, the project aims to leverage modern technology to enhance the functionality, user experience, and operational efficiency of the club's administrative and communication processes.

1.3. Technology

What technology will you use to achieve what you have set out to do and how will you use it?

To achieve the set objectives, the project uses a range of modern technologies, each playing a vital role in the overall functionality and efficiency of the website:

- Next.js: An open-source JavaScript framework, Next.js forms the backbone of the website's user interface. Its server-side rendering capability ensures a fast, SEO-optimized frontend, laying the groundwork for an interactive and user-friendly experience.

- Tailwind CSS: This utility-first CSS framework is used to enhance the aesthetic appeal of the website. Tailwind CSS provides low-level utility classes that help to design custom styles, thereby crafting a responsive and visually engaging user interface.
- Supabase DB: Supabase, an open-source alternative to Firebase, serves as the database and backend service for the website. It handles the creation, reading, updating, and deleting of membership data, alongside managing user authentication. This ensures a secure login functionality and efficient management of user data.
- Amazon SES: For sending authentication emails as part of the user authentication process managed by Supabase, Amazon Simple Email Service (SES) is used. This ensures the delivery of reliable, secure, and scalable email-sending capabilities.
- Strapi CMS: Hosted on Heroku, Strapi is a headless CMS used to manage and distribute content for the website's news and downloads sections. This provides a flexible platform for club administrators to update news and upload resources without needing to modify any code.
- Stripe: Stripe's payment platform handles all transactions on the website, including membership fees and lotto ticket purchases. By integrating Stripe, the website offers a secure and streamlined payment process, ensuring a seamless user experience.
- Vercel & Heroku: Vercel is the hosting platform for the Next.js frontend, offering optimized performance for Next.js applications and continuous deployment from Git repositories. Heroku, on the other hand, is used for hosting the Strapi CMS, ensuring the reliable performance and availability of the content management system.

By strategically utilizing these technologies, the project effectively delivers a comprehensive, efficient, and user-friendly platform that caters to the varied needs of the local sports club.

1.4. Structure

The document begins with an introduction that lays the foundation, presenting the challenges faced by the club treasurer and the inspiration for the project.

In the project overview, we look into the project's aims and how it improves existing processes.

The technologies used section describes the chosen technology stack, emphasizing the role of Next.js, Strapi CMS, Supabase, Stripe, and Amazon SES in the framework.

This leads into the requirements and system architecture, which presents a high level view of the flow of the applications and connection to various services.

In implementation, the focus shifts to the project's functions, and the router structure of the Next.js app. Then testing, a look at the tools and methodologies used, including unit, and end-user testing, while also highlighting performance, accessibility, and SEO metrics.

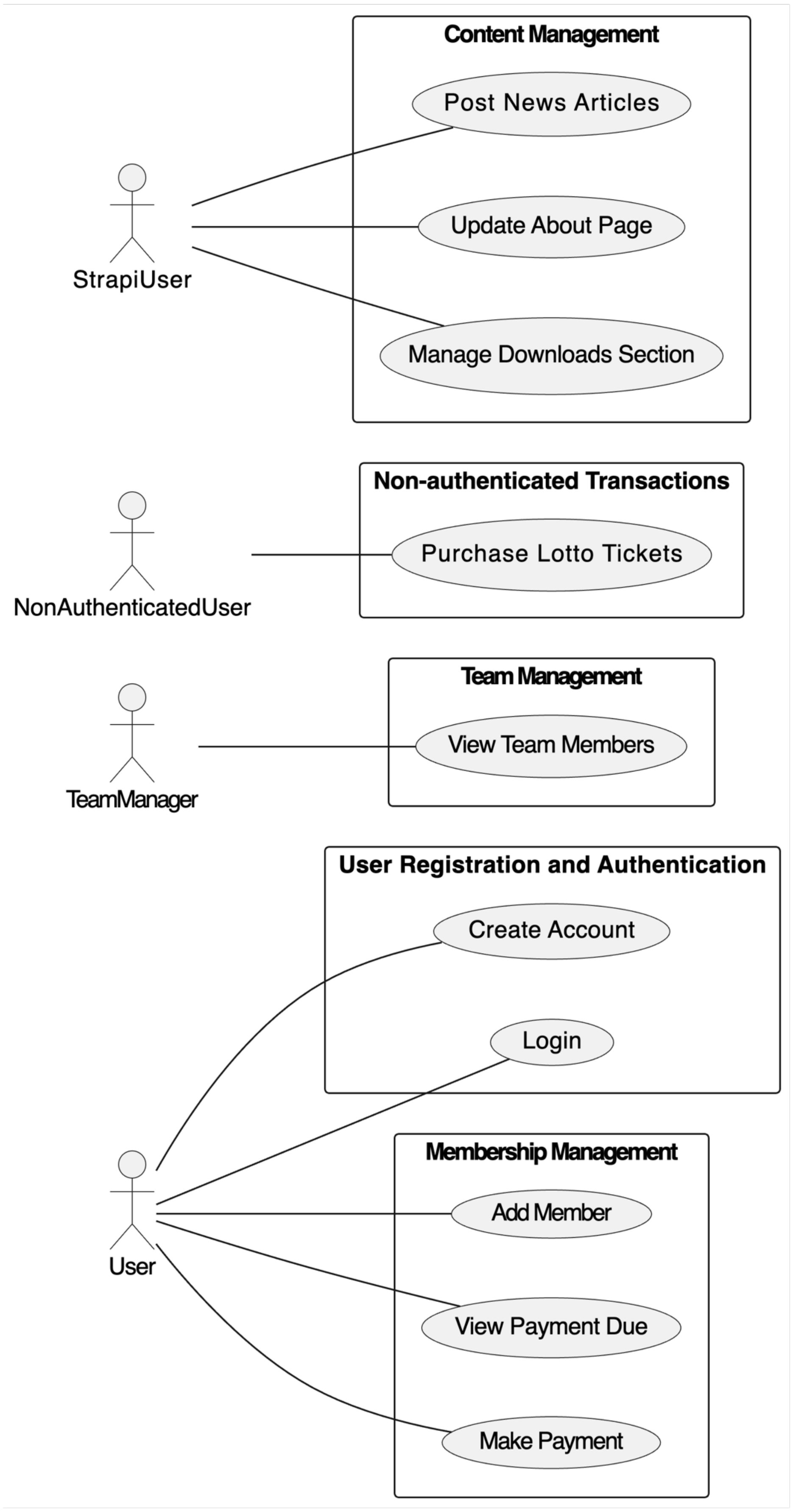
The project's advantages and disadvantages are discussed in the conclusion while future aspirations are then discussed in future directions.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

2.1.1.1. Use Case Diagram


Content Management
Post News Articles
Update About Page
Manage Downloads Section
StrapiUser
Non-authenticated Transactions
Purchase Lotto Tickets
NonAuthenticatedUser
Team Management
View Team Members
TeamManager
User Registration and Authentication
Create Account
Login
Membership Management
Add Member
View Payment Due
Make Payment
User

2.1.1.2. Requirement 1: User Registration and Authentication

2.1.1.3. Description & Priority

High Priority. The system must allow users to register for an account and authenticate their credentials for secure login.

2.1.1.4. Use Case

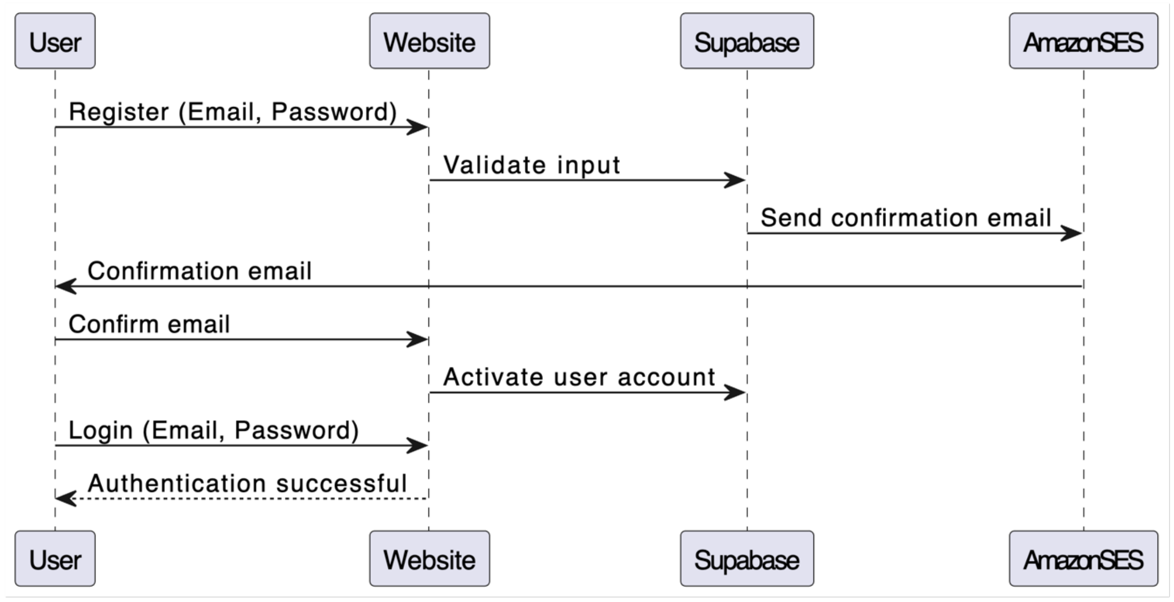
Scope

The scope of this use case is to establish a secure user registration and login process.

Description

This use case describes how users register for an account and authenticate their credentials for secure login.

Use Case Diagram



Flow Description

Precondition

The system is operational, with Supabase, Next.js, and Amazon SES correctly configured and functioning.

Activation

This use case starts when a potential User chooses to create an account.

Main flow

1. User navigates to the registration page and inputs necessary details for registration, email and password.
2. Supabase validates and stores the data, then sends a confirmation email through Amazon SES.
3. User verifies their email address via the confirmation email.
4. System acknowledges the verification and activates the user account.
5. User is able to log in using their registered email and password.

Alternate flow

N/A

Exceptional flow

If the user inputs invalid or incomplete details, the system notifies the user and prompts for correct information.

Termination

The use case ends when the user has successfully registered and is able to log in using their credentials.

Post condition

The user's account is active and ready for secure login and use of the sports club's website.

2.1.1.5. Requirement 2: Membership Management

2.1.1.6. Description & Priority

High Priority. The system must facilitate users to join the club, add members and make online payments securely.

2.1.1.7. Use Case

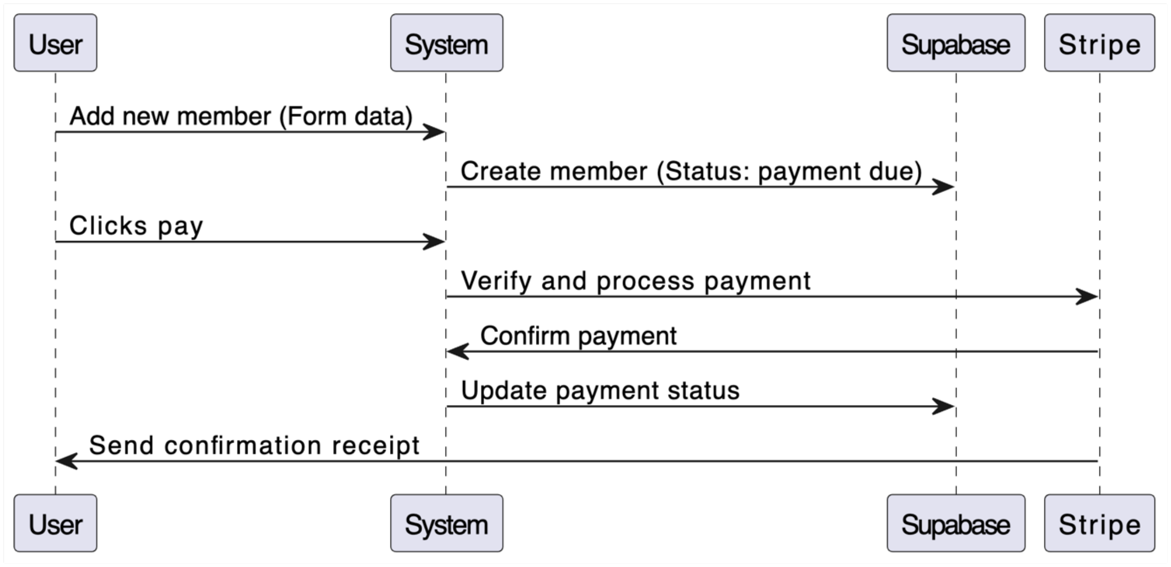
Scope

The scope of this use case is to enable the members of the sports club to manage their memberships online.

Description

This use case describes the process of joining the club, add members, and making online payments securely.

Use Case Diagram


User
System
Supabase
Stripe
Add new member (Form data)
Create member (Status: payment due)
Clicks pay
Verify and process payment
Confirm payment
Update payment status
Send confirmation receipt
User
System
Supabase
Stripe

Flow Description

Precondition

The system is online, and the user has successfully registered and logged into their account.

Activation

This use case starts when a User navigates to the membership page after login.

Main flow

1. User navigates to the registration page and inputs necessary details for registration, email and password.
2. Supabase validates and stores the data, then sends a confirmation email through Amazon SES.
3. User verifies their email address via the confirmation email.
4. System acknowledges the verification and activates the user account.
5. User is able to log in using their registered email and password.

Alternate flow

N/A

Exceptional flow

If the payment fails, the system notifies the user about the failed transaction and prompts to try again.

Termination

The use case ends when the user has successfully purchased their membership and received a confirmation receipt.

Post condition

The user's membership status is updated in the system, and the user receives a confirmation receipt in their email.

2.1.1.8. Requirement 3: Content Management

2.1.1.9. Description & Priority

High Priority. The system must allow authorized Strapi users to manage content on the website, specifically, post news articles, update the about page, and manage the downloads section.

2.1.1.10. Use Case

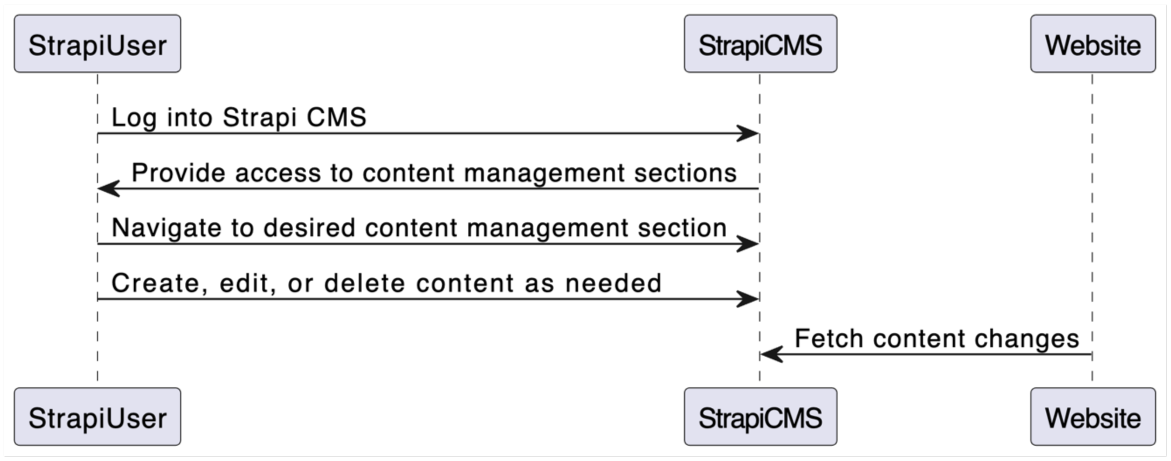
Scope

The scope of this use case is to facilitate effective management of website content, including the news articles, about page, and downloads section.

Description

This use case describes how authorized Strapi users can log in and manage content on the website.

Use Case Diagram


StrapiUser
StrapiCMS
Website
Log into Strapi CMS
Provide access to content management sections
Navigate to desired content management section
Create, edit, or delete content as needed
Fetch content changes
StrapiUser
StrapiCMS
Website

Flow Description

Precondition

The system is operational, with Strapi CMS correctly configured and functioning.

Activation

This use case starts when an authorized Strapi User chooses to manage content on the website.

Main flow

1. Strapi User logs into the Strapi CMS.
2. Strapi User navigates to the desired content management section.
3. Strapi User creates, edits, or deletes content as needed.
4. The website fetchs the content changes.

Alternate flow

N/A

Exceptional flow

If the Strapi User inputs invalid or incomplete details, the system notifies the user and prompts for correct information.

Termination

The use case ends when the Strapi User has successfully updated the desired content.

Post condition

The website's content is updated as per the Strapi User's input.

2.1.1.11. Requirement 4: Team Management

2.1.1.12. Description & Priority

High Priority. The system must enable team managers to view members of their respective teams.

2.1.1.13. Use Case

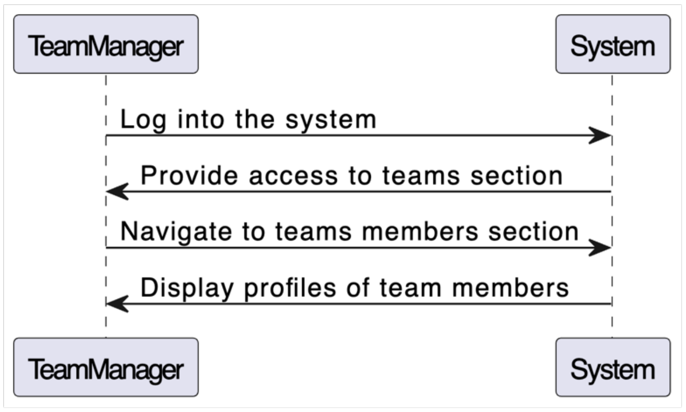
Scope

The scope of this use case is to provide team managers with access to the profiles of members in their respective teams.

Description

This use case describes how team managers can log in and view the profiles of members in their team.

Use Case Diagram


TeamManager
System
Log into the system
Provide access to teams section
Navigate to teams members section
Display profiles of team members
TeamManager
System

Flow Description

Precondition

The system is operational, and team managers have been granted the necessary permissions to view member profiles.

Activation

This use case starts when a Team Manager logs into the system to view team member profiles.

Main flow

1. Team Manager logs into the system.
2. Team Manager navigates to the teams section.
3. The system the members in the Team Manager's team.

Alternate flow

N/A

Exceptional flow

If the Team Manager does not have the permission to view the teams he must contact the admin.

Termination

The use case ends when the Team Manager has successfully viewed the desired profiles.

Post condition

The system remains in a secure state, and no unauthorized access to member profiles is allowed

2.1.1.14. Requirement 5: Lotto Transactions

2.1.1.15. Description & Priority

High Priority. The system must allow non-authenticated users to purchase lotto tickets securely

2.1.1.16. Use Case

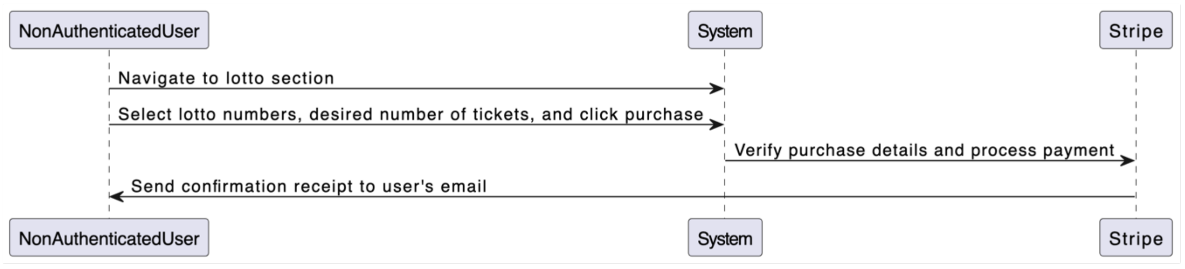
Scope

The scope of this use case is to enable non-authenticated users to purchase lotto tickets.

Description

This use case describes how non-authenticated users can interact with the website to purchase lotto tickets.

Use Case Diagram


NonAuthenticatedUser
System
Stripe
Navigate to lotto section
Select lotto numbers, desired number of tickets, and click purchase
Verify purchase details and process payment
Send confirmation receipt to user's email
NonAuthenticatedUser
System
Stripe

Flow Description

Precondition

The system is operational and capable of processing lotto ticket purchases for non-authenticated users.

Activation

This use case starts when a non-authenticated user initiates a lotto ticket purchase.

Main flow

1. Non-authenticated user navigates to the lotto section.
2. Non-authenticated user selects lotto numbers and the desired number of tickets and clicks on purchase.
3. The system verifies the purchase details and processes the payment using Stripe.
4. Stripe sends a confirmation receipt to the user's email.

Alternate flow

N/A

Exceptional flow

If the payment fails, the system notifies the user and prompts for re-trying the transaction.

Termination

The use case ends when the user successfully purchases the desired lotto tickets and receives a confirmation receipt.

Post condition

Supabase logs the ticket purchase.

2.1.2. Data Requirements

- User Data: This includes email addresses, and encrypted passwords for each user. The user's role (user, team manager, or admin) also needs to be stored.
- Membership Data: For each membership, the system needs to store details like the membership type, the associated user, the payment status, and all other personal information such as name, address and email.
- Payment Data: Each payment record should include details such as amount, date and payment id.
- News Data: The system needs to store each news post, including the post content, , the date and time of posting, and any associated imagery.
- Lotto Data: The system needs to store each lotto transaction, name, phone number, ticket numbers, the number of tickets, the date and time, and the payment confirmation.
- Team Data: For each team, the system should store the team name, the team code, id, and any associated year groups.

2.1.3. User Requirements

- Club Member Users: These users need to be able to create an account, login, purchase and manage their memberships, and view their membership status. They should also be able to add family members to their account and manage their memberships as well. Additionally, they should be able to purchase lotto tickets.
- Team Manager Users: These users need to be able to login and view the list of members in their team. They need to have access to the current status of the team members' memberships.
- Admin Users (Strapi CMS Users): These users need to be able to login, create, update and delete posts in the news section, manage the content of the about page, and manage files in the downloads section.
- Non-authenticated Users: These users need to be able to access the public pages of the website, such as the news section, about page, and downloads section. They should also be able to purchase lotto tickets.
- System Users (Stripe, Supabase, Amazon SES): These entities need to be able to interact with the website to process payments (Stripe), manage user and membership data (Supabase), and send authentication and confirmation emails (Amazon SES).

Each of these user roles have specific needs and capabilities. The well-designed user interface and robust backend services are essential to meet these user requirements.

2.1.4. Environmental Requirements

- Infrastructure Requirements: The application is hosted on a reliable and scalable infrastructure. This project uses Vercel for hosting the Next.js application, Heroku for hosting the Strapi CMS, and Supabase as the backend service.
- Software Environment: The application has a stable runtime environment to function correctly. This includes Node.js for running the Next.js and Strapi CMS applications, and PostgreSQL database (part of Supabase) for storing data.
- Network Environment: The system requires a stable and secure internet connection for accessing the hosted services, APIs, and the database.
- Security Environment: The system is secure to protect sensitive user and transaction data. This includes using HTTPS for secure connections, secure handling of user passwords,and secure transaction processing through Stripe.
- Browser Compatibility: The web application should be compatible with the latest versions of popular web browsers like Google Chrome, Firefox, Safari, and Edge to ensure a broad user reach.
- Responsive Design: The application should be compatible with desktop, tablet, mobile for seamless user experience across multiple device types.
- Email Server: Amazon Simple Email Service (SES) is correctly set up to send authentication and confirmation emails to users.
- Third-Party API Environment: The application needs to interact correctly with the Stripe API for processing payments, Supabase for user and data management, and Strapi CMS for content management.

