Software Engineering

Uni⭐Radar

Your Experience. Your Opinion.

UniRadar Project Report

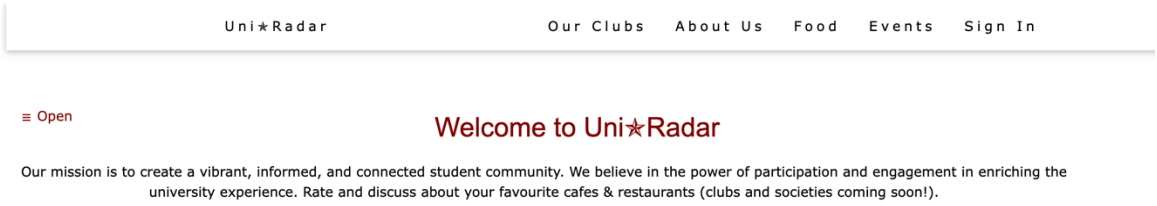
Prepared by
George Chukwuma Anayo-Ezikeoha ~ 22358526
Ciaran Daly ~ 21381943
Habel Sebastian ~ 22342896

20th March 2024

Table of Contents

I. PROJECT DESCRIPTION

PROJECT OVERVIEW



UniRadar is a comprehensive web application designed to serve the student community at the University of Galway by providing a centralized platform for accessing information about events, societies, and clubs on campus. The primary objective of UniRadar is to enhance the collective experience for the students by offering easy access to relevant information and fostering community engagement.

Key Features:

1. **Event Exploration**: Users can explore a diverse range of upcoming events hosted by different departments, student organizations, and clubs within the university. From academic seminars to social gatherings, UniRadar offers a comprehensive overview of campus activities.

2. **Society and Club Directory:** UniRadar maintains a directory of student-run societies and clubs, allowing users to discover opportunities to join or participate in. activities aligned with their interests. The platform categorizes societies and clubs based on various interest areas, facilitating easy navigation.

3. **Ratings and Reviews:** UniRadar enables users to provide feedback and ratings for events, societies, and clubs they have attended or engaged with. This feature allows users to share their experiences, offer insights, and help others make informed decisions about their involvements on campus.

4. **Community Engagement:** UniRadar promotes community engagement by facilitating communication and collaboration among students, organizers, and members of various campus groups. Users can interact through chat features, fostering connections and a sense of belonging within the University of Galway community.

Benefits:

- Enhanced Accessibility: UniRadar provides students with a centralized platform to access information about campus activities and organizations, streamlining the process of discovering and participating in social events, clubs, and societal gatherings.
- Informed Decision-Making: the rating and review system empowers users to make informed decisions about their involvement in events, societies, and clubs based on peer feedback and recommendations.
- Community Building: UniRadar contributes to the development of a vibrant and inclusive University of Galway community by promoting engagement and interaction among students and campus groups.

2. Purpose of the Project

The purpose of the UniRadar project is to serve as a comprehensive platform catering specifically to the needs of University of Galway students. Through UniRadar, students can access a centralized source for all campus-related information,

streamlining their university experience. Furthermore, UniRadar aims to facilitate knowledge-sharing among students by allowing them to provide personalized reviews and comments based on their individual experiences. By leveraging these insights, UniRadar not only enhances decision-making processes but also fosters a sense of community among students.

Additionally, UniRadar serves as a valuable resource for event and club organizers within the University of Galway community. By providing a platform where organizers can view and respond to feedback, UniRadar contributes to continuous improvements of campus events and club experiences. This ensures that the activities organized within the university are tailored to meet the preferences and expectations of the student body, ultimately enhancing overall satisfaction and engagement.

2a. Goal of the Project

The goal of the UniRadar project is to implement a robust web application that encompasses all the features, thereby offering the best possible campus-centric experience for the University of Galway students. By prioritizing usability, accessibility, and functionality, our aim is to create a user-friendly platform that meets the diverse needs of the student population.

Once we have established UniRadar as the premier campus-centric app for University of Galway, our next objective is to expand its reach beyond the confines of the university campus. With a solid foundation in place, we intend to broaden our geographical radar to encompass the entirety of Galway. This expansion will further solidify UniRadar's position as the go-to platform for accessing comprehensive information and facilitating community engagement within the broader Galway area.