2.1.5. Usability Requirements

- Ease of Use: The website should have an intuitive user interface that is easy to navigate. New users should be able to learn how to use the system quickly, with experienced users being able to accomplish tasks efficiently. For example, experienced users should be able to purchase a membership within five minutes.

- Consistency: The system should maintain consistency in the layout, design, and terminology across all pages. This will make the system predictable and easier to understand for users.
- Accessibility: The website should comply with accessibility guidelines to cater to users with different abilities. This includes colour contrast, text size, keyboard navigation, and screen reader compatibility.
- Responsiveness: The system should provide timely and appropriate feedback in response to user actions. This includes loading indicators during long operations and clear confirmation messages after successful actions.

2.2. Design & Architecture

The architecture of the Sports Club Membership Management System is rooted in a modern, decoupled, and cloud-based approach, with distinct services assigned to specific components of the application. This structure not only allows for effective scaling but also ensures each service is robustly capable of performing its dedicated role.

Frontend - Next.js: Next.js, hosted on Vercel's cloud platform, was chosen for its scalability, performance, and versatility. Its server-side rendering and static site generation capabilities significantly bolster the website's performance and enhance the user experience. This cloud-ready framework provides the necessary scalability to accommodate increasing traffic and content.

Backend - Supabase and Strapi CMS: Supabase and Strapi CMS, both cloud-based systems, were chosen for their comprehensive features and the scalability they offer. Supabase, an open-source alternative to Firebase, provides real-time databases, authentication, and storage. Strapi CMS is a headless solution that permits content updates for the news, about, and downloads sections of the website without restricting output to a particular format, offering room for future adaptability. Both these systems can be scaled up or down based on the demand, a key advantage of cloud-based technologies.

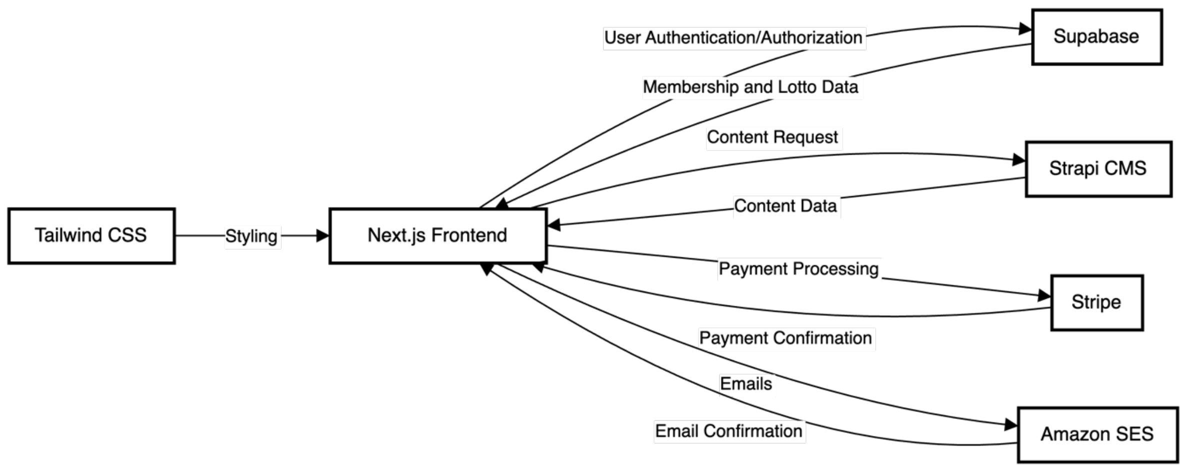
Payment Processing - Stripe: Stripe, a cloud-based online payment processing platform, was selected for its robust security and ease of implementation. It can handle a large volume of transactions, making it a scalable choice for the growing needs of the sports club.

Email Service - Amazon SES: Amazon's Simple Email Service (SES) is used for email communications. This cloud-based service is known for its scalability, reliability, and cost-effectiveness, making it an optimal choice for a growing organization.

Styling - Tailwind CSS: Tailwind CSS, though not a cloud-based technology, plays a crucial role in the UI of the system. It offers superior customization and control, encouraging design consistency and speeding up the development process.

The main algorithms used in this system revolve around CRUD (Create, Read, Update, Delete) operations on data. These operations are carried out in response to user interactions on the frontend, which makes calls to the respective APIs, processes the data and updates the user interface accordingly.

Overall, the system's cloud-based architecture allows for enhanced scalability and maintainability. The modular design provides the flexibility to interchange individual components if necessary, ensuring the Sports Club Membership Management System is resilient and capable of accommodating growth and change.



2.3. Implementation

The Next.js App Router framework uses the concept of a folder directory. Page.js file inside the folder renders the content of that route. Layout.js can be used to wrap the all the sub pages in the folder. Eg. Protected routes for unauthenticated users.

Next.js App Router Structure:

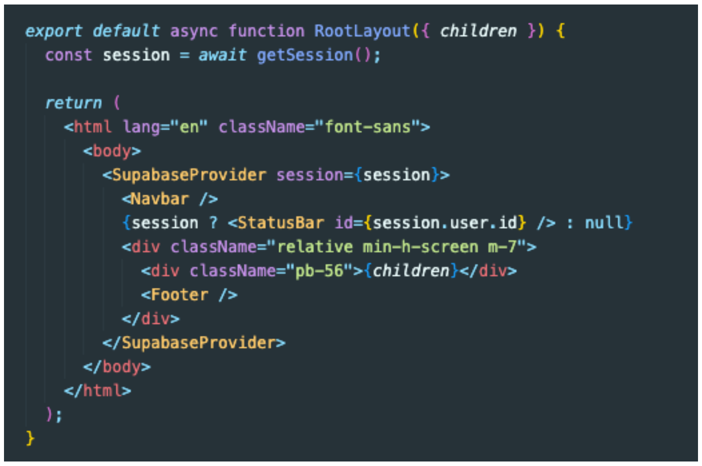
Public Routes:

These routes are accessible to everyone, including non-authenticated users.

/:

The root directory, the homepage where the Supabase provider, Navbar component and Footer component wrap the site.

In the code below you can also see a check for a current session before rendering the StatusBar component.

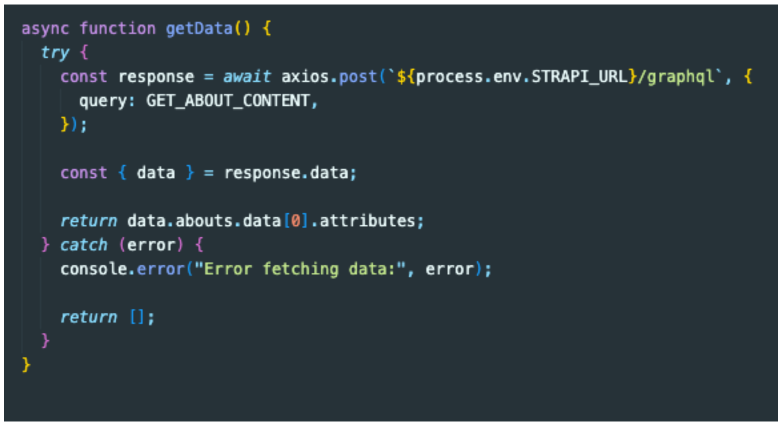


```
export default async function RootLayout({ children }) {
  const session = await getSession();

  return (
    <html lang="en" className="font-sans">
      <body>
        <SupabaseProvider session={session}>
          <Navbar />
          {session ? <StatusBar id={session.user.id} /> : null}
          <div className="relative min-h-screen m-7">
            <div className="pb-56">{children}</div>
            <Footer />
          </div>
        </SupabaseProvider>
      </body>
    </html>
  );
}
```

/about:

The About page, providing information about the sports club.

In the code below you will see the function within the About page to query the StrapiCMS for About content

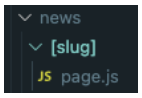


```
async function getData() {
  try {
    const response = await axios.post(`${process.env.STRAPI_URL}/graphql`, {
      query: GET_ABOUT_CONTENT,
    });

    const { data } = response.data;

    return data.abouts.data[0].attributes;
  } catch (error) {
    console.error("Error fetching data:", error);

    return [];
  }
}
```

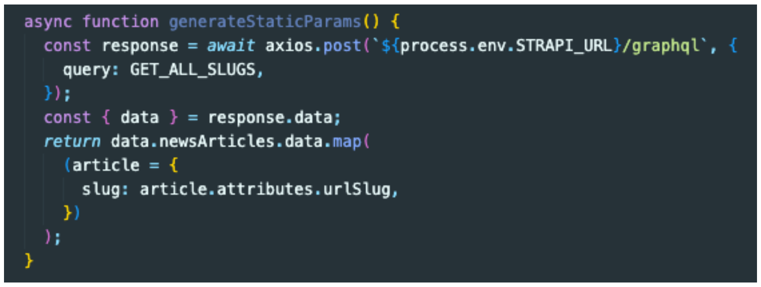
/news:

Displays news articles, similar to About, however in this folder, introduced is the [slug] feature of the Next JS App Router. The below screenshots show the folder structure and also the fetch of the url slugs for each article in the generateStaticParams() function.

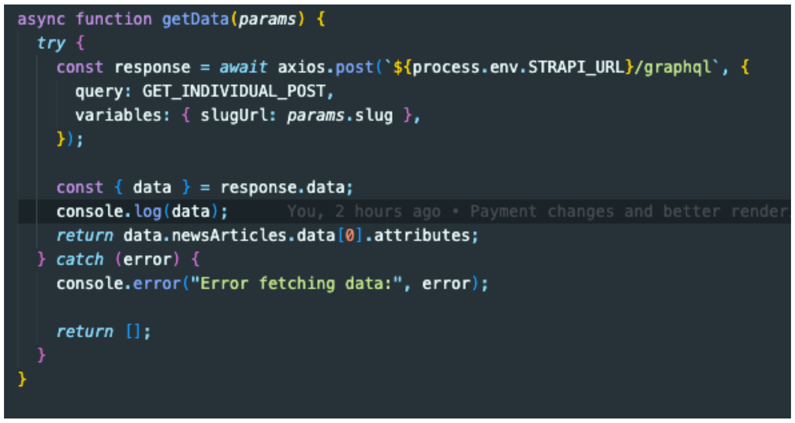
On the render of each "slug" the params slug is used to fetch the individual article details.



```
news
  [slug]
    page.js
```



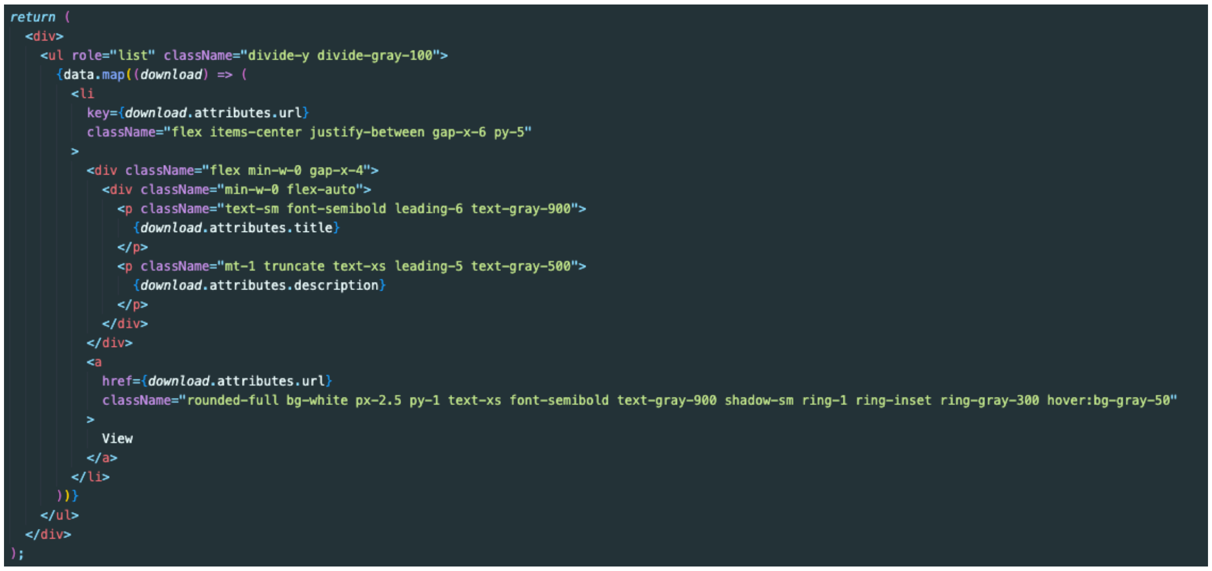
```
async function generateStaticParams() {
  const response = await axios.post(`${process.env.STRAPI_URL}/graphql`, {
    query: GET_ALL_SLUGS,
  });
  const { data } = response.data;
  return data.newsArticles.data.map(
    (article = {
      slug: article.attributes.urlSlug,
    })
  );
}
```



```
async function getData(params) {
  try {
    const response = await axios.post(`${process.env.STRAPI_URL}/graphql`, {
      query: GET_INDIVIDUAL_POST,
      variables: { slugUrl: params.slug },
    });

    const { data } = response.data;
    console.log(data);       You, 2 hours ago • Payment changes and better render
    return data.newsArticles.data[0].attributes;
  } catch (error) {
    console.error("Error fetching data:", error);

    return [];
  }
}
```

/downloads:

A section where users can access downloadable resources. This section is similar to about where data is fetched using axios from StrapiCMS. Example of mapping through data and return JSX array

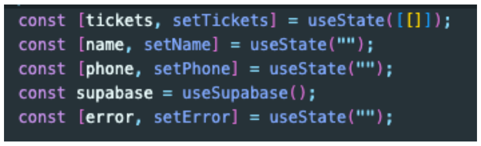


```
return (
  <div>
    <ul role="list" className="divide-y divide-gray-100">
      {data.map((download) => (
        <li
          key={download.attributes.url}
          className="flex items-center justify-between gap-x-6 py-5"
        >
          <div className="flex min-w-0 gap-x-4">
            <div className="min-w-0 flex-auto">
              <p className="text-sm font-semibold leading-6 text-gray-900">
                {download.attributes.title}
              </p>
              <p className="mt-1 truncate text-xs leading-5 text-gray-500">
                {download.attributes.description}
              </p>
            </div>
          </div>
          <a
            href={download.attributes.url}
            className="rounded-full bg-white px-2.5 py-1 text-xs font-semibold text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300 hover:bg-gray-50"
          >
            View
          </a>
        </li>
      ))}
    </ul>
  </div>
);
```

/lotto:

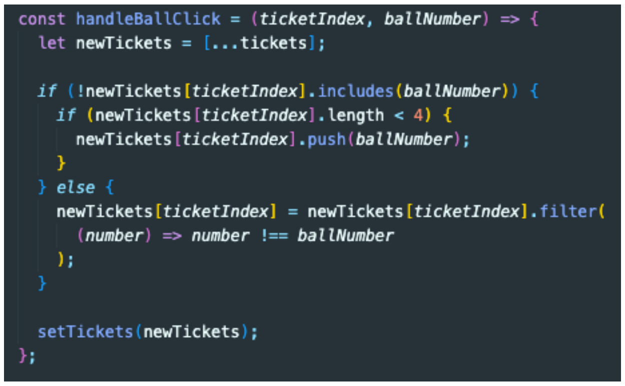
Allows non-authenticated users to purchase lotto tickets. This component has complex logic to achieve the desired outcome. Given the interactive nature of the component client side code is used.

Client side react functionality useState() is used to manage the state of tickets, name, phone and errors.



```
const [tickets, setTickets] = useState([[]]);
const [name, setName] = useState("");
const [phone, setPhone] = useState("");
const supabase = useSupabase();
const [error, setError] = useState("");
```

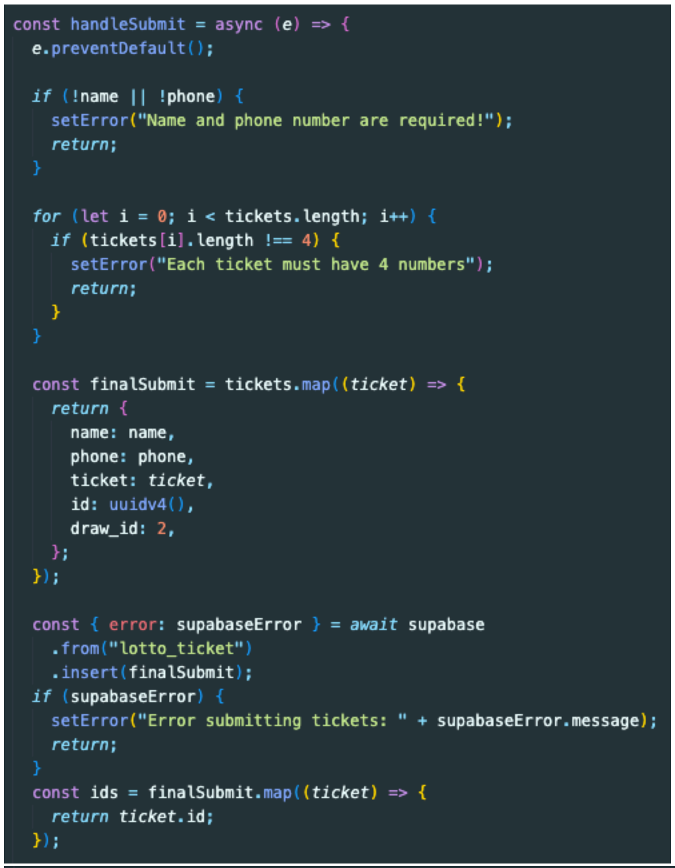
The below code handles the ball click. Only add the ball to the ticket if it is not already present and there is less that 4 numbers selected. If already selected, remove from ticket using the filter function. Then set tickets array with new value.



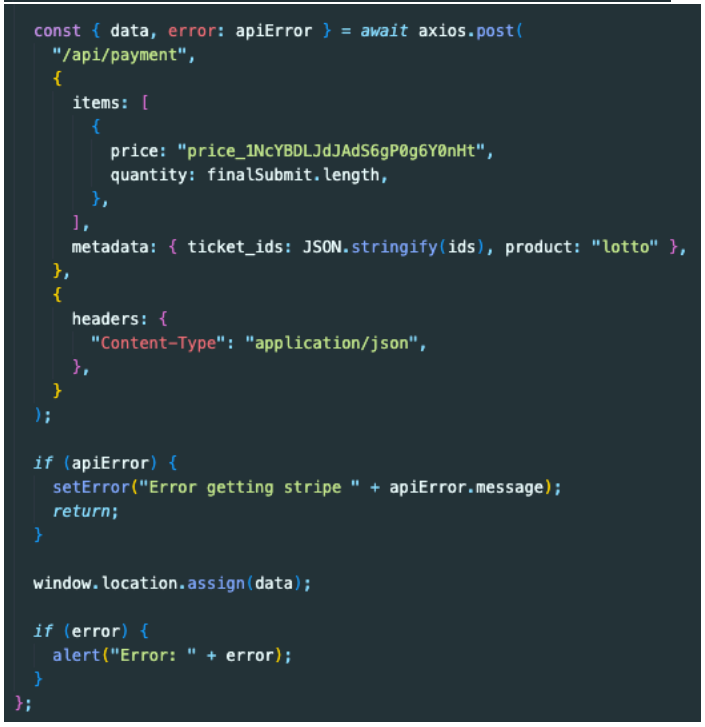
```
const handleBallClick = (ticketIndex, ballNumber) => {
  let newTickets = [...tickets];

  if (!newTickets[ticketIndex].includes(ballNumber)) {
    if (newTickets[ticketIndex].length < 4) {
      newTickets[ticketIndex].push(ballNumber);
    }
  } else {
    newTickets[ticketIndex] = newTickets[ticketIndex].filter(
      (number) => number !== ballNumber
    );
  }

  setTickets(newTickets);
};
```

The handle submit function carries out the following tasks.

- Checks for errors
- Prepares data for insert to Supabase.
- Inserts data to Supabase
- Calls the /api/payment route which handles the Stripe checkout session
- Then navigates to the Stripe Checkout if there are no errors.



```
const handleSubmit = async (e) => {
  e.preventDefault();

  if (!name || !phone) {
    setError("Name and phone number are required!");
    return;
  }

  for (let i = 0; i < tickets.length; i++) {
    if (tickets[i].length !== 4) {
      setError("Each ticket must have 4 numbers");
      return;
    }
  }

  const finalSubmit = tickets.map((ticket) => {
    return {
      name: name,
      phone: phone,
      ticket: ticket,
      id: uuidv4(),
      draw_id: 2,
    };
  });

  const { error: supabaseError } = await supabase
    .from("lotto_ticket")
    .insert(finalSubmit);
  if (supabaseError) {
    setError("Error submitting tickets: " + supabaseError.message);
    return;
  }
  const ids = finalSubmit.map((ticket) => {
    return ticket.id;
  });
```

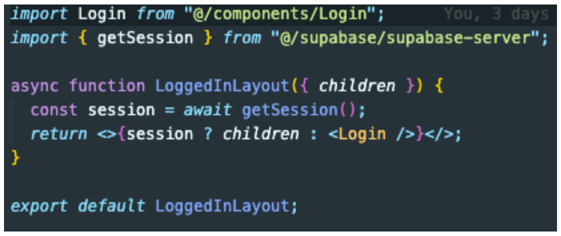


```
  const { data, error: apiError } = await axios.post(
    "/api/payment",
    {
      items: [
        {
          price: "price_1NcYBDLJdJAdS6gP0g6Y0nHt",
          quantity: finalSubmit.length,
        },
      ],
      metadata: { ticket_ids: JSON.stringify(ids), product: "lotto" },
    },
    {
      headers: {
        "Content-Type": "application/json",
      },
    }
  );

  if (apiError) {
    setError("Error getting stripe " + apiError.message);
    return;
  }

  window.location.assign(data);

  if (error) {
    alert("Error: " + error);
  }
};
```

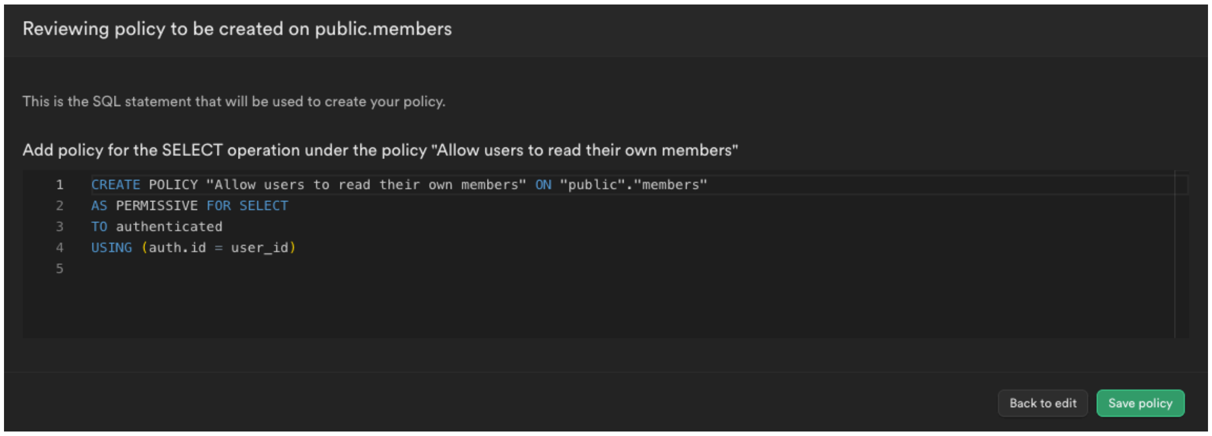
Authenticated Routes:

These routes require user authentication.