3. Scope of Work

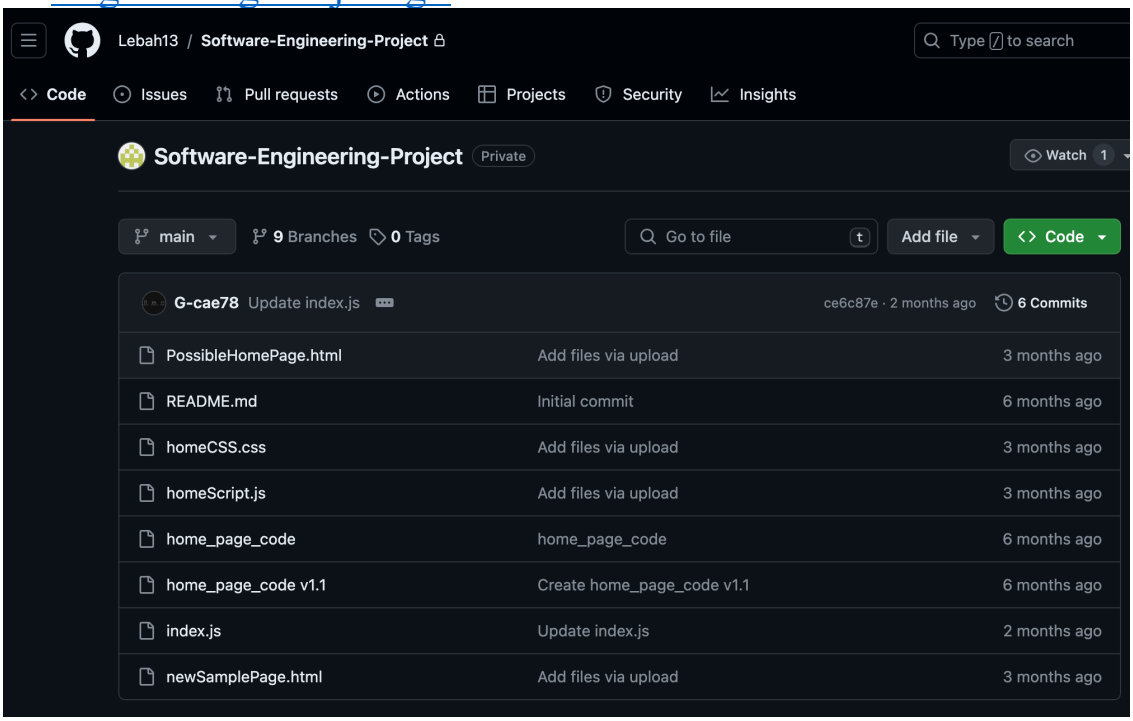
3a. Project Timeline and Work Partitioning

Task Description	Start Date	Finish Date	Duration	Status	Task Allocation
Define Project Objective	3rd October 2023	10th October 2023	1 week	Finished	Group
Layout web app functionality and UI design	20h October 2023	23rd December 2023	2 Days	Finished	Group
Design initial UI using HTML for hello world day	23rd December 2023	2nd December 2023	1 Week	Finished	Ciaran and Habel
Set up GitHub to share and store code	3rd December 2023	4th December 2023	1 Day	Finished	Habel
Convert HTML pages to Vue components	17th December 2023	22nd January 2024	1 Month	Finished	Ciaran
Code up Firebase Functions	20th December 2023	23rd March 2024	3 Months	Finished	George
Set up Firebase Database	22nd February 2024	1st March 2024	1 Week	Finished	George
Set up communication between front-end and back-end	17th March 2023	23rd March 2024	1 Week	Finished	George & Ciaran
Build Log in and Registration pages	17th March 2024	19th March 2024	2 Days	Finished	Ciaran
Set up Firebase Authentication for log in and Registration	19th March 2024	22nd March 2024	3 Days	Finished	George
Deploy Back-End to firebase	20th March 2024	20th March 2024	1 Hour	Finished	George
Deploy Front-	26th March	26th March	3 Hours	Finished	Ciaran

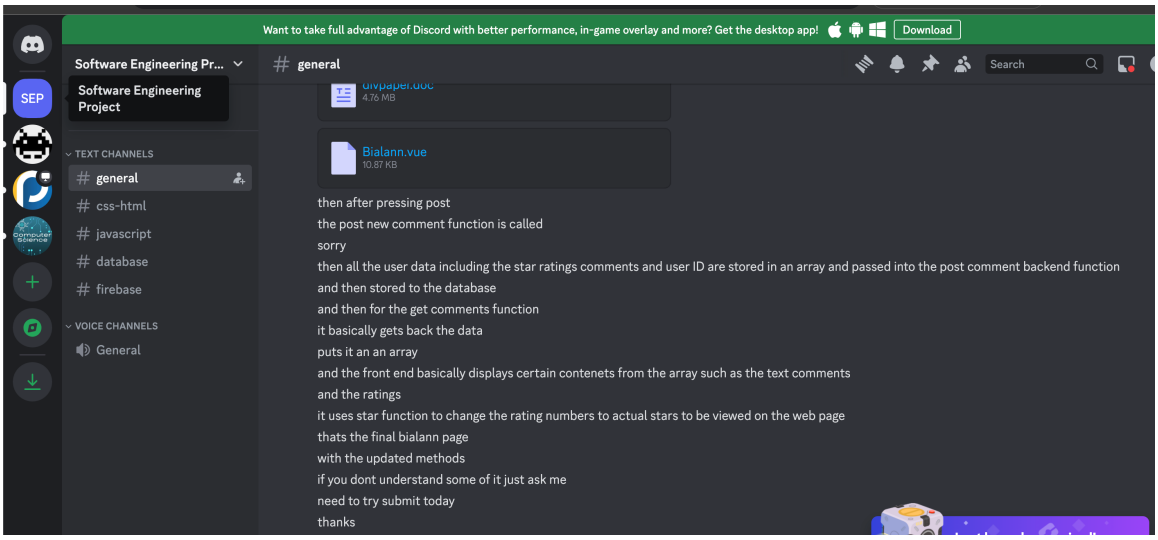
End to firebase	2024	2024			
Hosting App on Firebase	27th March 2024	27Th March 2024	4 Hours	Finished	George
Writing Project Report	20th March, 2024	1st April 2024	2 Weeks	Finished	Group

3c. Project Tools

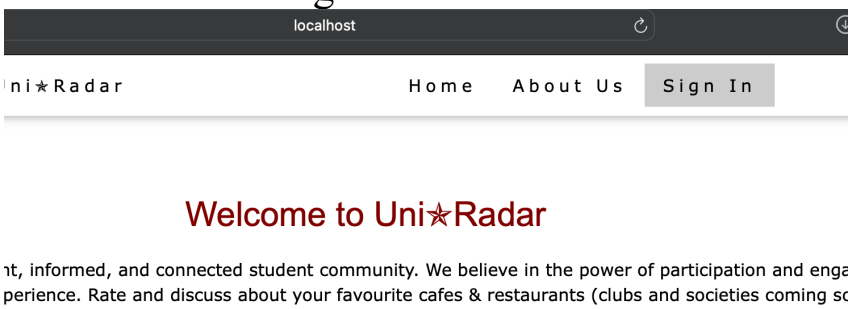
- GitHub- Storing changes to code and for easy access to code.
- GitHub Url- https://github.com/Lebah13/Software-Engineering-Project.git



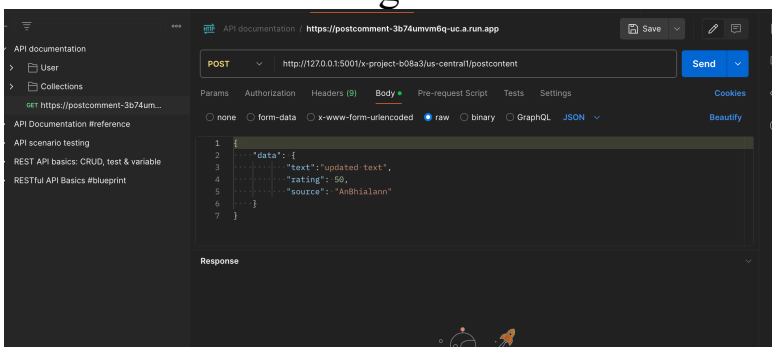
- Discord- Setting up meetings and allocating tasks.
- Discord invite link- https://discord.gg/cAafDDG9.