Firstly in the layout.js file in the members directory, there is a check for a Supabase authenticated session. All the Supabase tables are protected by Row Level Security policies allowing the Supabase admin to define who has permissions to take what actions on the data. See examples below.



```
import Login from "@/components/Login";        You, 3 days
import { getSession } from "@/supabase/supabase-server";

async function LoggedInLayout({ children }) {
  const session = await getSession();
  return <>{session ? children : <Login />}</>;
}

export default LoggedInLayout;
```



Reviewing policy to be created on public.members

This is the SQL statement that will be used to create your policy.

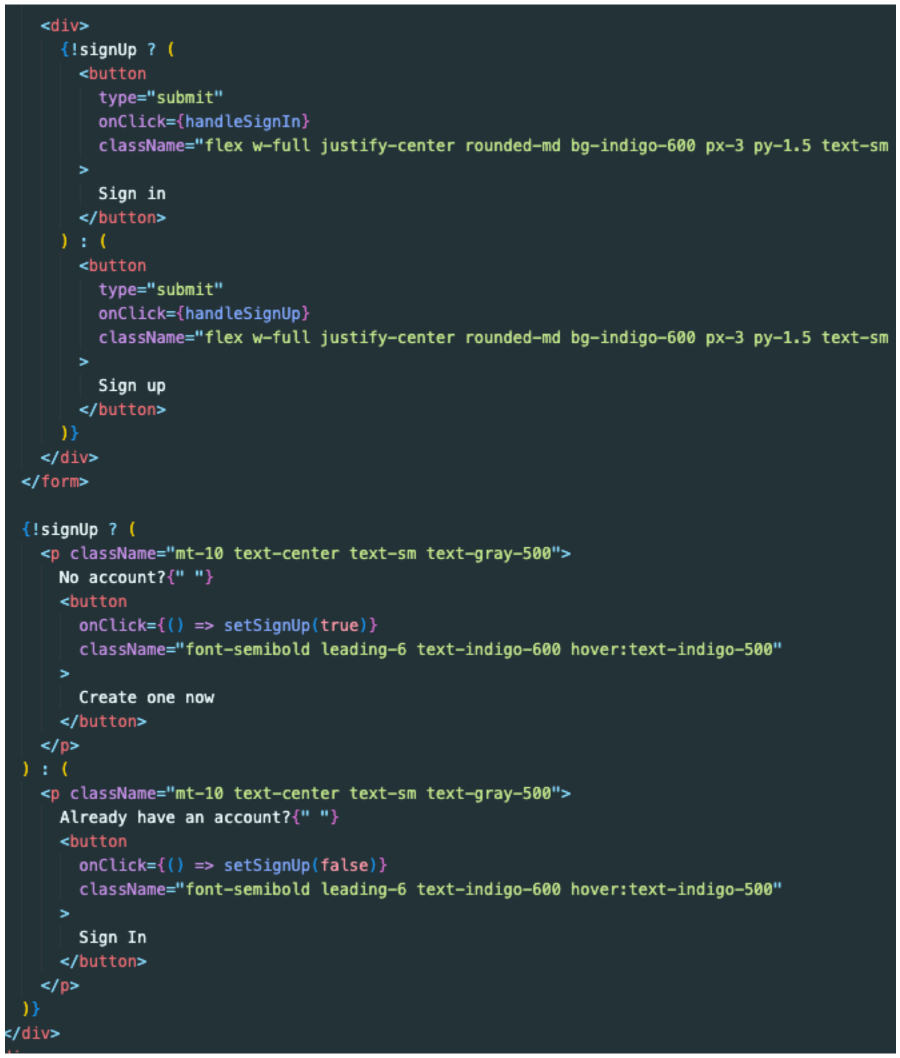
Add policy for the SELECT operation under the policy "Allow users to read their own members"

```
CREATE POLICY "Allow users to read their own members" ON "public"."members"
AS PERMISSIVE FOR SELECT
TO authenticated
USING (auth.id = user_id)
```

/members:

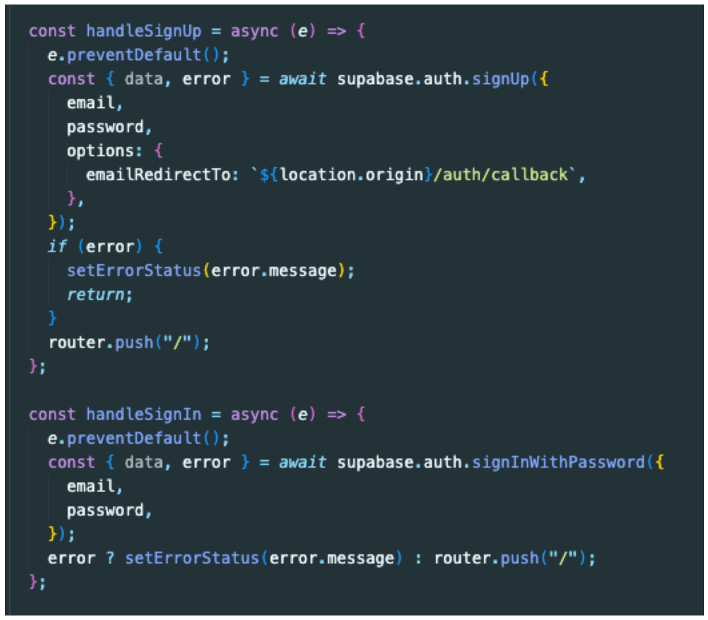
After a club member logs in, they are directed here to view their membership status, add members and purchase memberships

The Login component is loaded here. Another interactive client component. It uses state to switch between the Sign In and Create Account flow. In the screenshot below the signUp state is used to conditionally render either sign in or create account content.



```
    <div>
      {!signUp ? (
        <button
          type="submit"
          onClick={handleSignIn}
          className="flex w-full justify-center rounded-md bg-indigo-600 px-3 py-1.5 text-sm
        >
          Sign in
        </button>
      ) : (
        <button
          type="submit"
          onClick={handleSignUp}
          className="flex w-full justify-center rounded-md bg-indigo-600 px-3 py-1.5 text-sm
        >
          Sign up
        </button>
      )}
    </div>
  </form>

  {!signUp ? (
    <p className="mt-10 text-center text-sm text-gray-500">
      No account?{" "}
      <button
        onClick={() => setSignUp(true)}
        className="font-semibold leading-6 text-indigo-600 hover:text-indigo-500"
      >
        Create one now
      </button>
    </p>
  ) : (
    <p className="mt-10 text-center text-sm text-gray-500">
      Already have an account?{" "}
      <button
        onClick={() => setSignUp(false)}
        className="font-semibold leading-6 text-indigo-600 hover:text-indigo-500"
      >
        Sign In
      </button>
    </p>
  )}
</div>
```

Supabase functions used to handle sign in and sign up flows.

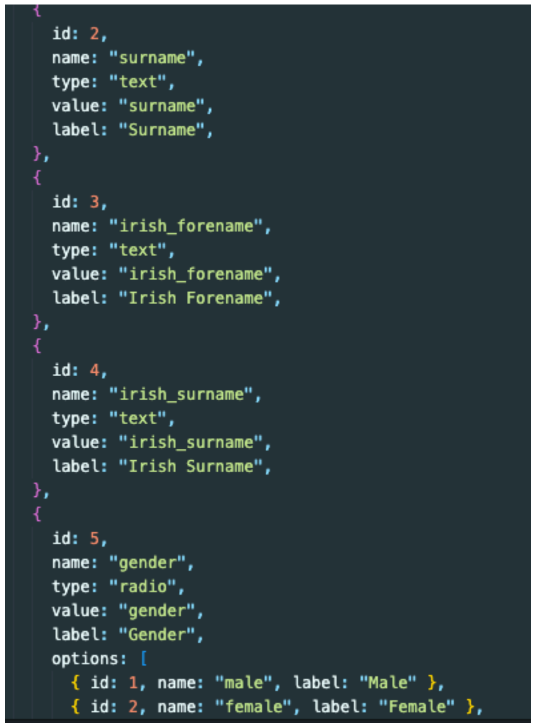


```
const handleSignUp = async (e) => {
  e.preventDefault();
  const { data, error } = await supabase.auth.signUp({
    email,
    password,
    options: {
      emailRedirectTo: `${location.origin}/auth/callback`,
    },
  });
  if (error) {
    setErrorStatus(error.message);
    return;
  }
  router.push("/");
};

const handleSignIn = async (e) => {
  e.preventDefault();
  const { data, error } = await supabase.auth.signInWithPassword({
    email,
    password,
  });
  error ? setErrorStatus(error.message) : router.push("/");
};
```

/members/register:

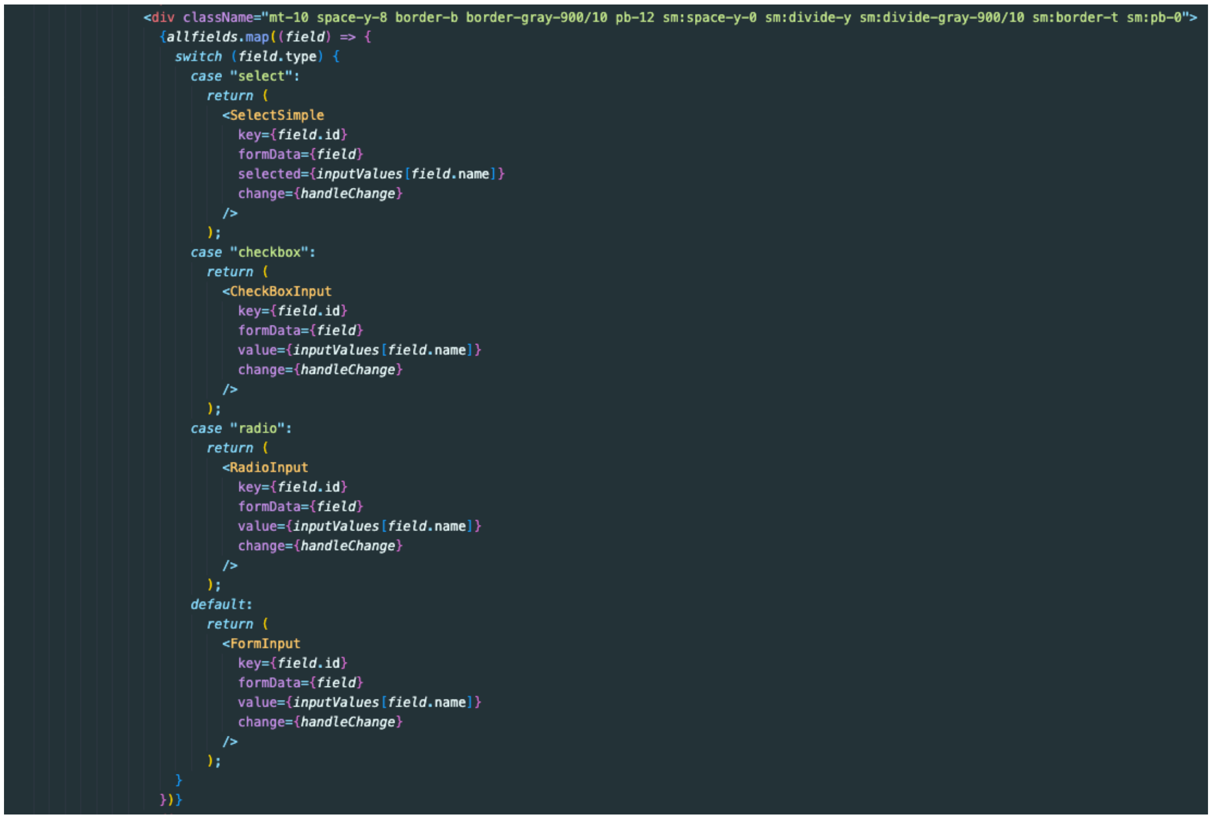
Where users can add members to their profile. A form data JSON object was used to populate the inputs for a registration form dynamically. This is the beauty of react and client side code.

Sample definition



```
    id: 2,
    name: "surname",
    type: "text",
    value: "surname",
    label: "Surname",
  },
  {
    id: 3,
    name: "irish_forename",
    type: "text",
    value: "irish_forename",
    label: "Irish Forename",
  },
  {
    id: 4,
    name: "irish_surname",
    type: "text",
    value: "irish_surname",
    label: "Irish Surname",
  },
  {
    id: 5,
    name: "gender",
    type: "radio",
    value: "gender",
    label: "Gender",
    options: [
      { id: 1, name: "male", label: "Male" },
      { id: 2, name: "female", label: "Female" },
```

Form component to that checks field type and passes the data(props) to the relevant field component. 18 fields rendered in a few lines of code.



```
<div className="mt-10 space-y-8 border-b border-gray-900/10 pb-12 sm:space-y-0 sm:divide-y sm:divide-gray-900/10 sm:border-t sm:pb-0">
  {allfields.map((field) => {
    switch (field.type) {
      case "select":
        return (
          <SelectSimple
            key={field.id}
            formData={field}
            selected={inputValues[field.name]}
            change={handleChange}
          />
        );
      case "checkbox":
        return (
          <CheckBoxInput
            key={field.id}
            formData={field}
            value={inputValues[field.name]}
            change={handleChange}
          />
        );
      case "radio":
        return (
          <RadioInput
            key={field.id}
            formData={field}
            value={inputValues[field.name]}
            change={handleChange}
          />
        );
      default:
        return (
          <FormInput
            key={field.id}
            formData={field}
            value={inputValues[field.name]}
            change={handleChange}
          />
        );
    }
  })}
```

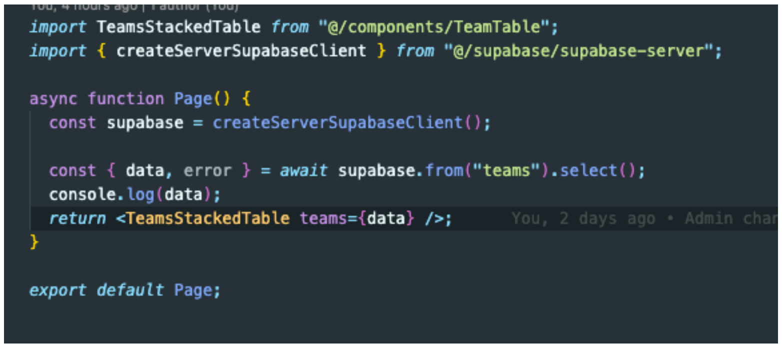
Team Manager Routes:

Accessible by team managers to view team memberships.

/team-manager:

Lists all the teams available and provides a link to their slug url.

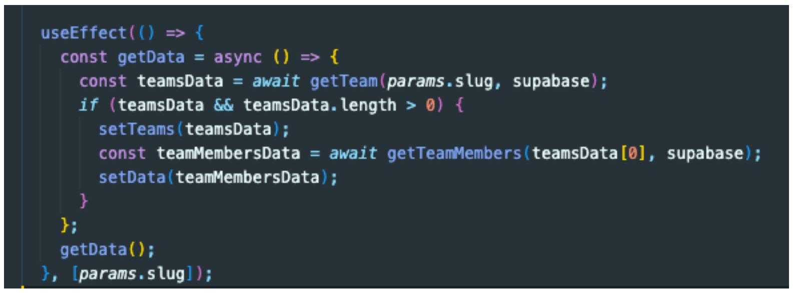
Get all teams from Supabase and populate the Teams table



```
You, 4 hours ago | 1 author (You)
import TeamsStackedTable from "@/components/TeamTable";
import { createServerSupabaseClient } from "@/supabase/supabase-server";

async function Page() {
  const supabase = createServerSupabaseClient();

  const { data, error } = await supabase.from("teams").select();
  console.log(data);
  return <TeamsStackedTable teams={data} />;      You, 2 days ago • Admin char
}

export default Page;
```

/team-manager/[slug]

Dynamic route to populate a members table with team members. The react hook useEffect is use to fetch the team data and then the members of that team from Supabase.

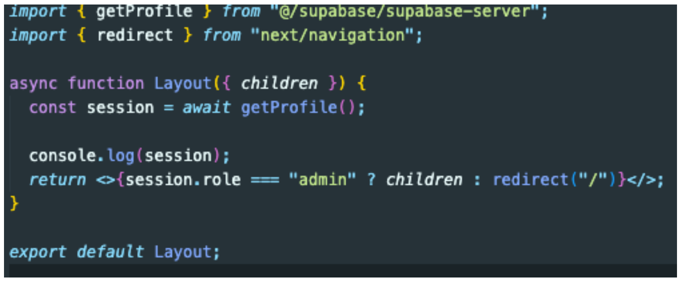


```
  useEffect(() => {
    const getData = async () => {
      const teamsData = await getTeam(params.slug, supabase);
      if (teamsData && teamsData.length > 0) {
        setTeams(teamsData);
        const teamMembersData = await getTeamMembers(teamsData[0], supabase);
        setData(teamMembersData);
      }
    };
    getData();
  }, [params.slug]);
```

Admin Routes:

These routes are specific for the Supabase admin role.

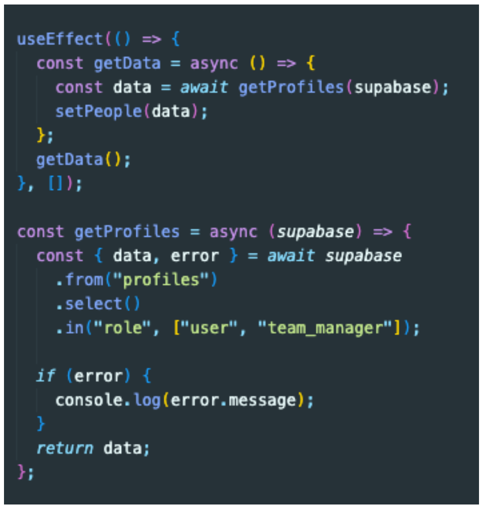
Here is the check on the layout.js page for the session.role === admin, if it’s not true. Redirect to homepage



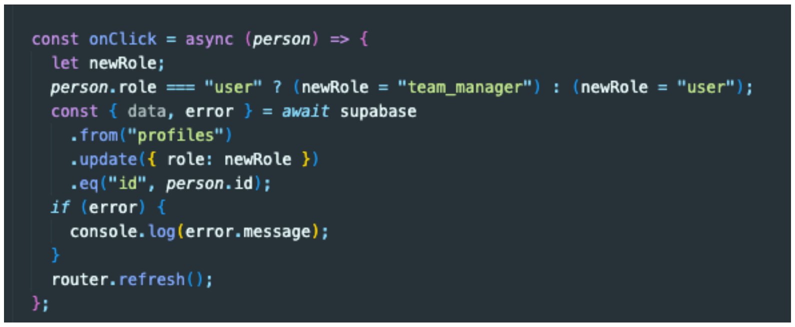
```
import { getProfile } from "@/supabase/supabase-server";
import { redirect } from "next/navigation";

async function Layout({ children }) {
  const session = await getProfile();

  console.log(session);
  return <>{session.role === "admin" ? children : redirect("/")}</>;
}

export default Layout;
```

<u>/members/admin:</u> Where the admin can change the role of users and team managers.

useEffect() used again to preload the profiles and then the onClick function using the Supabase JS library to update the role of the selected user.



```
useEffect(() => {
  const getData = async () => {
    const data = await getProfiles(supabase);
    setPeople(data);
  };
  getData();
}, []);

const getProfiles = async (supabase) => {
  const { data, error } = await supabase
    .from("profiles")
    .select()
    .in("role", ["user", "team_manager"]);

  if (error) {
    console.log(error.message);
  }
  return data;
};
```



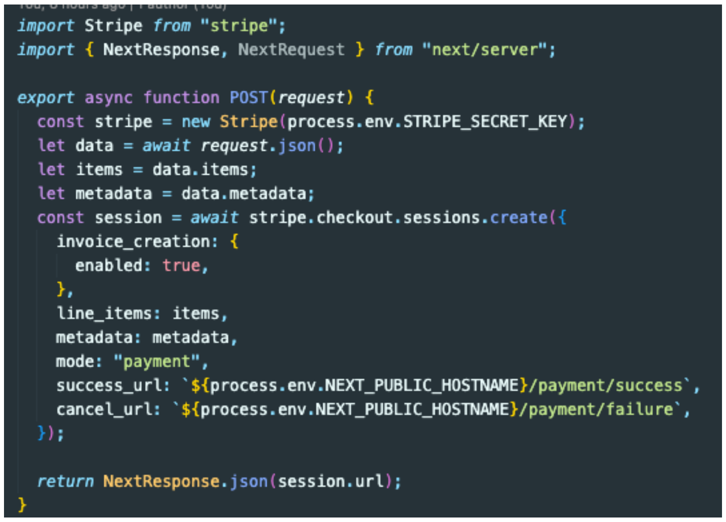
```
const onClick = async (person) => {
  let newRole;
  person.role === "user" ? (newRole = "team_manager") : (newRole = "user");
  const { data, error } = await supabase
    .from("profiles")
    .update({ role: newRole })
    .eq("id", person.id);
  if (error) {
    console.log(error.message);
  }
  router.refresh();
};
```

API Routes:

Route.js means that this code can only run on the server and is not accessible on the frontend. Any secrets keys can be used here.