- Jira- Initial app for work partitioning and timelines.
- Notion- Final tool used for work partitioning and monitoring progress.
- LocalHost- Testing Front-End.



- Postman- Testing Back-End.



3d. Project Coding Stack

Front-End- CSS, Vue.js, HTML
Back-End- Node.js,
Database- For storing data Firebase Firestore was used.
The web app was deployed onto Firebase and hosted on their server.

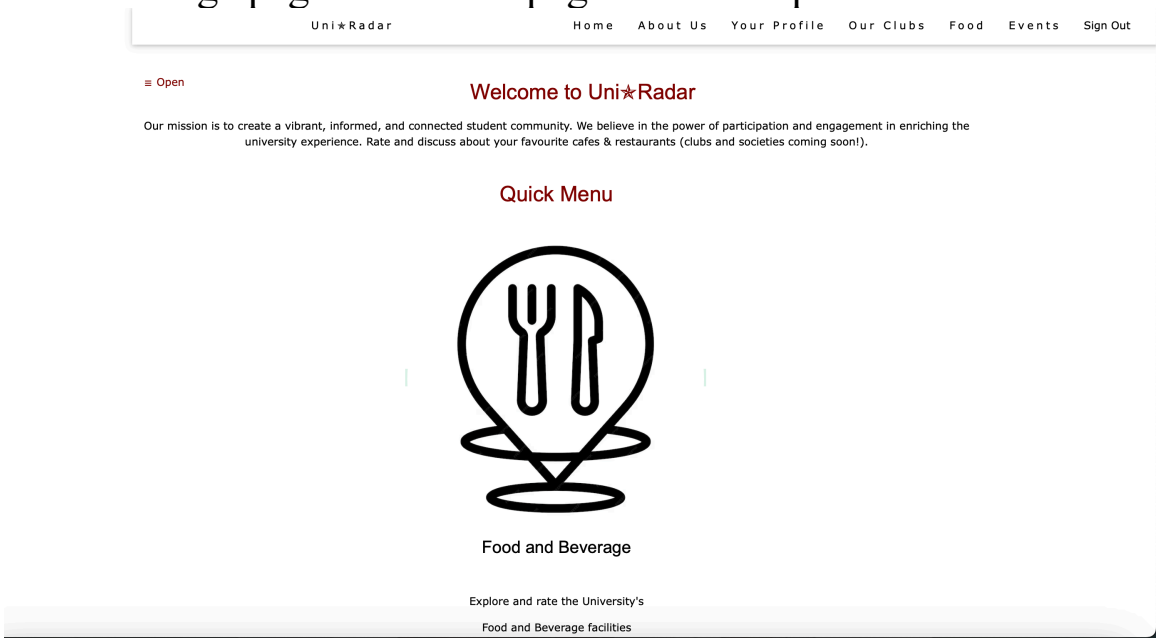
II. Requirements

6. Look and Feel Requirements

6a. User Interface

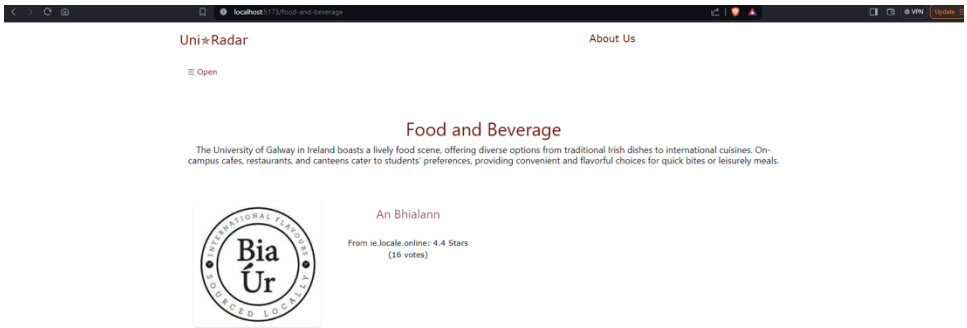
The user interface design of UniRadar was developed with simplicity and responsiveness as its core values. It utilizes a single page router with VueJs to integrate seamless loaded between the projects various components/pages.
The

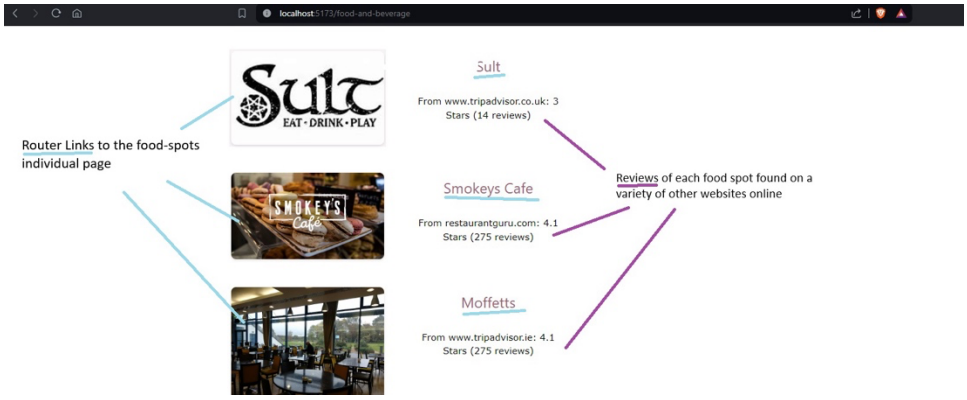
Home Page (Home.vue)
The home page is designed to be a simple and sleek landing page. It contains a short and informative introduction to users to the web app's purpose, and a menu below containing router links to Food and Beverage pages and other pages in development.



Food and Beverage Landing Page (FoodAndBeverage.vue)
Like the home page, the Food and Beverage Landing Page is designed with simplicity and user friendliness as a priority. It features a simple, single list of the food and beverage spots on

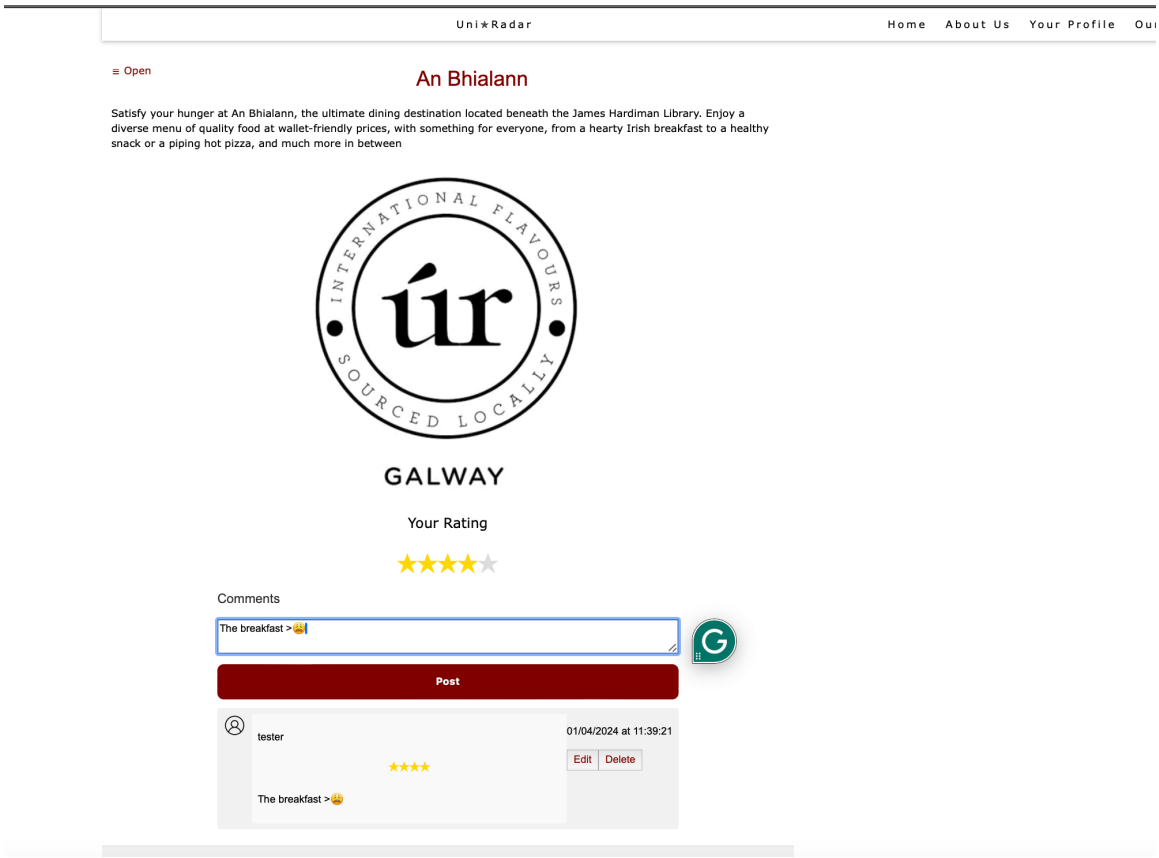
campus, with a router link to each individual food spot page, and ratings of that food spot from other online sites.