<u>/api/payment/route.js.</u>

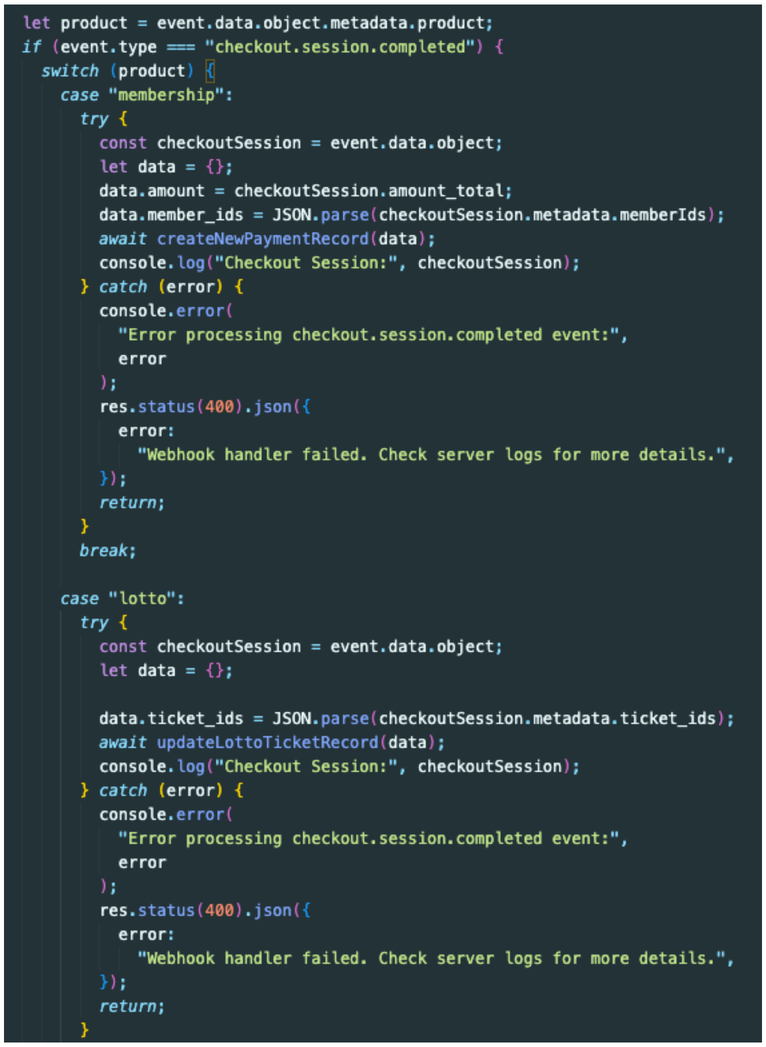
Payment route used to create Stripe checkout object.



```javascript
import Stripe from "stripe";
import { NextResponse, NextRequest } from "next/server";

export async function POST(request) {
  const stripe = new Stripe(process.env.STRIPE_SECRET_KEY);
  let data = await request.json();
  let items = data.items;
  let metadata = data.metadata;
  const session = await stripe.checkout.sessions.create({
    invoice_creation: {
      enabled: true,
    },
    line_items: items,
    metadata: metadata,
    mode: "payment",
    success_url: `${process.env.NEXT_PUBLIC_HOSTNAME}/payment/success`,
    cancel_url: `${process.env.NEXT_PUBLIC_HOSTNAME}/payment/failure`,
  });

  return NextResponse.json(session.url);
}
```

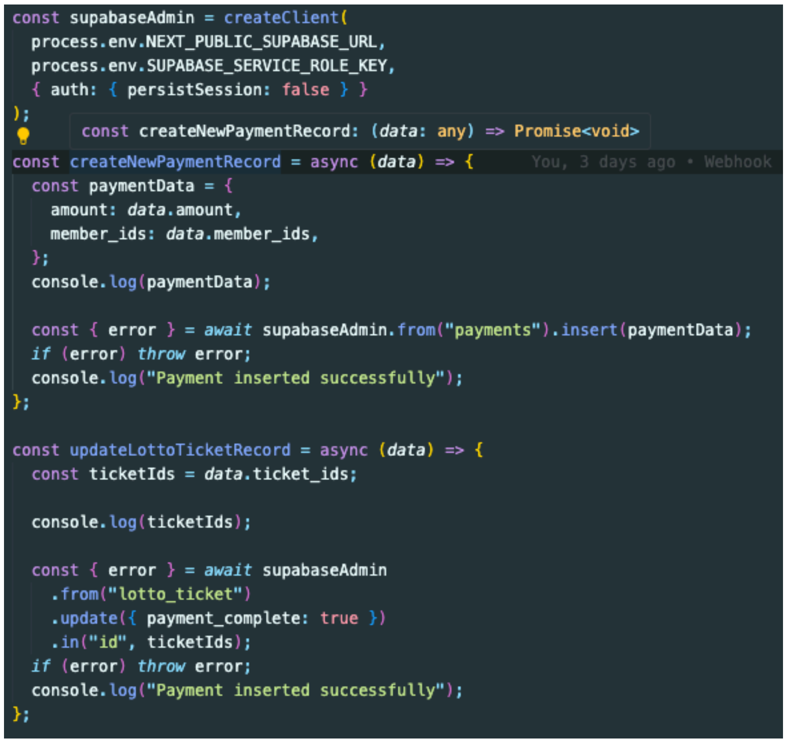
<u>/api/webhooks/route.js</u>

A webhook endpoint for Stripe to send payment confirmation back to the NextJS server and handle that data. Here we insert and update payment information in Supabase. Webhook code below checks for the product type metadata defined by the Stripe Checkout event and actions it based on that.



```
let product = event.data.object.metadata.product;
if (event.type === "checkout.session.completed") {
  switch (product) {
    case "membership":
      try {
        const checkoutSession = event.data.object;
        let data = {};
        data.amount = checkoutSession.amount_total;
        data.member_ids = JSON.parse(checkoutSession.metadata.memberIds);
        await createNewPaymentRecord(data);
        console.log("Checkout Session:", checkoutSession);
      } catch (error) {
        console.error(
          "Error processing checkout.session.completed event:",
          error
        );
        res.status(400).json({
          error:
            "Webhook handler failed. Check server logs for more details.",
        });
        return;
      }
      break;

    case "lotto":
      try {
        const checkoutSession = event.data.object;
        let data = {};

        data.ticket_ids = JSON.parse(checkoutSession.metadata.ticket_ids);
        await updateLottoTicketRecord(data);
        console.log("Checkout Session:", checkoutSession);
      } catch (error) {
        console.error(
          "Error processing checkout.session.completed event:",
          error
        );
        res.status(400).json({
          error:
            "Webhook handler failed. Check server logs for more details.",
        });
        return;
      }
```

The Supabase Admin.js functions that are called by the webhooks functions to update Supabase using admin service key credentials on the server.

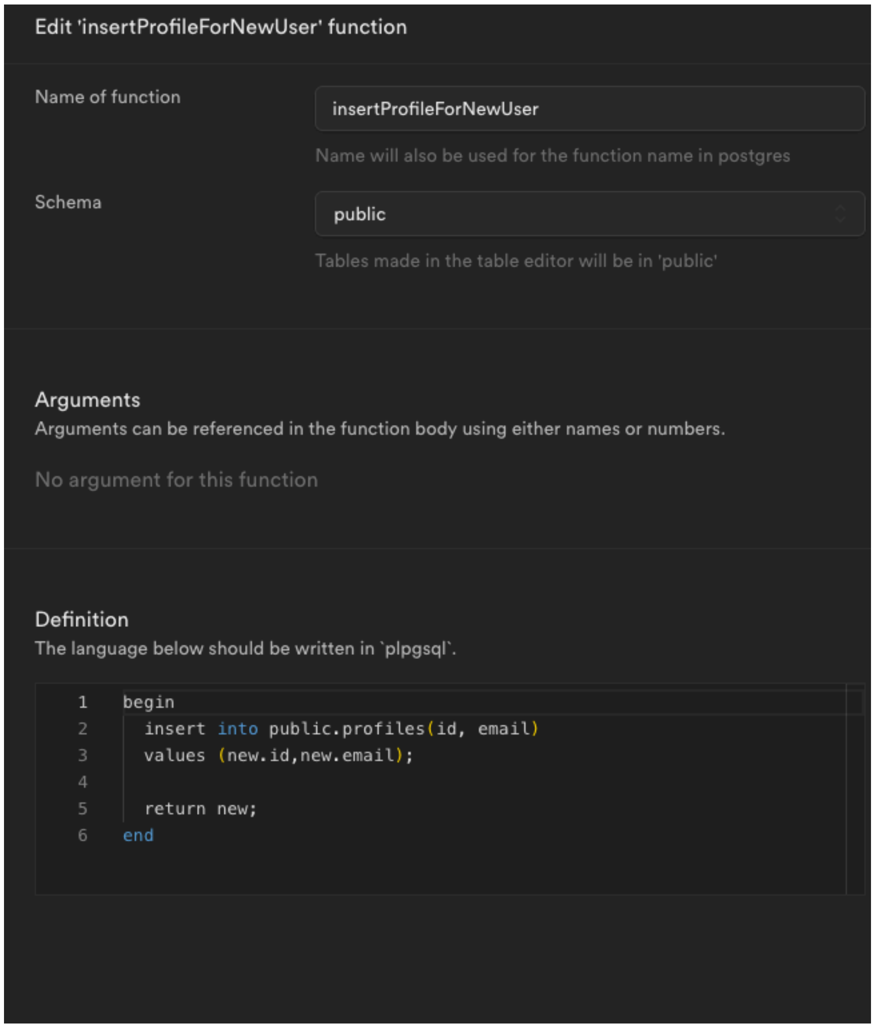


```
const supabaseAdmin = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL,
  process.env.SUPABASE_SERVICE_ROLE_KEY,
  { auth: { persistSession: false } }
);
       const createNewPaymentRecord: (data: any) => Promise<void>
const createNewPaymentRecord = async (data) => {        You, 3 days ago • Webhook
  const paymentData = {
    amount: data.amount,
    member_ids: data.member_ids,
  };
  console.log(paymentData);

  const { error } = await supabaseAdmin.from("payments").insert(paymentData);
  if (error) throw error;
  console.log("Payment inserted successfully");
};

const updateLottoTicketRecord = async (data) => {
  const ticketIds = data.ticket_ids;

  console.log(ticketIds);

  const { error } = await supabaseAdmin
    .from("lotto_ticket")
    .update({ payment_complete: true })
    .in("id", ticketIds);
  if (error) throw error;
  console.log("Payment inserted successfully");
};
```

Supabase Trigger Functions

Supabase trigger functions have been utilised to insert and update data in Supabase tables based on other actions.

Below shows a function that creates a record in the profile table when a new user account is created



Edit 'insertProfileForNewUser' function

Name of function: insertProfileForNewUser

Name will also be used for the function name in postgres

Schema: public

Tables made in the table editor will be in 'public'

Arguments

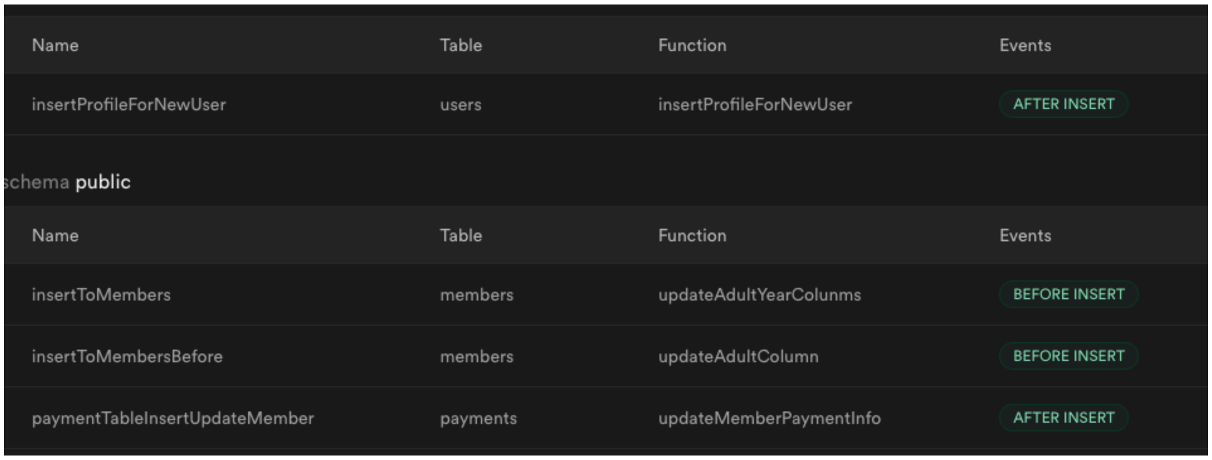
Arguments can be referenced in the function body using either names or numbers.

No argument for this function

Definition

The language below should be written in `plpgsql`.

```
begin
  insert into public.profiles(id, email)
  values (new.id,new.email);

  return new;
end
```

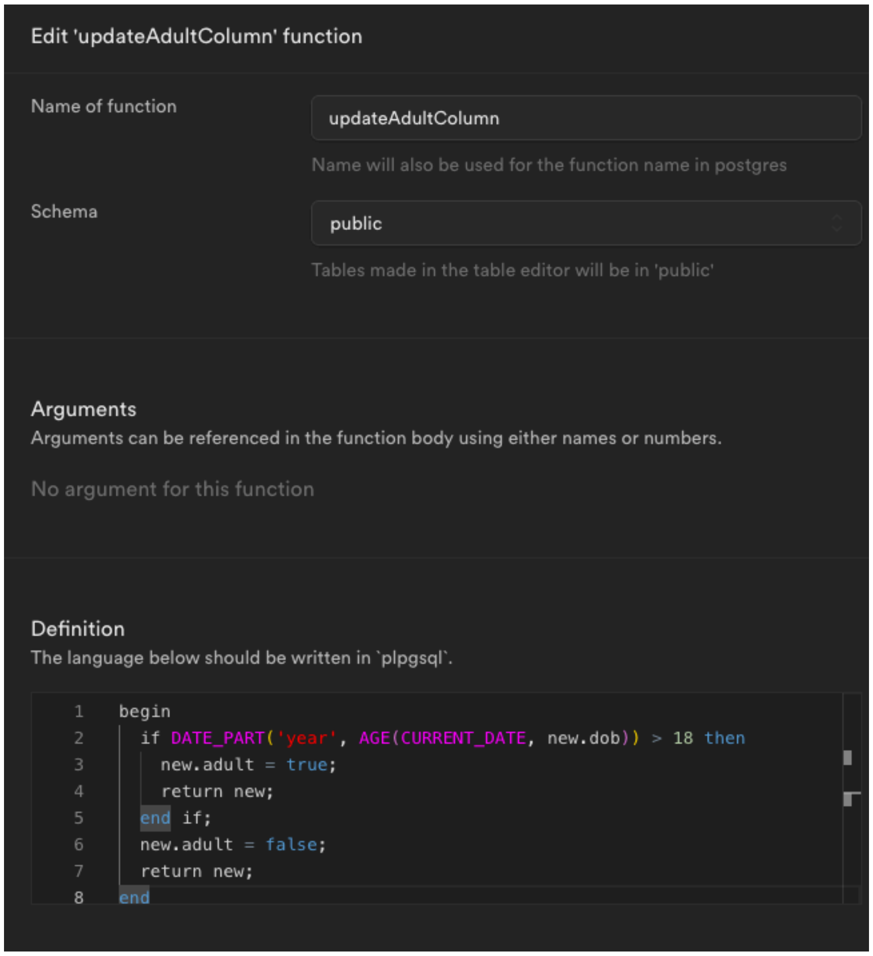


Name	Table	Function	Events
insertProfileForNewUser	users	insertProfileForNewUser	AFTER INSERT

schema public

Name	Table	Function	Events
insertToMembers	members	updateAdultYearColunms	BEFORE INSERT
insertToMembersBefore	members	updateAdultColumn	BEFORE INSERT
paymentTableInsertUpdateMember	payments	updateMemberPaymentInfo	AFTER INSERT

Another function to check whether the new member is over the age on 18 and the adult column is populated with true or false based on that.


Edit 'updateAdultColumn' function

Name of function: updateAdultColumn

Name will also be used for the function name in postgres

Schema: public

Tables made in the table editor will be in 'public'

Arguments

Arguments can be referenced in the function body using either names or numbers.

No argument for this function

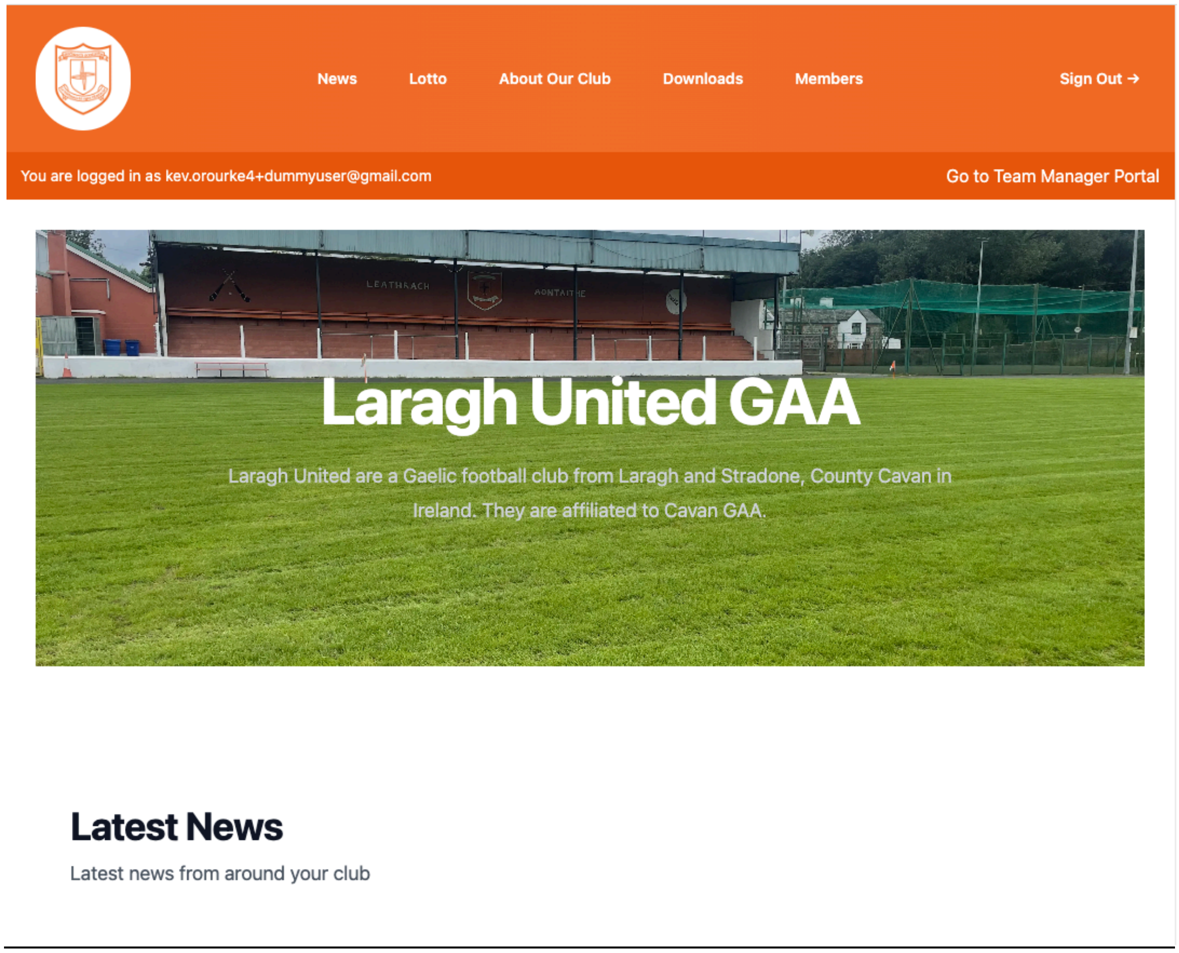
Definition

The language below should be written in `plpgsql`.

```
begin
  if DATE_PART('year', AGE(CURRENT_DATE, new.dob)) > 18 then
    new.adult = true;
    return new;
  end if;
  new.adult = false;
  return new;
end
```

2.4. Graphical User Interface (GUI)

Homepage with logged in team manager status bar



You are logged in as kev.orourke4+dummyuser@gmail.com

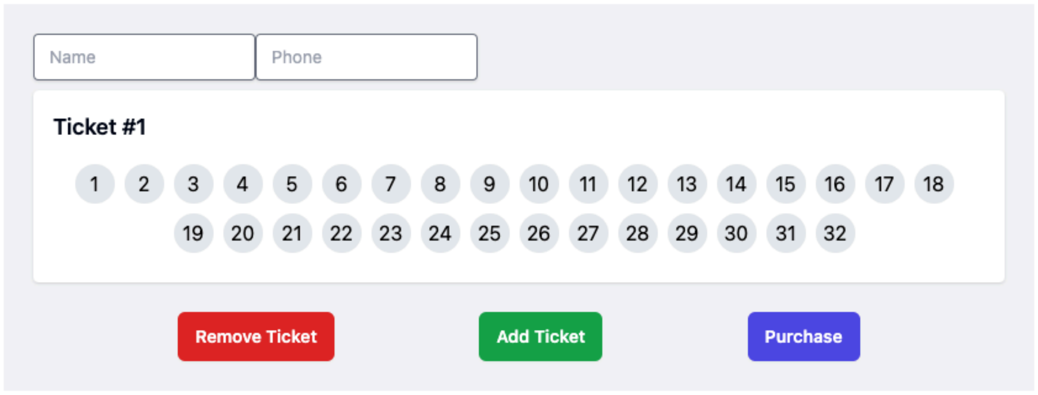
Laragh United GAA

Laragh United are a Gaelic football club from Laragh and Stradone, County Cavan in Ireland. They are affiliated to Cavan GAA.

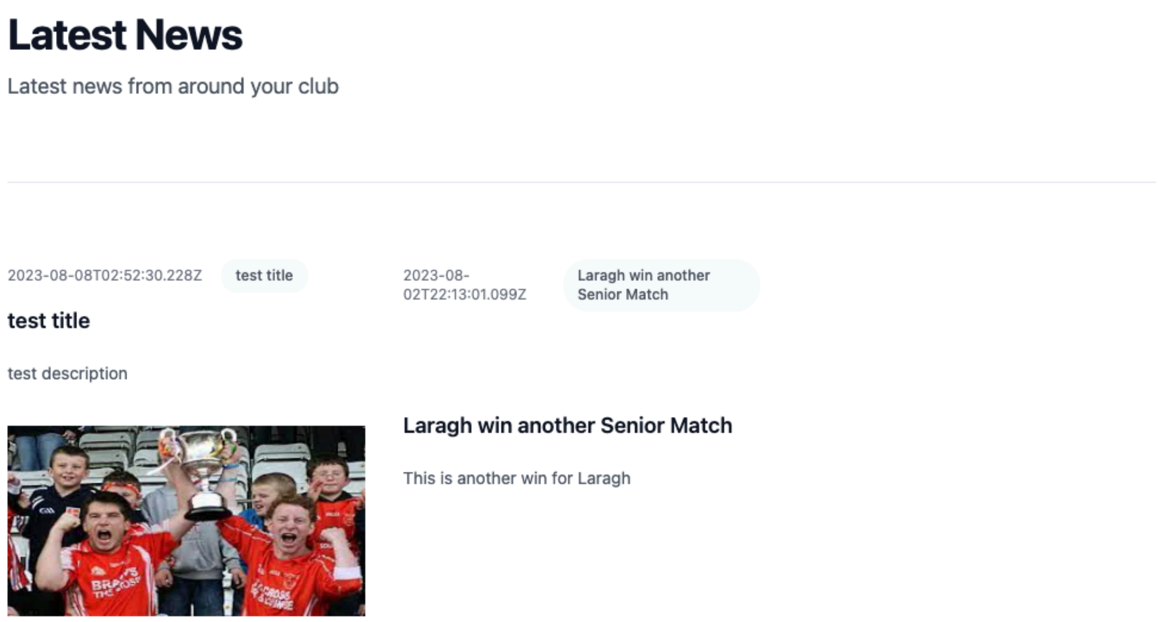
Latest News

Latest news from around your club

<u>Lotto Ticket Generator</u>



News page



Latest News

Latest news from around your club

2023-08-08T02:52:30.228Z test title

test title

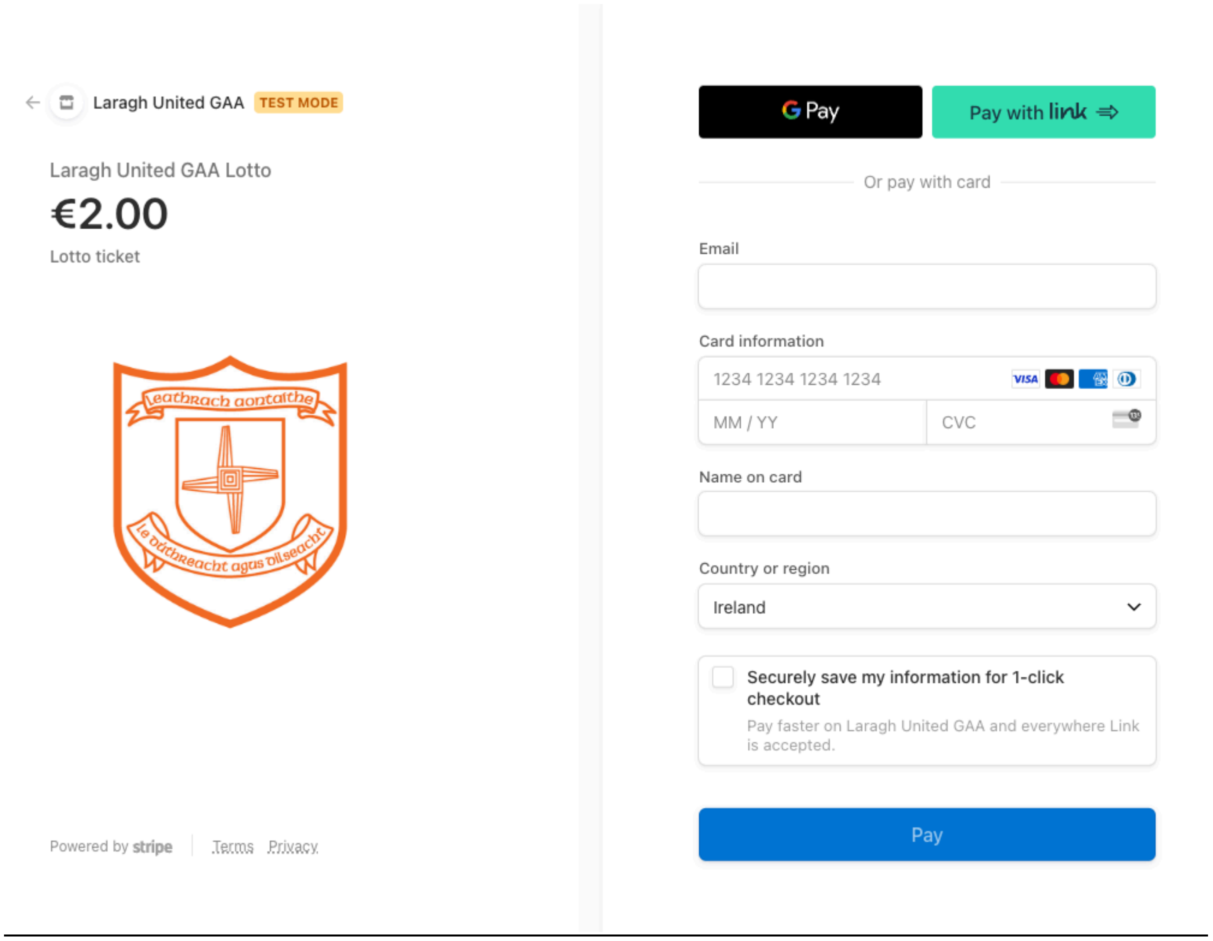
test description

2023-08-02T22:13:01.099Z Laragh win another Senior Match

Laragh win another Senior Match

This is another win for Laragh

Stripe Lotto Ticket Checkout



Laragh United GAA TEST MODE

Laragh United GAA Lotto

€2.00

Lotto ticket

Powered by stripe | Terms Privacy

G Pay | Pay with link

Or pay with card

Email

Card information

1234 1234 1234 1234

MM / YY

CVC

Name on card

Country or region

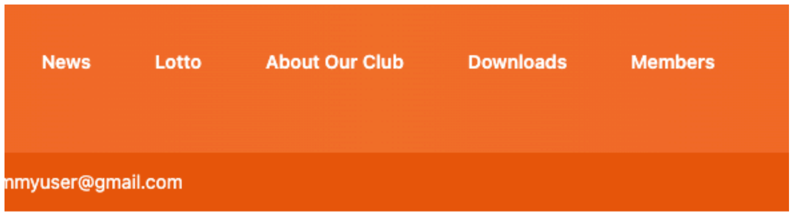
Ireland

Securely save my information for 1-click checkout

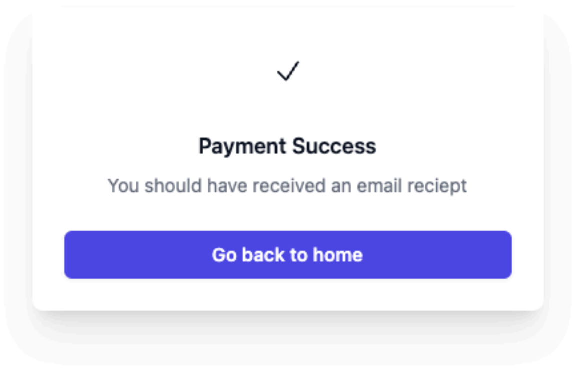
Pay faster on Laragh United GAA and everywhere Link is accepted.

Pay

Payment Success


News Lotto About Our Club Downloads Members

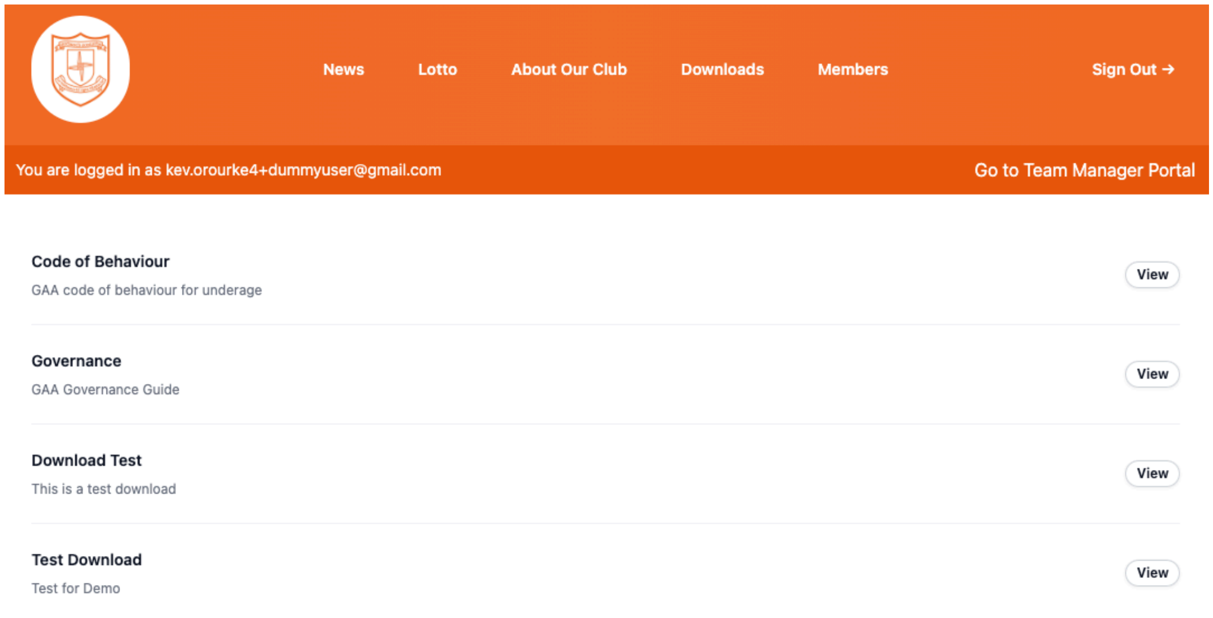
nmyuser@gmail.com


Payment Success

You should have received an email reciept

Go back to home

Download Content


News Lotto About Our Club Downloads Members Sign Out →

You are logged in as kev.orourke4+dummyuser@gmail.com

Go to Team Manager Portal

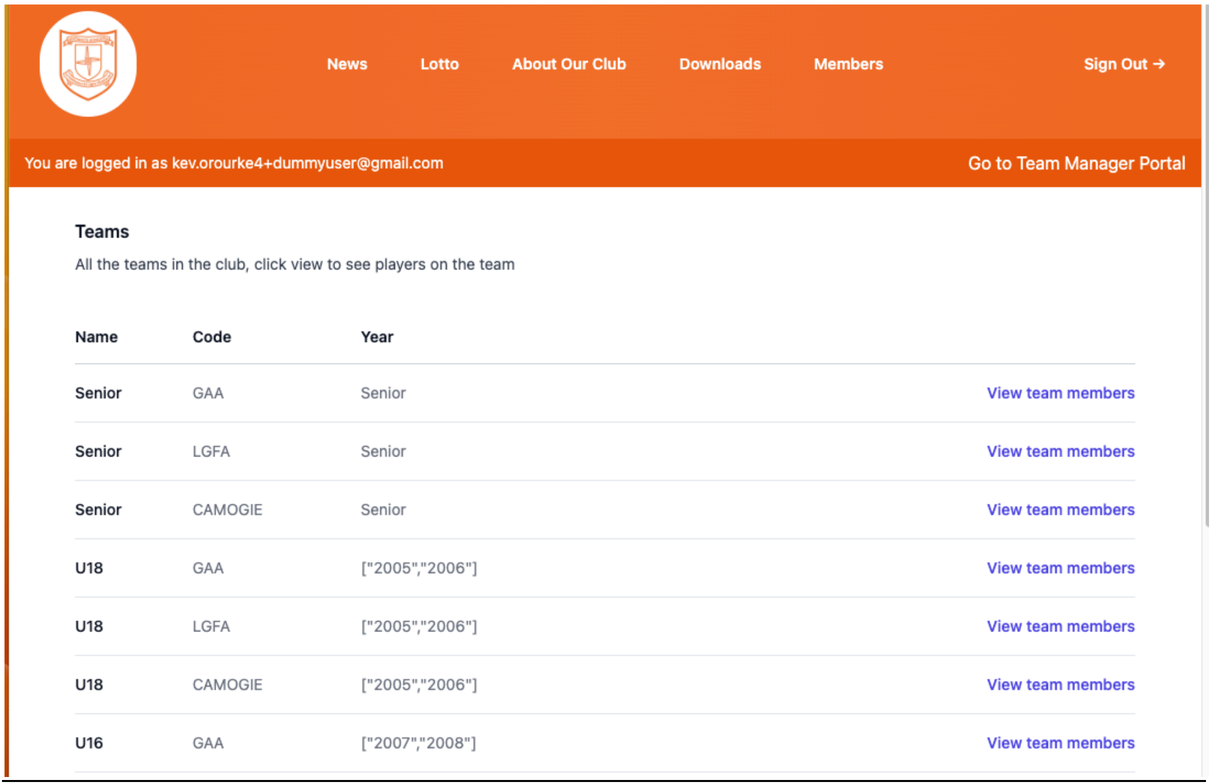
Code of Behaviour
GAA code of behaviour for underage
View

Governance
GAA Governance Guide
View

Download Test
This is a test download
View

Test Download
Test for Demo
View

Team Manager Portal



You are logged in as kev.orourke4+dummyuser@gmail.com

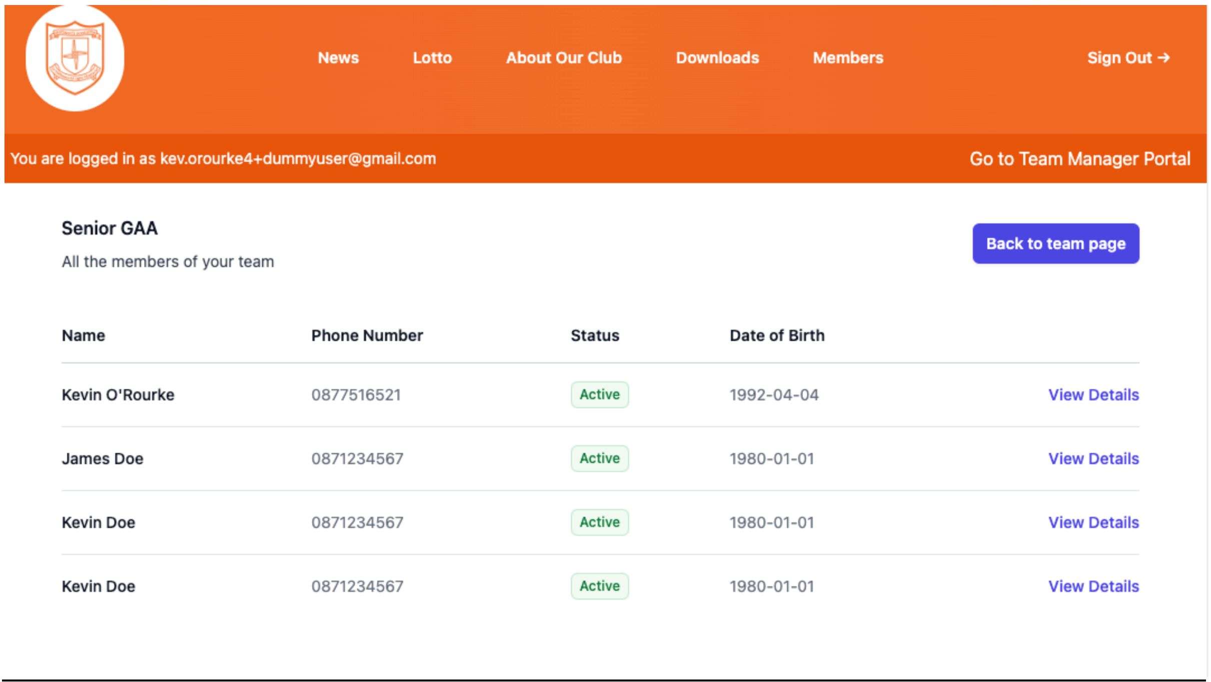
Go to Team Manager Portal

Teams

All the teams in the club, click view to see players on the team

Name	Code	Year	
Senior	GAA	Senior	View team members
Senior	LGFA	Senior	View team members
Senior	CAMOGIE	Senior	View team members
U18	GAA	["2005","2006"]	View team members
U18	LGFA	["2005","2006"]	View team members
U18	CAMOGIE	["2005","2006"]	View team members
U16	GAA	["2007","2008"]	View team members

Team Members List



You are logged in as kev.orourke4+dummyuser@gmail.com

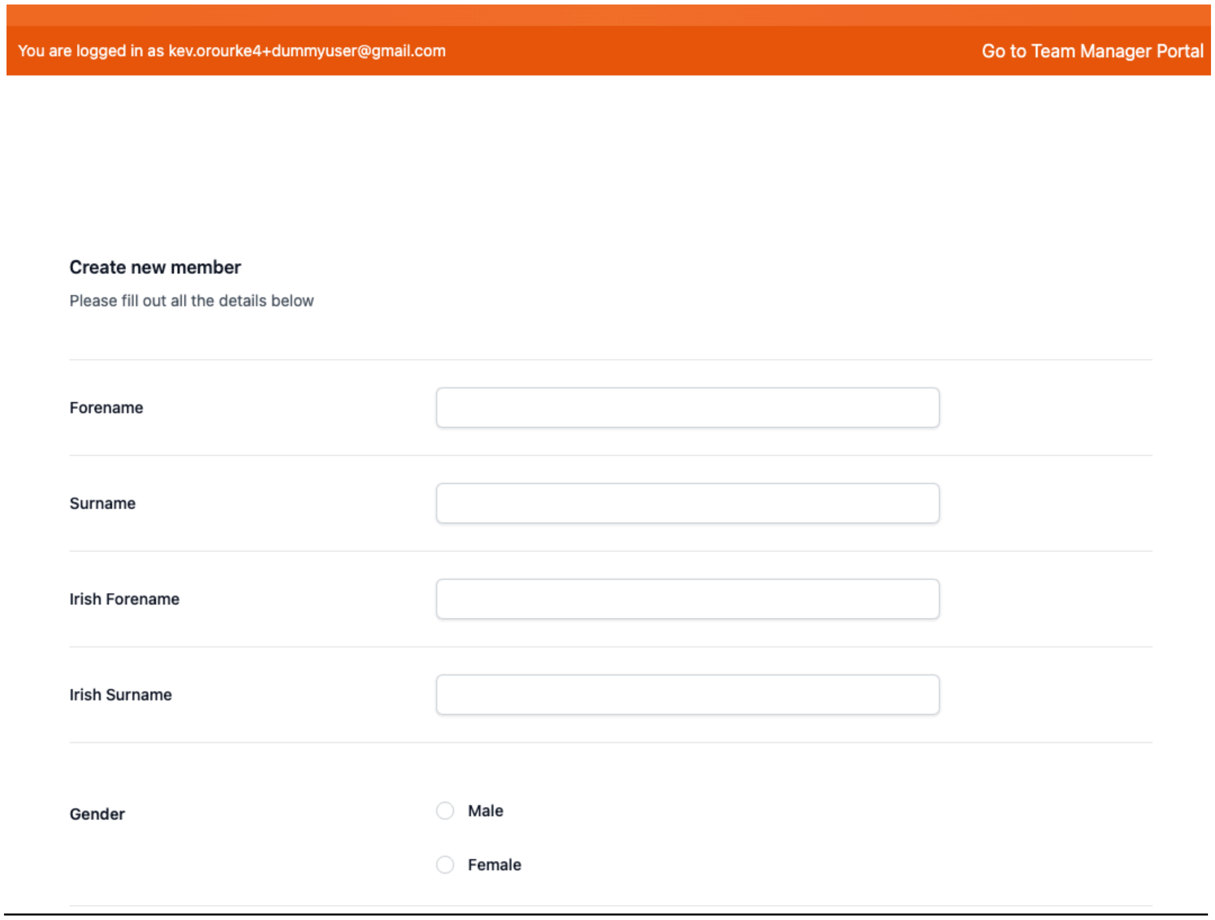
Go to Team Manager Portal

Senior GAA

All the members of your team

Name	Phone Number	Status	Date of Birth	
Kevin O'Rourke	0877516521	Active	1992-04-04	View Details
James Doe	0871234567	Active	1980-01-01	View Details
Kevin Doe	0871234567	Active	1980-01-01	View Details
Kevin Doe	0871234567	Active	1980-01-01	View Details

Add New Member



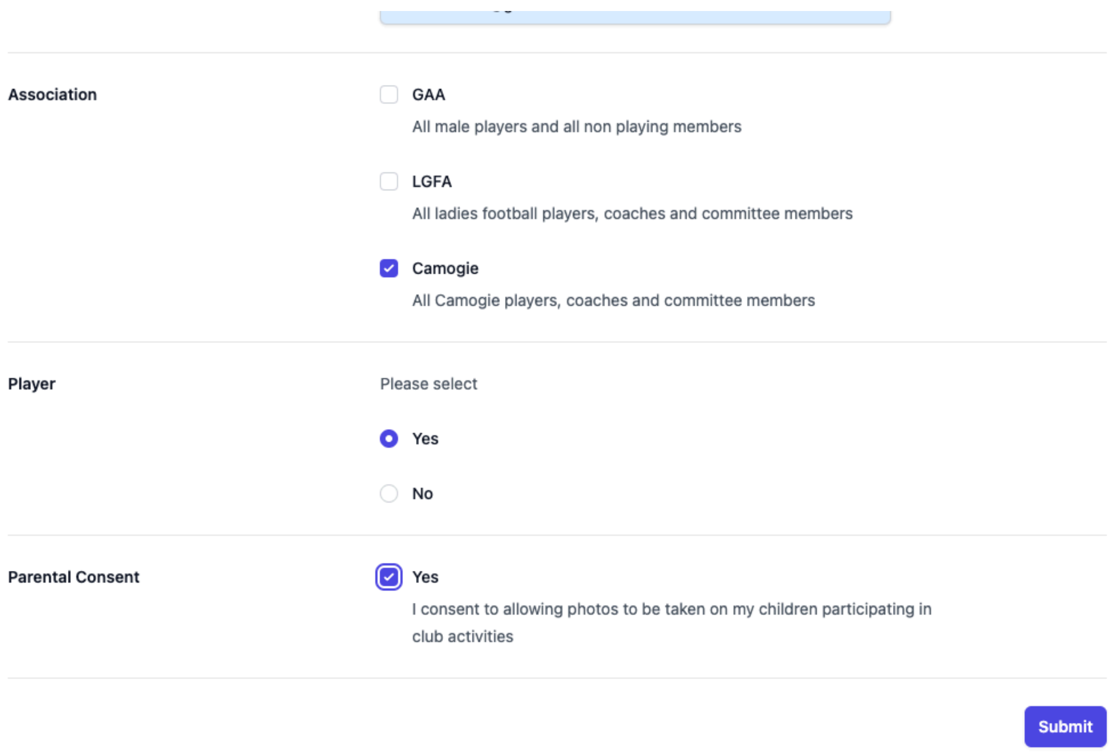
You are logged in as kev.orourke4+dummyuser@gmail.com

Go to Team Manager Portal

Create new member

Please fill out all the details below

Forename	
Surname	
Irish Forename	
Irish Surname	
Gender	○ Male ○ Female



Association

[✓ Camogie] GAA, LGFA

GAA
All male players and all non playing members

LGFA
All ladies football players, coaches and committee members

✓ Camogie
All Camogie players, coaches and committee members

Player

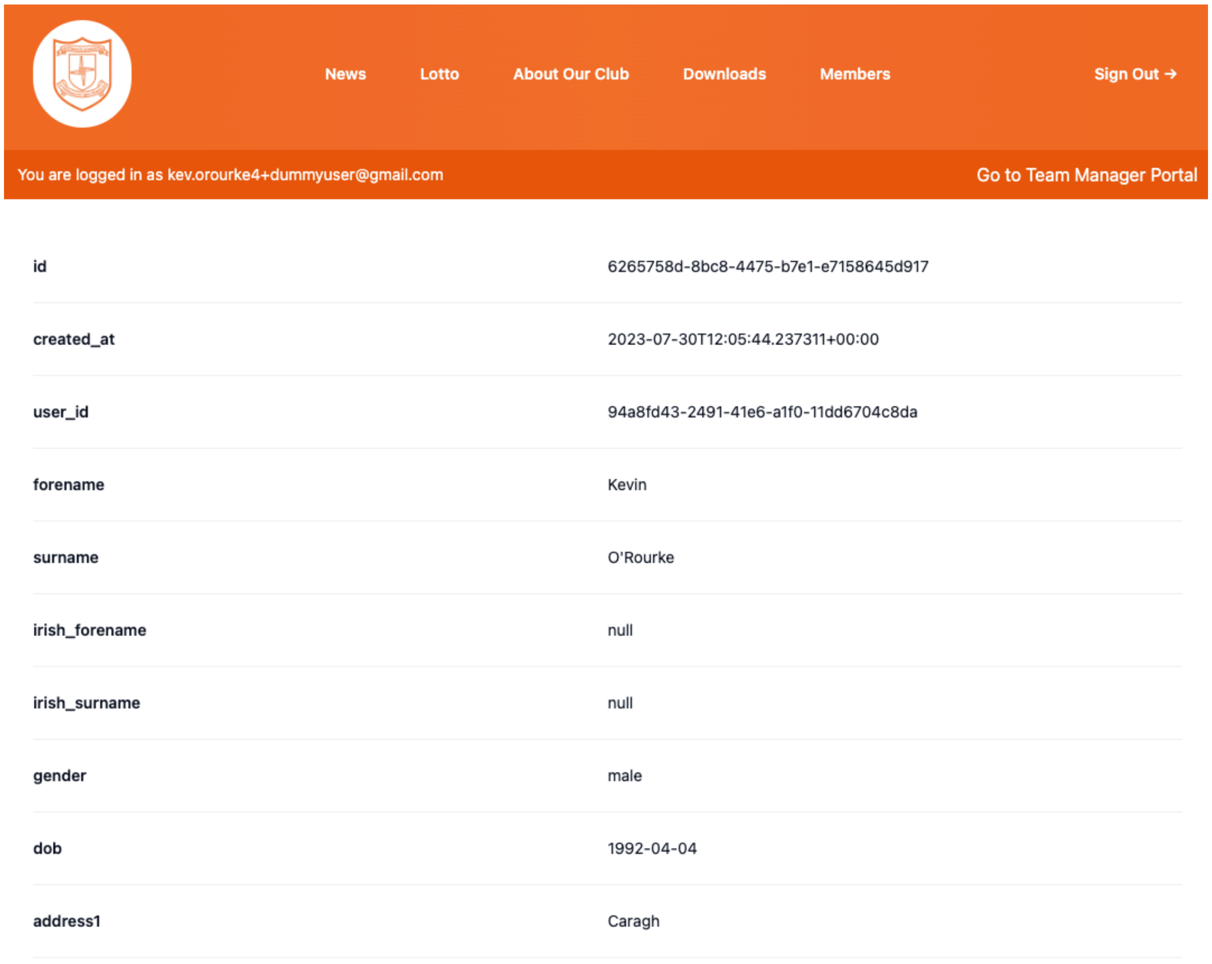
Please select

[✓ Yes] No

Parental Consent

✓ Yes
I consent to allowing photos to be taken on my children participating in club activities

Members Details Page

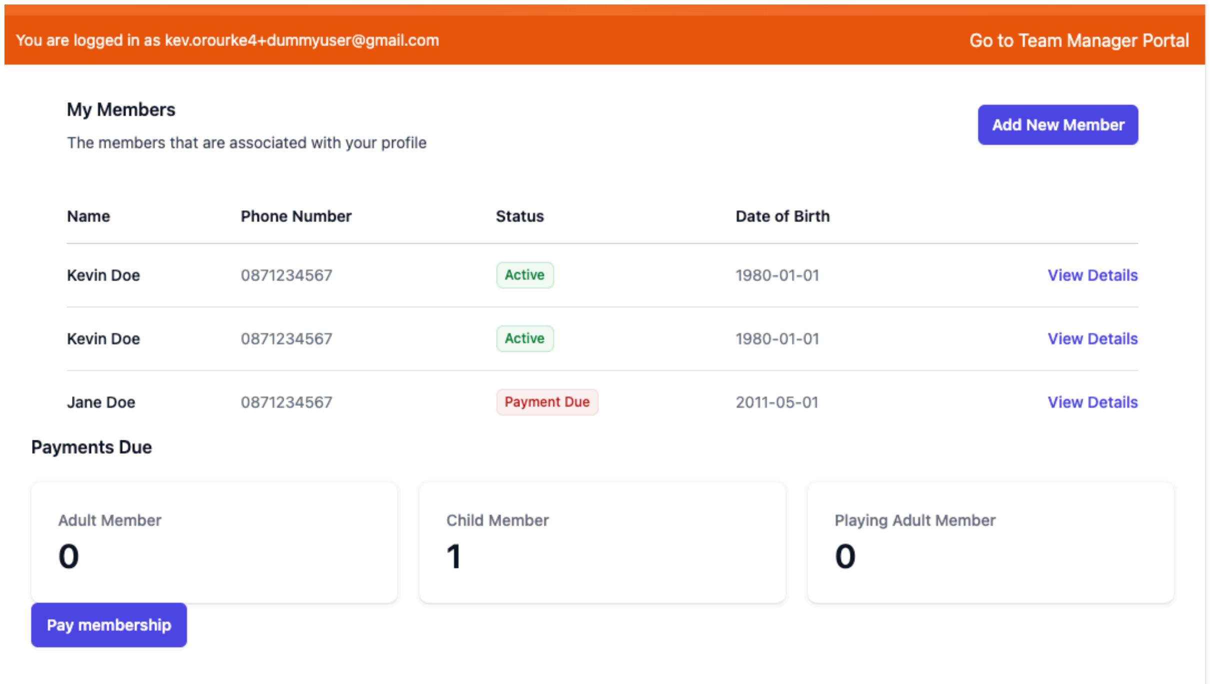


You are logged in as kev.orourke4+dummyuser@gmail.com

Go to Team Manager Portal

id	6265758d-8bc8-4475-b7e1-e7158645d917
created_at	2023-07-30T12:05:44.237311+00:00
user_id	94a8fd43-2491-41e6-a1f0-11dd6704c8da
forename	Kevin
surname	O'Rourke
irish_forename	null
irish_surname	null
gender	male
dob	1992-04-04
address1	Caragh

Member dashboard and payments due components


You are logged in as kev.orourke4+dummyuser@gmail.com

Go to Team Manager Portal

My Members

The members that are associated with your profile

Add New Member

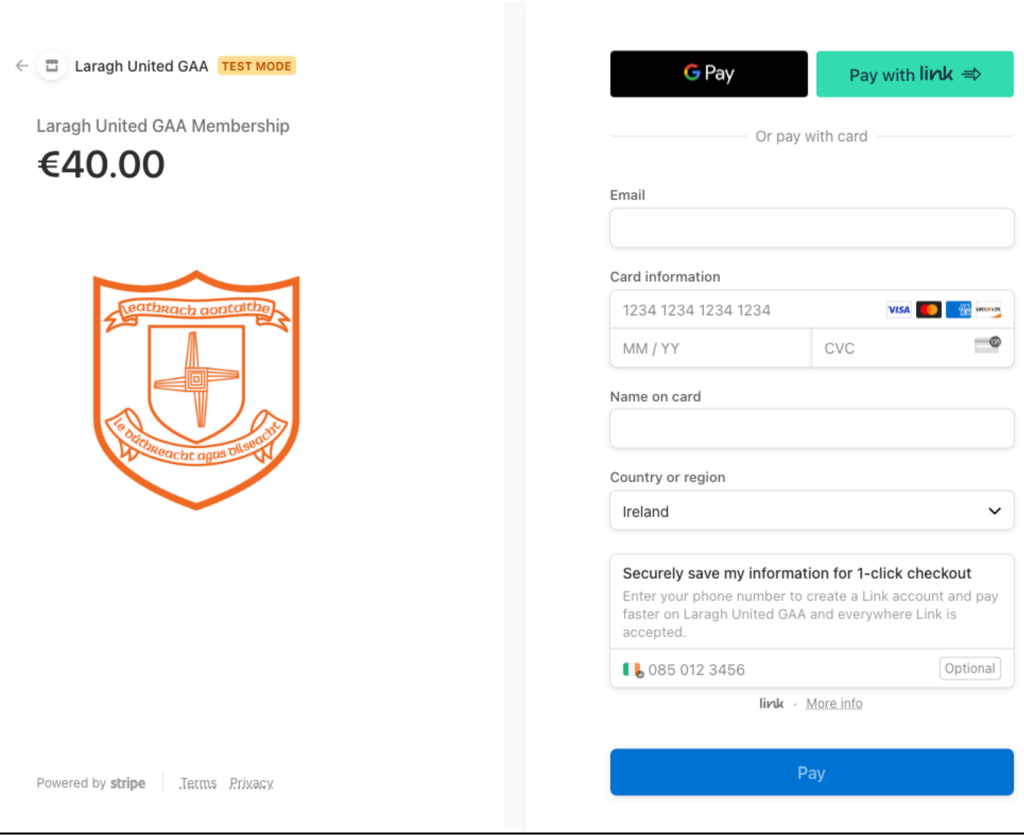
Name	Phone Number	Status	Date of Birth	
Kevin Doe	0871234567	Active	1980-01-01	View Details
Kevin Doe	0871234567	Active	1980-01-01	View Details
Jane Doe	0871234567	Payment Due	2011-05-01	View Details

Payments Due

Adult Member	Child Member	Playing Adult Member
0	1	0

Pay membership

Pay membership through Stripe


Laragh United GAA TEST MODE

Laragh United GAA Membership

€40.00

Powered by stripe | Terms Privacy

G Pay | Pay with link

Or pay with card

Email

Card information

1234 1234 1234 1234

MM / YY | CVC

Name on card

Country or region

Ireland

Securely save my information for 1-click checkout

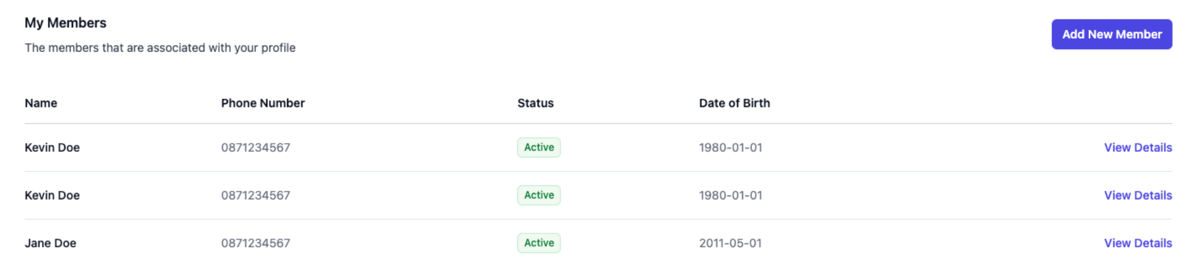
Enter your phone number to create a Link account and pay faster on Laragh United GAA and everywhere Link is accepted.

085 012 3456 Optional

link · More info

Pay

Payment complete and payment due status and components have changed



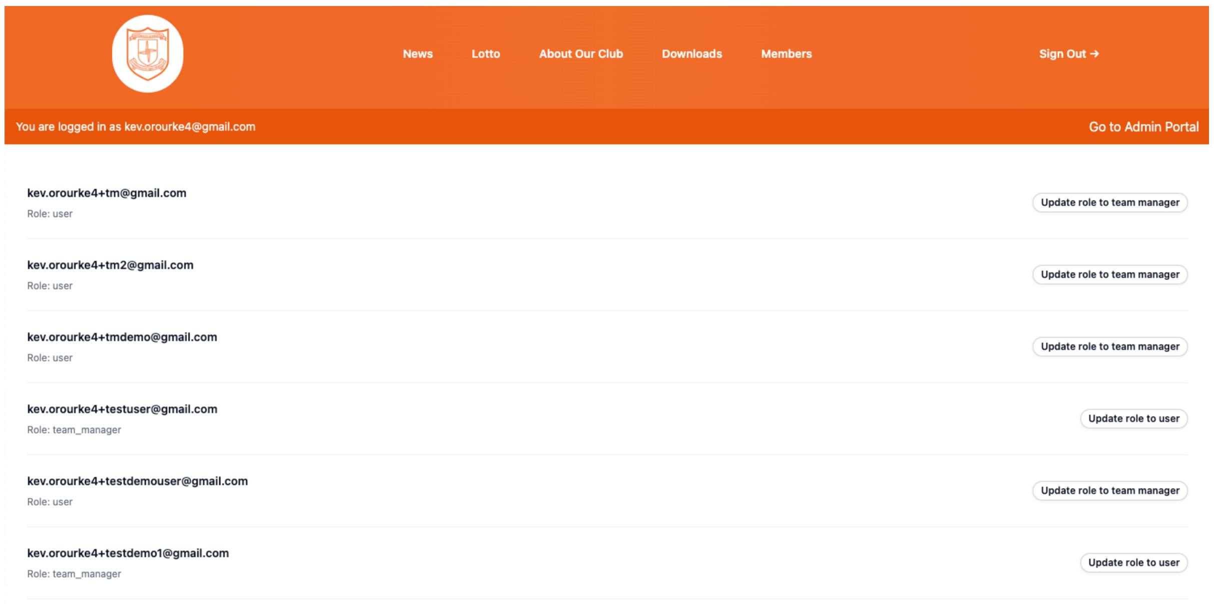
My Members

The members that are associated with your profile

Add New Member

Name	Phone Number	Status	Date of Birth	
Kevin Doe	0871234567	Active	1980-01-01	View Details
Kevin Doe	0871234567	Active	1980-01-01	View Details
Jane Doe	0871234567	Active	2011-05-01	View Details

Admin view to change the role of profiles



You are logged in as kev.orourke4@gmail.com

Go to Admin Portal

kev.orourke4+tm@gmail.com
Role: user
Update role to team manager

kev.orourke4+tm2@gmail.com
Role: user
Update role to team manager

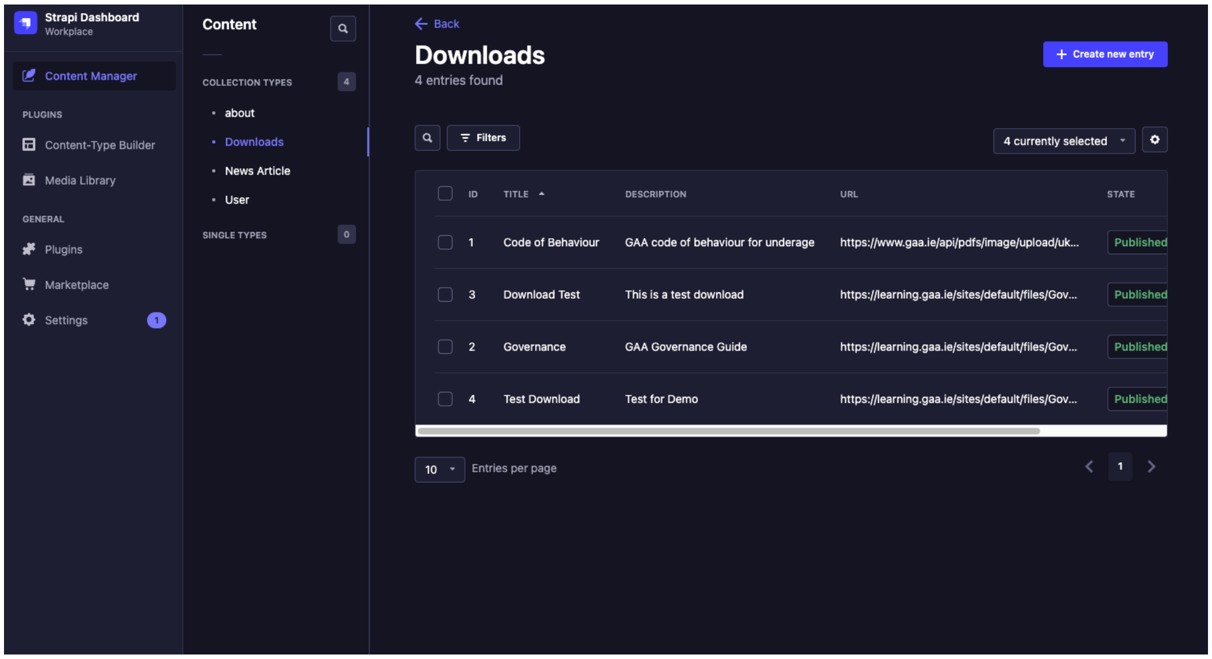
kev.orourke4+tmdemo@gmail.com
Role: user
Update role to team manager

kev.orourke4+testuser@gmail.com
Role: team_manager
Update role to user

kev.orourke4+testdemouser@gmail.com
Role: user
Update role to team manager

kev.orourke4+testdemo1@gmail.com
Role: team_manager
Update role to user

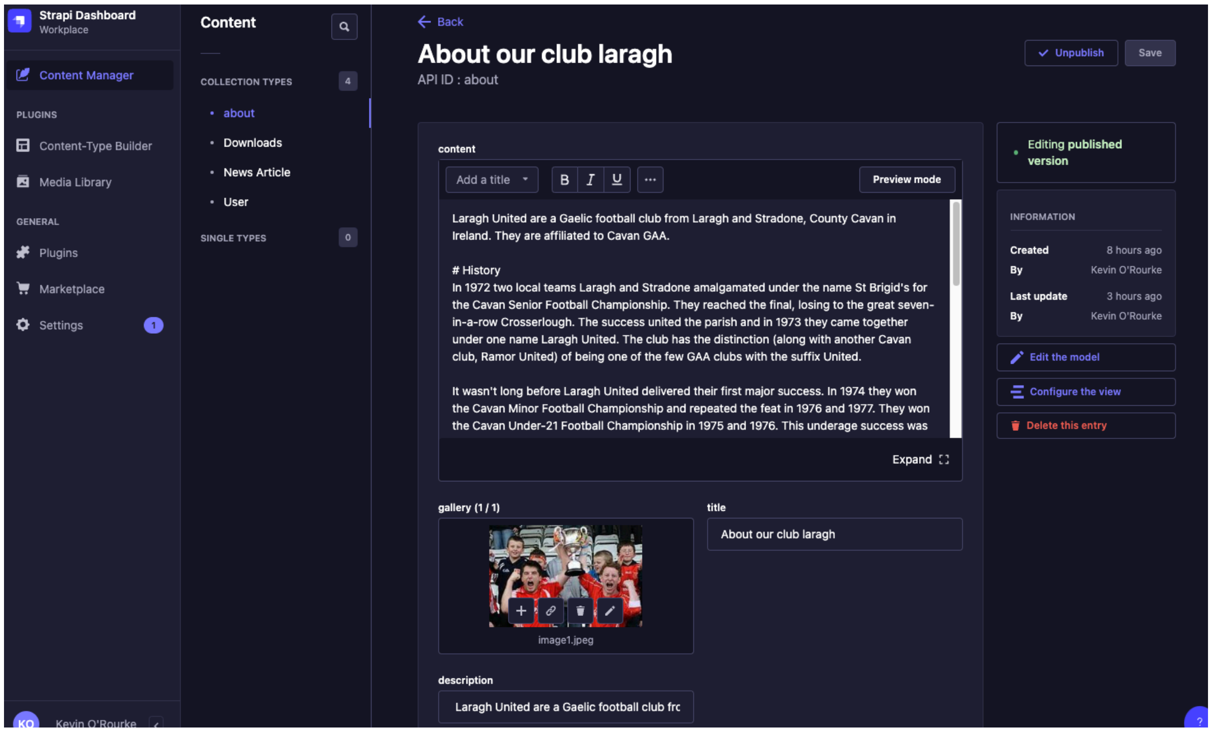
Strapi UI



Downloads
4 entries found

ID	TITLE	DESCRIPTION	URL	STATE
1	Code of Behaviour	GAA code of behaviour for underage	https://www.gaa.ie/api/pdfs/image/upload/uk...	Published
3	Download Test	This is a test download	https://learning.gaa.ie/sites/default/files/Gov...	Published
2	Governance	GAA Governance Guide	https://learning.gaa.ie/sites/default/files/Gov...	Published
4	Test Download	Test for Demo	https://learning.gaa.ie/sites/default/files/Gov...	Published

10 Entries per page 1



About our club laragh
API ID : about

content

Laragh United are a Gaelic football club from Laragh and Stradone, County Cavan in Ireland. They are affiliated to Cavan GAA.

History
In 1972 two local teams Laragh and Stradone amalgamated under the name St Brigid's for the Cavan Senior Football Championship. They reached the final, losing to the great seven-in-a-row Crosserlough. The success united the parish and in 1973 they came together under one name Laragh United. The club has the distinction (along with another Cavan club, Ramor United) of being one of the few GAA clubs with the suffix United.

It wasn't long before Laragh United delivered their first major success. In 1974 they won the Cavan Minor Football Championship and repeated the feat in 1976 and 1977. They won the Cavan Under-21 Football Championship in 1975 and 1976. This underage success was

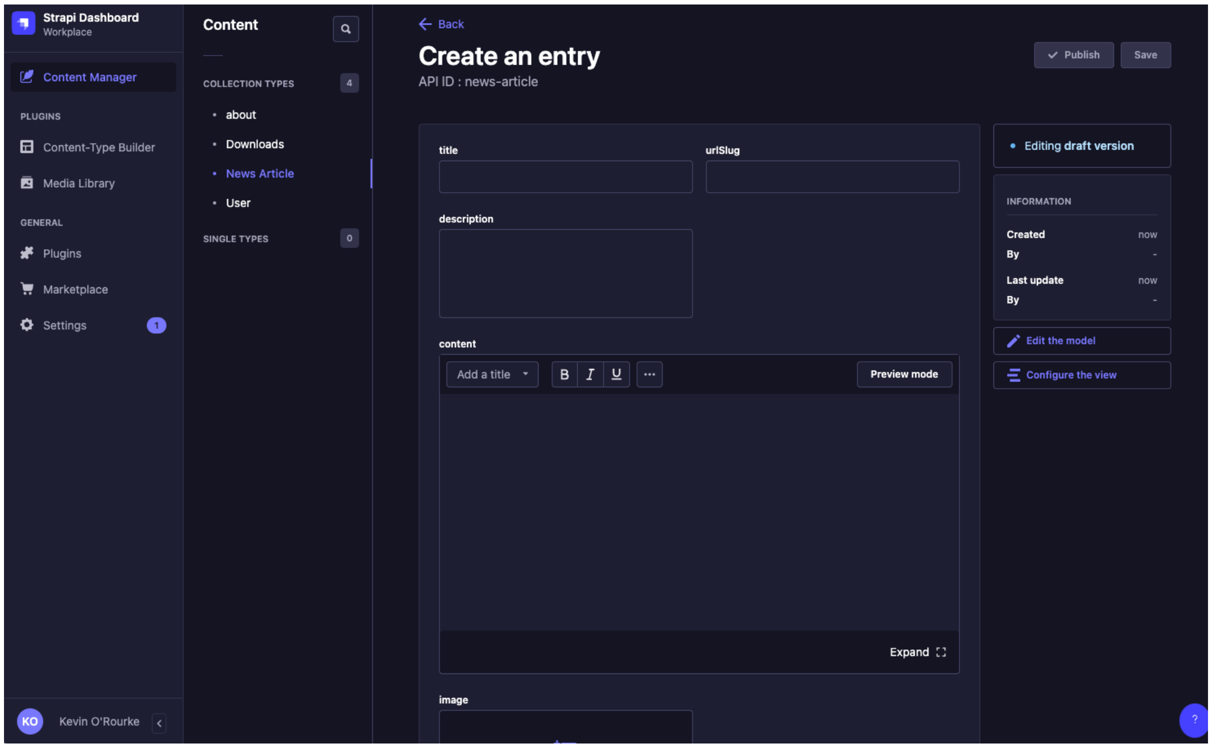
gallery (1 / 1)
image1.jpeg

title
About our club laragh

description
Laragh United are a Gaelic football club frc

Editing published version

INFORMATION
Created 8 hours ago
By Kevin O'Rourke
Last update 3 hours ago
By Kevin O'Rourke



2.5. Testing

Testing Tools:

Jest: A comprehensive testing framework used primarily for unit testing in JavaScript applications. Its ability to quickly run tests in watch mode, generate snapshots, and work seamlessly with various libraries makes it an essential tool for testing Next.js applications.

React Testing Library: A utility built on top of Jest to test React components. It emphasizes testing components as users would, making it perfect for unit and integration tests.

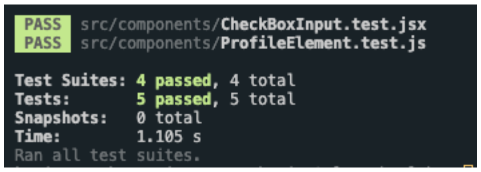
Testing Completed

Unit Tests:

- Testing individual functions or components in isolation.
- Ensuring core utility functions return expected outputs for given inputs.

- Validating UI components display correctly and react to user inputs as intended, using snapshots and React Testing Library.

Evidence of unit tests passing on functions below

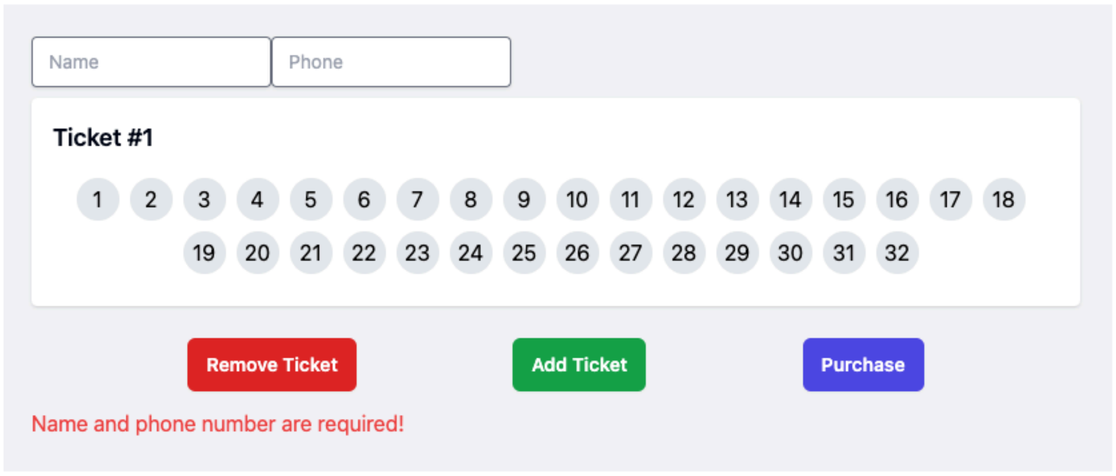


```
PASS  src/components/CheckBoxInput.test.jsx
PASS  src/components/ProfileElement.test.js

Test Suites: 4 passed, 4 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        1.105 s
Ran all test suites.
```

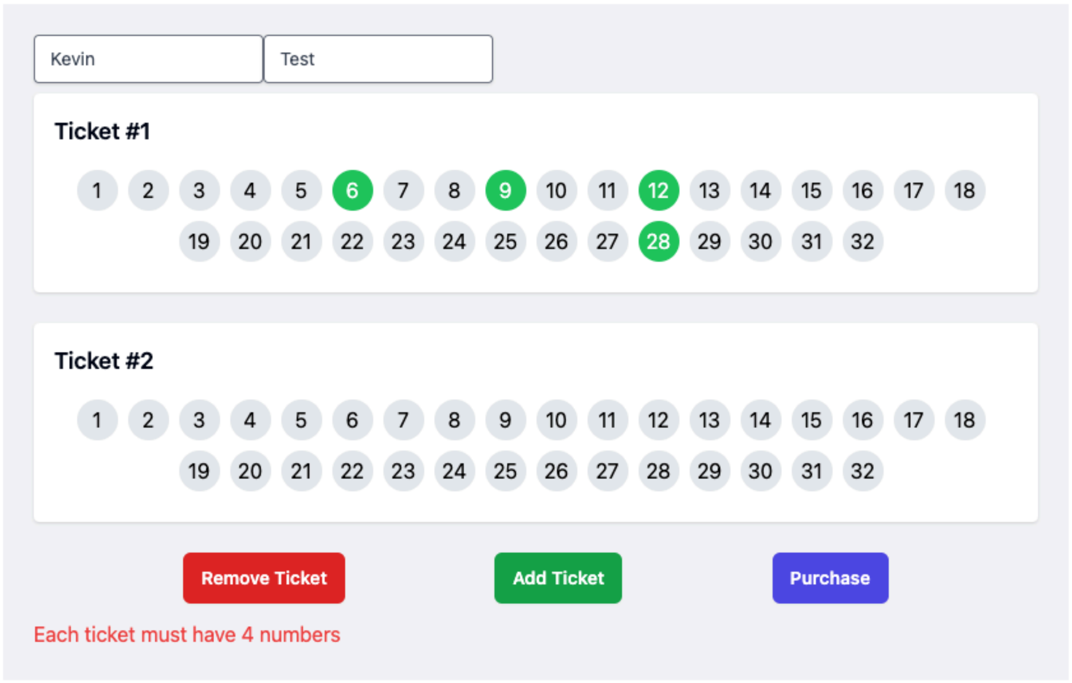
<u>End User Testing:</u>

- Simulating real-world scenarios where a user interacts with the application.
- This was manual.
- Gathered feedback from actual users to find usability pain points or uncaught bugs.

Client side validation introduced below as a result of this.



Name and phone number are required!

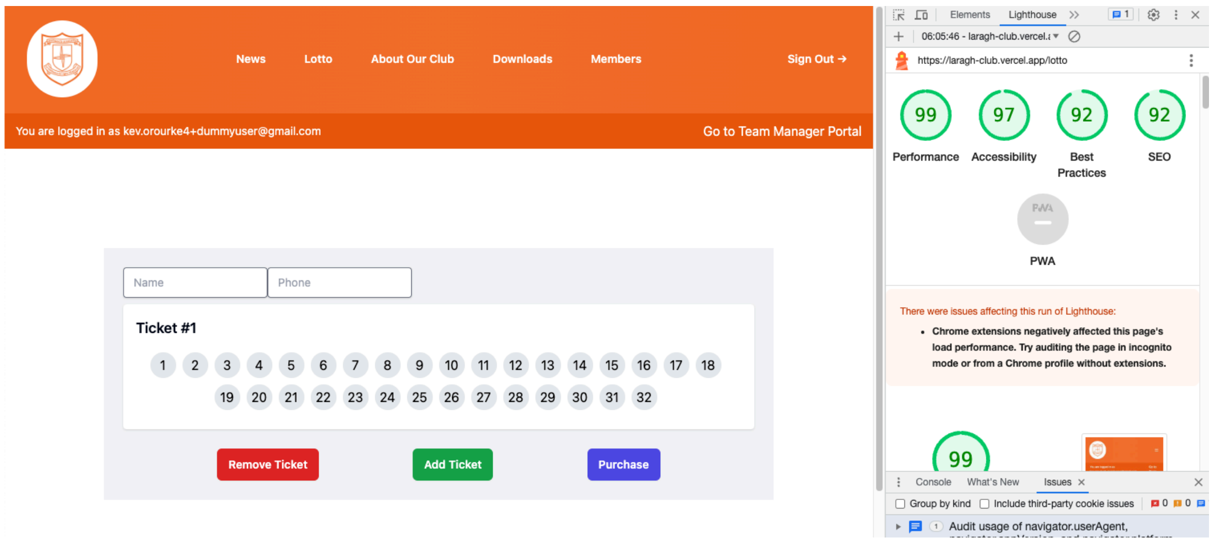


Each ticket must have 4 numbers

<u>Speed, SEO and Accessibility testing:</u>

- Speed testing is crucial in evaluating the performance of web applications, especially in today's fast-paced digital world where user patience is limited. Slow-loading websites can deter users and negatively affect user experience.
- SEO testing ensures that the web application is optimized for search engines, leading to better visibility, higher organic traffic, and improved user engagement.
- Accessibility testing ensures that the application is usable by everyone, including people with disabilities.

Results of Google Lighthouse test below on both a client and server component.


News Lotto About Our Club Downloads Members Sign Out →

You are logged in as kev.orourke4+dummyuser@gmail.com Go to Team Manager Portal

Name Phone

Ticket #1

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
19 20 21 22 23 24 25 26 27 28 29 30 31 32

Remove Ticket Add Ticket Purchase

06:05:46 - laragh-club.vercel.a

https://laragh-club.vercel.app/lotto

99 Performance 97 Accessibility 92 Best Practices 92 SEO PWA

There were issues affecting this run of Lighthouse:

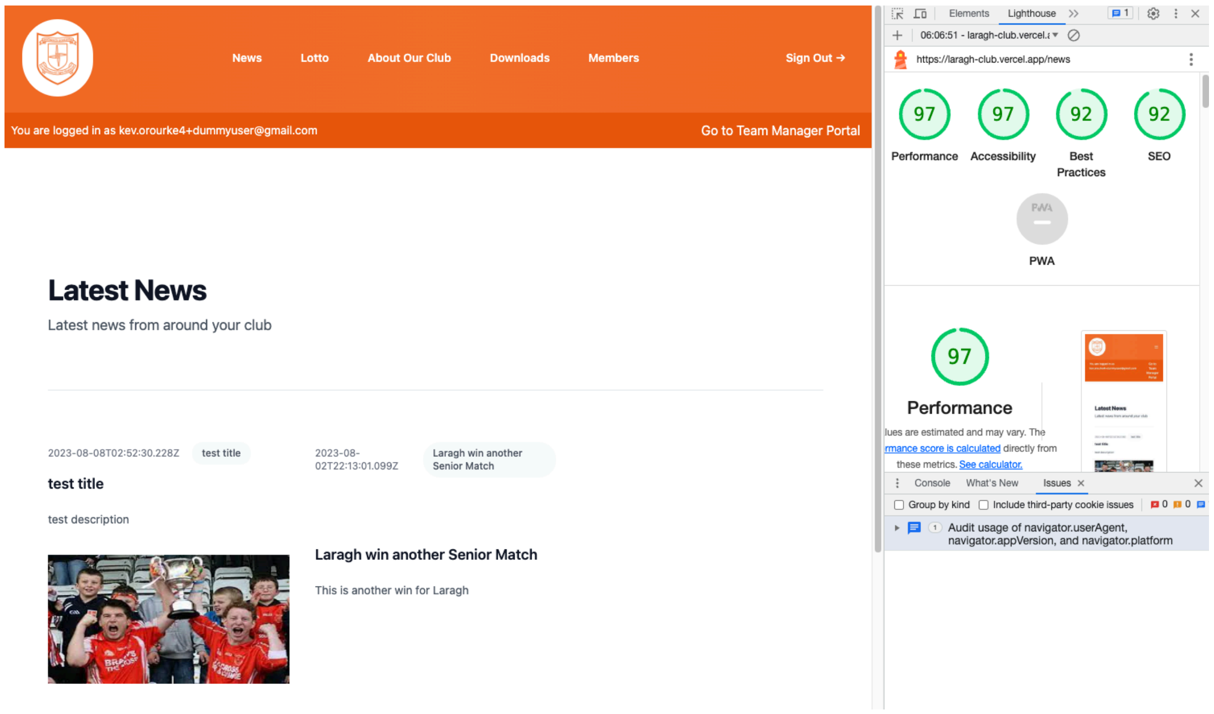
- Chrome extensions negatively affected this page's load performance. Try auditing the page in incognito mode or from a Chrome profile without extensions.

99

Console What's New Issues

Group by kind Include third-party cookie issues

Audit usage of navigator.userAgent, navigator.appVersion, and navigator.platform


News Lotto About Our Club Downloads Members Sign Out →

You are logged in as kev.orourke4+dummyuser@gmail.com Go to Team Manager Portal

Latest News

Latest news from around your club

2023-08-08T02:52:30.228Z test title

test title

test description

2023-08-02T22:13:01.099Z Laragh win another Senior Match

Laragh win another Senior Match

This is another win for Laragh

06:06:51 - laragh-club.vercel.a

https://laragh-club.vercel.app/news

97 Performance 97 Accessibility 92 Best Practices 92 SEO PWA

97 Performance

lues are estimated and may vary. The rmance score is calculated directly from these metrics. See calculator.

Console What's New Issues

Group by kind Include third-party cookie issues

Audit usage of navigator.userAgent, navigator.appVersion, and navigator.platform

2.6. Evaluation

The evaluation process proved the robustness and efficiency of the application. User feedback was positive, and technical benchmarks such as the lighthouse report shines a light on the technology used for the application. There are also many areas of improvement identified.

3.0 Conclusions

Advantages

User-Centric Design: The project caters to a diverse set of user roles, from club members to team managers and admin users. By ensuring each role has a tailored user experience, the system maximizes usability and user satisfaction.

Scalability: Leveraging technologies like Next.js, Strapi CMS, and Supabase, the project is designed to efficiently handle increased user loads and traffic spikes, ensuring consistent performance.

Security: The integration of established services like Stripe for payment processing ensures high standards of data protection and financial transaction security.

Modern Technologies: By employing a cutting-edge tech stack, including Next.js and Strapi CMS, the project offers speed, efficiency, and future-readiness.

Functionality: Beyond member management, the system offers functionalities like lotto ticket purchasing and content management, making it a comprehensive solution for club management needs.

Disadvantages

Complexity: The range of functionalities and user roles means that the system's backend is complex, which can pose challenges in maintenance and further development.

Dependence on Third-Party Services: The system's reliance on services like Stripe, Supabase, and Amazon SES means that any disruptions or changes in these services can directly impact the project's performance and functionality.

Steep Learning Curve: It was challenging to develop, integrate and deploy this cutting-edge tech stack. Integration with the third party services being a challenge as there is limited documentation on Next.js App Router.

The project, while having its set of strengths, particularly in usability, scalability, and integration of modern technologies, also has certain limitations, mainly around complexity and dependence on third-party services. However, the advantages significantly outweigh the disadvantages, making it a valuable solution for club management needs.

4.0 Further Development

Given additional time and resources I would expand in the following areas.

- Manage lotto draw, lotto results and winners within the application. Including an analytics dashboard to help drive revenue and recurring use.
- Enhance the user profile so they can manage more of their details.
- Migrate to a mobile app to enable push notifications for lotto draws, team events/messages and fundraisers.
- Expand to a number of clubs, given the backend system is all third party services. Templating the current application and rolling out at scale is a very feasible task.

5.0 Appendices

5.1. Project Proposal

National College of Ireland

BSc in Computing

2017/2018

Kevin O'Rourke

X15042782

X15042782@student.ncirl.ie

MyClubSubs

Technical Report

National College *of* Ireland

Table of Contents

Executive Summary

This report provides the background, goals and requirements for the MyClubSubs web application. The MyClubSubs application aims to fill a gap in the market for a club members management web application. The end goal of the application is to provide a members management for all club types. For the purpose of this project – the main focus will be on having the application production ready for all soccer and GAA clubs. There are over 2000 affiliated GAA clubs and over 1500 soccer clubs in Ireland.

The solution will be to provide a web application where users can register and join multiple clubs as a member. To join as a member the user will be required to pay the membership fee specified by the club. MyClubSubs will provide club admins with many tools to reduce the man hours spent managing the club. Digital and automated tools will allow club admins to manage key duties with maximum efficiency and free up some time.

The goal of this application is to be the complete solution for all club members and administrators.

Introduction

Background

I have been involved in my local GAA club since I was 5 years old. For many years, I was an active playing member. In 2015, I was given the role of club registrar. Club registrar is a role recently introduced by the GAA. The role of the club registrar is to collect membership for every member of the club. My predecessor passed me on sheet of paper with a list of names and addresses for all the club members of the previous year.

The club registrar has no system to work on – in the majority cases all the work is manual. This includes going to training sessions and organized registration nights collecting memberships. The membership fees would be collected in cash and would need to be lodged into a bank account. A written receipt would be given by the registrar to the member and the registrar would keep a hard copy for themselves.

Setting up email distribution groups and SMS distribution groups require a lot of work and management.

The situation is the same in soccer clubs around the country.

I believe this area of amateur sports has been very slow to adopt new technology to make life easier and free up time for club administrators.

Aims

This project aims to address current digital gaps in the running and administration of a volunteer club or organization. The objective is to create a web application to provide club administrators a time saving and cost-efficient platform to manage the day-to-day running of the club. The secondary objective is to provide club members a platform where they can manage their interactions with all the clubs they hold membership.

The aims will be met by providing the following services

- Club Admin services
  - Set memberships types and fees
  - Set up groups
  - Distribute email and text messages to all club members or specific groups.
  - Create membership payment reminders and installment plans
  - Upload fixtures via csv to be displayed on a calendar in the club website
  - Shop for club merchandise, tickets and fundraiser tickets eg. Weekly lotto
  - View income dashboards from membership fees and shop sales
- Club member services
  - View all clubs in central dashboard
  - All upcoming fixtures for the clubs you hold membership will be displayed in one calendar
  - Google maps to provide directions to away games
  - Option to add other family members to user account
  - Forum to chat with other members of the club/group

Technologies

The web application will be developed using a serverless architecture.

An Amazon S3 bucket will be used to host all the static content. I will use Amazon DynamoDB for the database. Lambda functions will be used to request data from the database and return it to the front end. React combined with HTML and CSS will be used to parse the JSON data returned to the front end and present the data on the page to the user.

For my application the following tools, applications and languages will be used.

- Amazon S3
- Amazon DynamoDB
- Amazon Lambda
- Amazon SQS
- HTML
- CSS
- React
- JS
- SQL
- Python
- Google Maps API

System

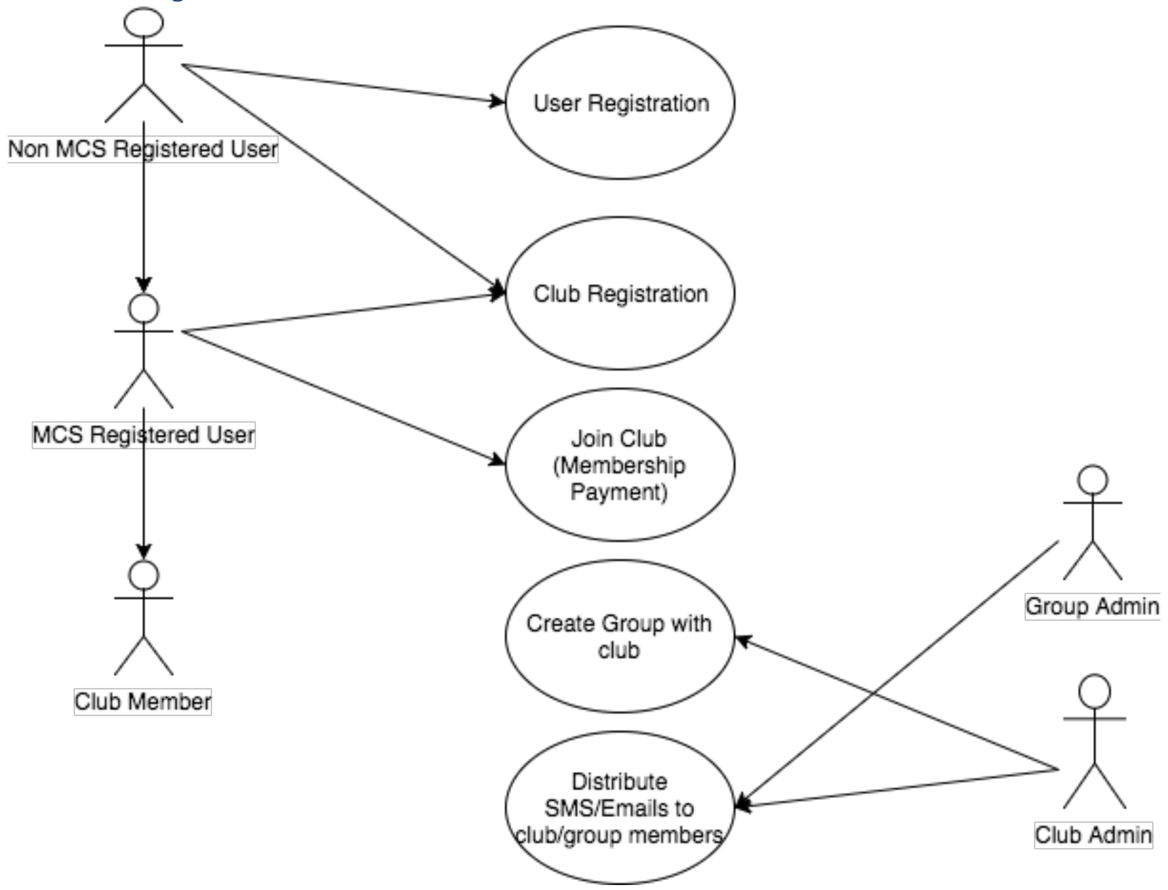
Requirements

Functional requirements

The key functional requirements are ranked in order of priority

- User Registration
- Club Registration
- Join Club (Membership payment)
- Create group within club
- Distribute SMS emails to group/club

Use Case Diagram



Requirement 1 User Registration

Description & Priority

User Registration is the highest priority functional requirement. To become a club member or to register a club – the user is required to be a registered user.

Use Case – User Registration

Scope

The scope of this use case is to allow users to register to the MCS platform.

Description

This use case describes the steps to be taken by the user to register.

Flow Description

Precondition

The web application is on the homepage

Activation

This use case starts when a user navigates to the register page.

Main flow

1. The system displays a form to the user with all the required fields

2. The User submits all the mandatory fields
3. The system validates the details and returns a confirmation
4. The user logs in to their dashboard

Alternate flow

User is already registered.

1. The system returns the message that the user is already registered.
2. The user enters some new details that aren't registered.
3. The use case continues at position 3 of the main flow

Exceptional flow

User does not enter all mandatory data

1. The system returns a message stating not all mandatory fields are filled
2. The user completes the mandatory fields
3. The use case continues at position 3 of the main flow

Termination

The system presents the logged in user dashboard.

Post condition

The system goes into a wait state

Requirement 2 Club Registration

Description & Priority

Add club to MyClubSubs is the highest priority functional requirement. This is the highest priority because without this functionality there would be no clubs for users to subscribe to.

Use Case – Club Registration

Scope

The scope of this use case is to allow registered users to register their club to the MCS platform.

Description

This use case describes the steps to be taken by the user to add the club to the MCS platform.

Flow Description

Precondition

The web application is on the homepage

Activation

This use case starts when a club administrator navigates to the add club page.

Main flow

1. The system displays a form to the user with all the required fields
2. The User submits all the mandatory fields
3. The system validates the details and returns a confirmation
4. The user logs in to manage the club

Alternate flow

Club is already registered.

1. The system returns the message that the club is already registered.
2. The user enters a new club that isn't registered.
3. The use case continues at position 3 of the main flow

Exceptional flow

User does not enter all mandatory data

1. The system returns a message stating not all mandatory fields are filled
2. The user completes the mandatory fields
3. The use case continues at position 3 of the main flow

Termination

The system presents the logged in club management dashboard.

Post condition

The system goes into a wait state

Requirement 3 Join Club (Membership payment)

Description & Priority

After a club has been created. Registered users need to join clubs

Use Case – Join Club (Membership payment)

Scope

The scope of this use case is to allow users to join a club.

Description

This use case describes the steps to be taken by the user to join a club. The payment process is included in this process

Flow Description

Precondition

The web application is on the homepage

Activation

This use case starts when a user navigates to the join club page.

Main flow

1. The system displays a list of clubs available in MCS

2. The User selects the club they wish to join
3. The system loads the selected club membership page
4. The user selects the membership type and completes the payment process
5. The system returns confirmation of club membership

Alternate flow

Payment failed

1. The system returns the message the payment has been declined.
2. The user enters some new payment details.
3. The use case continues at position 4 of the main flow

Termination

The system presents the logged in user dashboard.

Post condition

The system goes into a wait state

Requirement 4 Create Group within club

Description & Priority

The next functional requirement is to allow club administrators to create a group within club.

Use Case – User Registration

Scope

The scope of this use case is to allow club administrators to create a group within club.

Description

This use case describes the steps to be taken by the club admin to create a group.

Flow Description

Precondition

The web application is on the club admin dashboard

Activation

This use case starts when a user clicks the add group button.

Main flow

1. The system displays a form to the user with all the group details.
2. The User submits all the mandatory fields
3. The system validates the details and returns a confirmation
4. The user logs in to their dashboard

Termination

The system presents the logged in club dashboard.

Post condition

The system goes into a wait state

Requirement 5 Distribute SMS/Email - club/group members

Description & Priority

The club and group admins need the functionality to send group texts or emails to all club members or to a particular group.

Use Case – Distribute SMS/Email to club/group members

Scope

The scope of this use case is to allow users send group texts or emails.

Description

This use case describes the steps to be taken to send a group text or email.

Flow Description

Precondition

The web application is on the club dashboard

Activation

This use case starts when a user clicks the send message button on the club page.

Main flow

1. The system displays the message options
2. The User submits all the message details
3. The system validates the message is sent and returns a confirmation
4. The user returns to club dashboard

Alternate flow

User only has permissions to send to specific group.

1. The system returns the message that the user does not have permission to send message to all members.
2. The user selects a group they have permissions to send a message to.
3. The use case continues at position 3 of the main flow

Termination

The system presents the club dashboard.

Post condition

The system goes into a wait state

Non-Functional Requirements

Performance/Response time requirement

The performance and response times will need to meet customer expectations. There seems to be no defined industry standard for web applications but it is widely suggested that web applications need a response time of less than 1 second. MCS web application aims to achieve an average response time of 0.5 seconds. The application will use the AWS serverless web architecture. The site performance and response time will need to be monitored to ensure users are having a good user experience.

Availability requirement

The MCS web application will be available at all times. It is forecasted that the peak usage months for the application will be the first quarter of the year. It is expected that the majority of membership payments will take place in these three months, therefore 100% uptime will be required. The serverless web architecture will allow MCS to be fully scalable and meet all availability requirements.

Recover requirement

The MCS web application will be developed locally and all changes will be pushed to Github with version control in place before being deployed to production. All static content and database will be managed and maintained by AWS.

Robustness requirement

Each piece of functionality on MCS web application will go through a thorough QA phase after each development phase. This testing phase should uncover the majority of the bugs a user may encounter.

Security requirement

Security is major requirement for the MCS web application. Personal data is required to register. Payment details are also being processed on the web application. It is vital to the integrity of MCS that all data remains safe and secure and that there are no customer data breaches. MCS will use data encryption techniques and strict UAM policies and processes to ensure the security of the system.

Reliability requirement

There is an expectation that the MCS web application will be available over 99.5% of the time. This is the standard set by AWS. Therefor the number of failures of the system is expected to be at a minimum.

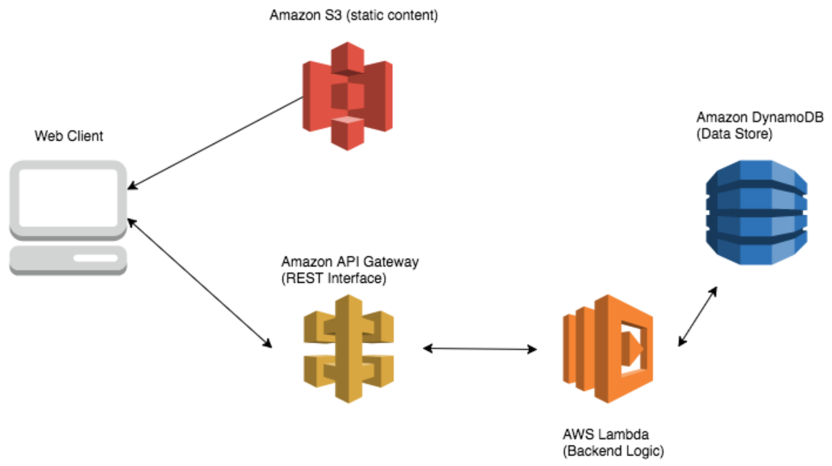
Maintainability requirement

All deployments to the production environment will be documented and version controlled. This will give the development team the option to roll back to previous deployments if an operational error was to occur. Therefore, the time taken to resume a fully operational service after failure should be less than six hours.

Portability requirement

The MCS web application will be developed to be mobile responsive. MCS should be usable on all devices

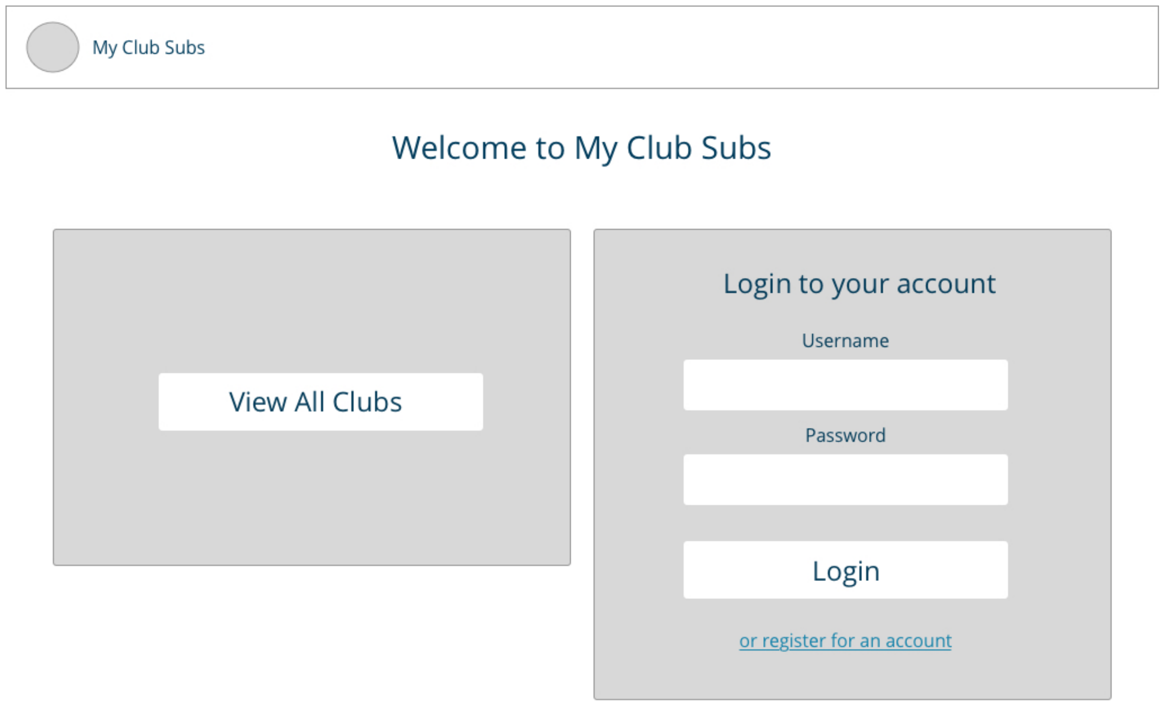
Design and Architecture

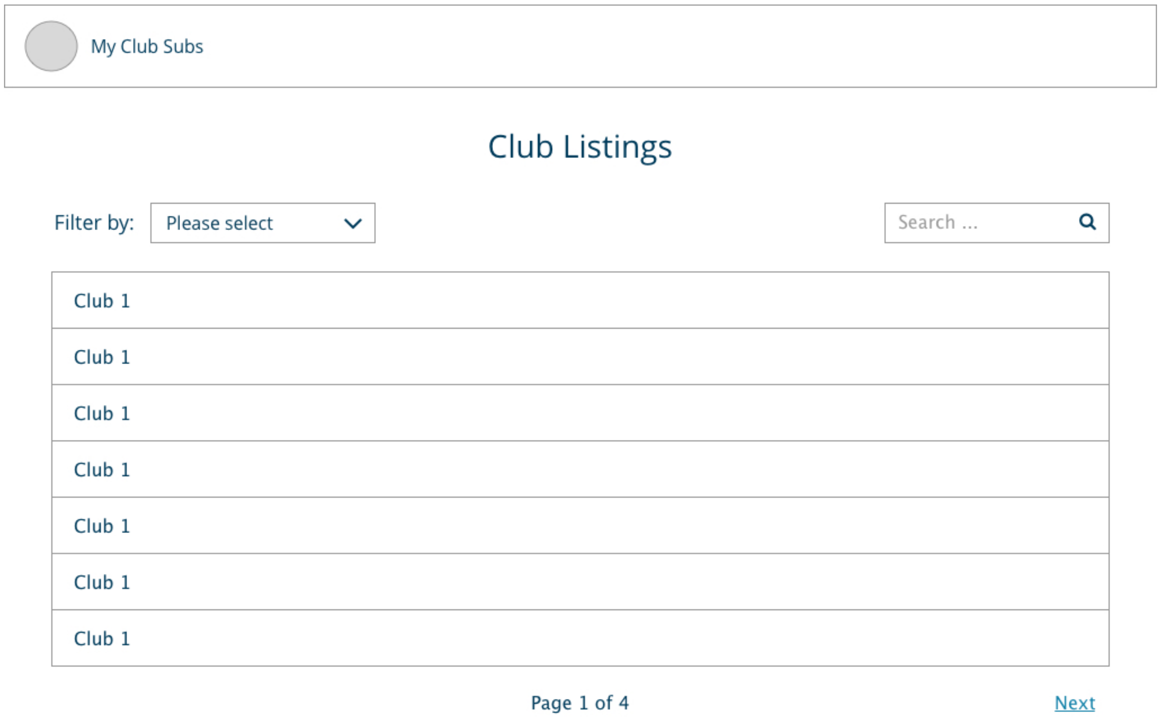

Amazon S3 (static content)
Web Client
Amazon DynamoDB (Data Store)
Amazon API Gateway (REST Interface)
AWS Lambda (Backend Logic)

Implementation

MyClubSubs will be implemented using AWS lambda functions developed in python. These functions will call data from the database and complete all the backend logic. The frontend will be display using static content form the S3 buckets. I will use the React JavaScript library to display that data on the front end retrieved by the lambda function

Graphical User Interface (GUI) Layout





My Club Subs

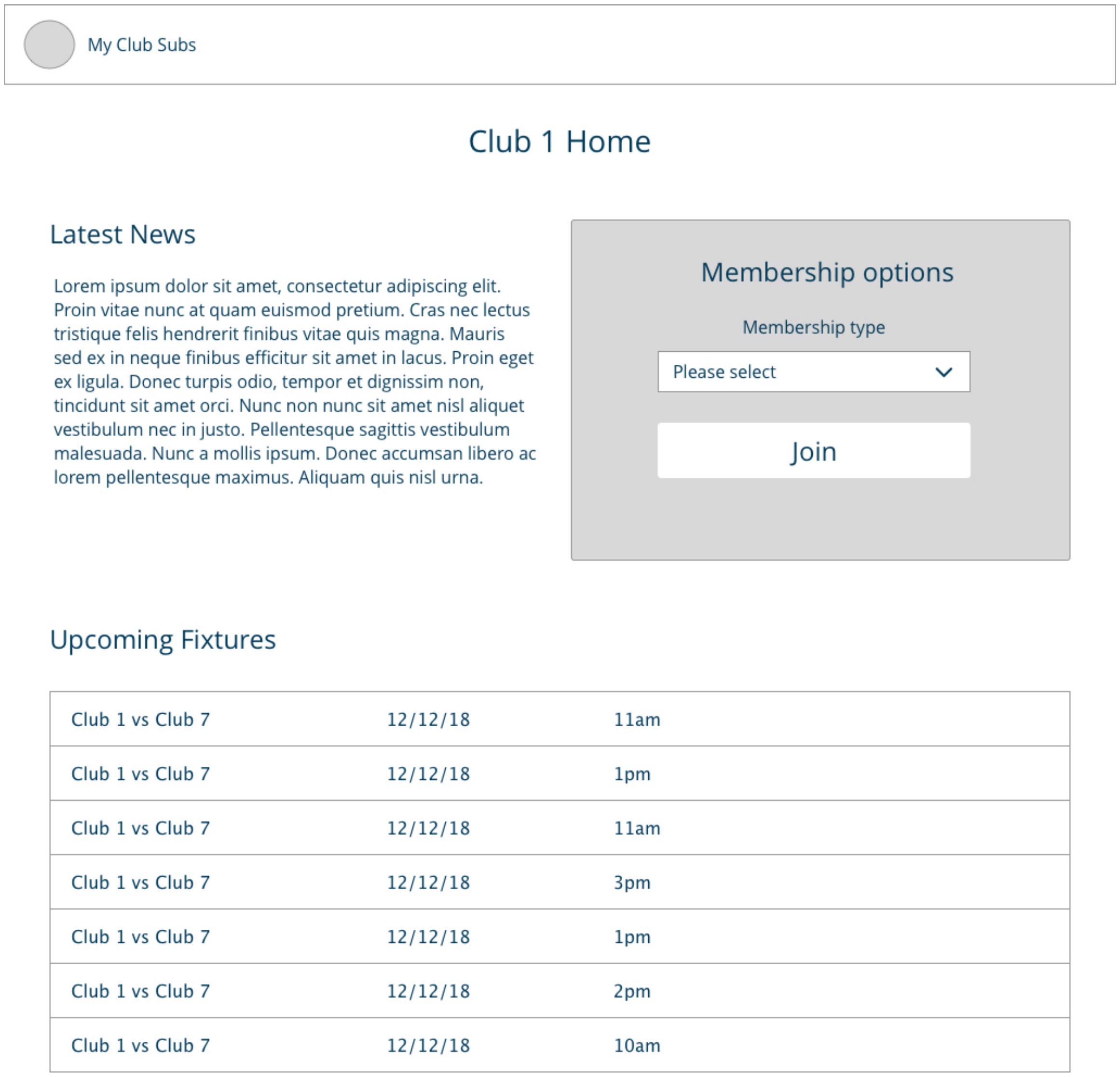
Club Listings

Filter by: Please select

Search ...

Club 1
Club 1
Club 1
Club 1
Club 1
Club 1
Club 1

Page 1 of 4 Next



My Club Subs

Club 1 Home

Latest News

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin vitae nunc at quam euismod pretium. Cras nec lectus tristique felis hendrerit finibus vitae quis magna. Mauris sed ex in neque finibus efficitur sit amet in lacus. Proin eget ex ligula. Donec turpis odio, tempor et dignissim non, tincidunt sit amet orci. Nunc non nunc sit amet nisl aliquet vestibulum nec in justo. Pellentesque sagittis vestibulum malesuada. Nunc a mollis ipsum. Donec accumsan libero ac lorem pellentesque maximus. Aliquam quis nisl urna.

Membership options

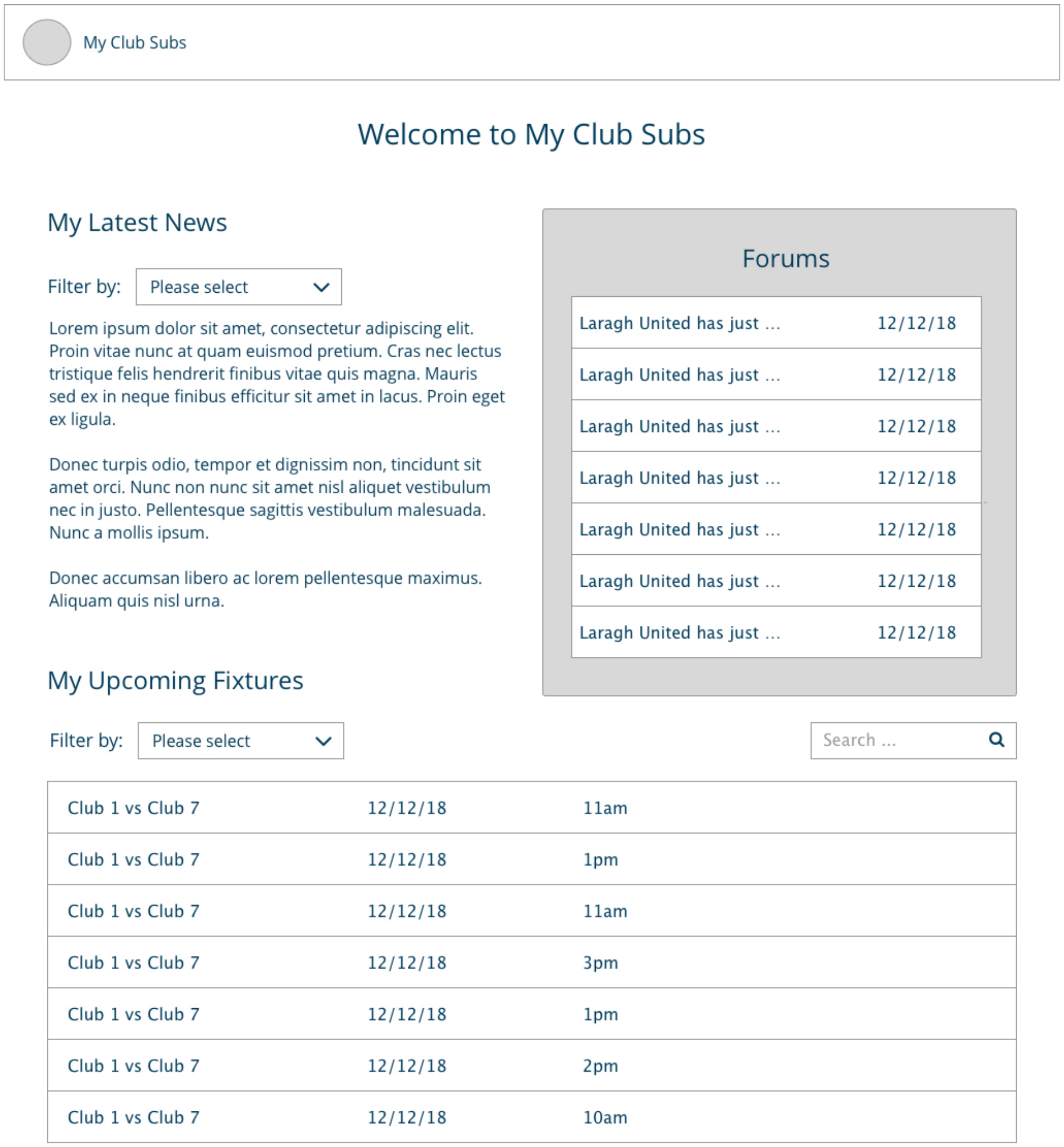
Membership type

Please select

Join

Upcoming Fixtures

Club 1 vs Club 7	12/12/18	11am
Club 1 vs Club 7	12/12/18	1pm
Club 1 vs Club 7	12/12/18	11am
Club 1 vs Club 7	12/12/18	3pm
Club 1 vs Club 7	12/12/18	1pm
Club 1 vs Club 7	12/12/18	2pm
Club 1 vs Club 7	12/12/18	10am



My Club Subs

Welcome to My Club Subs

My Latest News

Filter by: Please select

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin vitae nunc at quam euismod pretium. Cras nec lectus tristique felis hendrerit finibus vitae quis magna. Mauris sed ex in neque finibus efficitur sit amet in lacus. Proin eget ex ligula.

Donec turpis odio, tempor et dignissim non, tincidunt sit amet orci. Nunc non nunc sit amet nisl aliquet vestibulum nec in justo. Pellentesque sagittis vestibulum malesuada. Nunc a mollis ipsum.

Donec accumsan libero ac lorem pellentesque maximus. Aliquam quis nisl urna.

Forums

Laragh United has just ...	12/12/18
Laragh United has just ...	12/12/18
Laragh United has just ...	12/12/18
Laragh United has just ...	12/12/18
Laragh United has just ...	12/12/18
Laragh United has just ...	12/12/18
Laragh United has just ...	12/12/18

My Upcoming Fixtures

Filter by: Please select

Search ...

Club 1 vs Club 7	12/12/18	11am
Club 1 vs Club 7	12/12/18	1pm
Club 1 vs Club 7	12/12/18	11am
Club 1 vs Club 7	12/12/18	3pm
Club 1 vs Club 7	12/12/18	1pm
Club 1 vs Club 7	12/12/18	2pm
Club 1 vs Club 7	12/12/18	10am

Testing

There will be a number of test phases completed during the course of the development.

These tests include

- Integration testing
- Functional testing
- Stress testing
- Performance testing
- UAT testing
- User testing

Evaluation

MyClubSubs will provide the ultimate solution to club members and administrators around the country. The main functional requirements are very realistic and should be completed after a short period of development. Security will need to be given high priority as there will be personal data used in the application. This will add a lot of complexity to the development stages to ensure the application is secure. After the functional requirements are complete there will more complex features added to the application.

MyClubSubs has huge commercial potential as I feel it would be quickly adopted around the country by all types of clubs.

Appendix

Project Proposal

Objectives

This project aims to address current digital gaps in the running and administration of a volunteer club or organization. The objective is to create a web application to provide club administrators a time saving and cost-efficient platform to manage the day-to-day running of the club. The secondary objective is to provide club members a platform where they can manage their interactions with all the clubs they hold membership.

The aims will be met by providing the following services

- Club Admin services
    - Set memberships types and fees
    - Set up groups
    - Distribute email and text messages to all club members or specific groups.
    - Create membership payment reminders and instalment plans
    - Upload fixtures via csv to be displayed on a calendar in the club website
    - Shop for club merchandise, tickets and fundraiser tickets eg. Weekly lotto
    - View income dashboards from membership fees and shop sales
- Club member services
    - View all clubs in central dashboard

- All upcoming fixtures for the clubs you hold membership will be displayed in one calendar
- Google maps to provide directions to away games
- Option to add other family members to user account
- Forum to chat with other members of the club/group

Background

I have been involved in my local GAA club since I was 5 years old. For many years, I was an active playing member. In 2015, I was given the role of club registrar. Club registrar is a role recently introduced by the GAA. The role of the club registrar is to collect membership for every member of the club. My predecessor passed me on sheet of paper with a list of names and addresses for all the club members of the previous year.

The club registrar has no system to work on – in the majority cases all the work is manual. This includes going to training sessions and organized registration nights collecting memberships. The membership fees would be collected in cash and would need to be lodged into a bank account. A written receipt would be given by the registrar to the member and the registrar would keep a hard copy for themselves.

Setting up email distribution groups and SMS distribution groups require a lot of work and management.

The situation is the same in soccer clubs around the country.

I believe this area of amateur sports has been very slow to adopt new technology to make life easier and free up time for club administrators.

Technical Approach

Firstly, I will scope out all the requirements for the web application and document them in the Requirement Specifications. The next stage will be to build out the front end/UI for the mid-point prototype. The UI will use dummy data. The approach to develop the UI first will provide added insight into what data I will need to pull from the database and other web services.

A database will be used to store all member and club data. The web application will be built using a Serverless web architecture. This will require functions to be created that when triggered on the front end, will then run and retrieve data from the database. The function will return the data in JSON format. I will then then parse and present the data on the front end using a JavaScript library. There will be several other services required to perform specific tasks.

Special Resources Required

There will be a number of research resources required for the development of this application. There are multiple AWS services being used in this application therefore I will be using their online guideline documentation to aid the development.

Initially, I will meet with club administrators and members to gather requirements for the features in the web application. When the prototype is complete, I will get the same users to test the application and provide some feedback. The application will go through multiple rounds of testing. There will be many people required for this user testing.

Project Plan

Project plan attached.

Technical Details

The web application will be developed using a serverless architecture.

An Amazon S3 bucket will be used to host all the static content. I will use Amazon RDS for the database. Lambda functions will be used to request data from the database and return it to the front end. React combined with HTML and CSS will be used to parse the JSON data returned to the front end and present the data on the page to the user.

For my application the following tools, applications and languages will be used.

- Amazon S3
- Amazon RDS
- Amazon Lambda
- Amazon SQS
- HTML
- CSS
- React
- JS
- SQL
- Python
- Google Maps API

Evaluation

MyClubSubs will provide the ultimate solution to club members and administrators around the country. The main functional requirements are very realistic and should be completed after a short period of development. Security will need to be given high priority as there will be personal data used in the application. This will add a lot of complexity to the development stages to ensure the application is secure. After the functional requirements are complete there will more complex features added to the application.

MyClubSubs has huge commercial potential as I feel it would be quickly adopted around the country by all types of clubs.

Project Plan

attached