Individual Food and Beverage pages

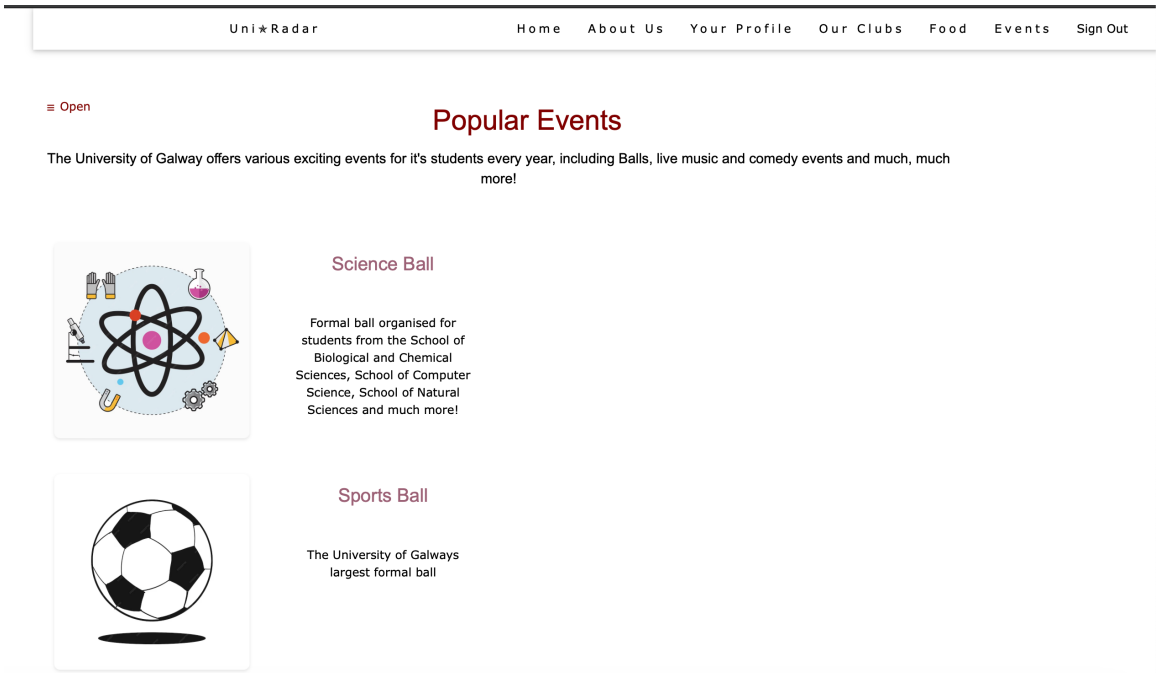
These pages allow the user to rate and comment on food spots. When a user rates a spot and comments, their rating will be displayed alongside their comment.



Comment Section: UniRadar uses Bootstrap Buttons for the 'post' button of the comment section, specifically *.btn-primary*, to achieve a modern and interactive design, with edited CSS (in homeCSS.css) to accommodate UniRadar's colour scheme.

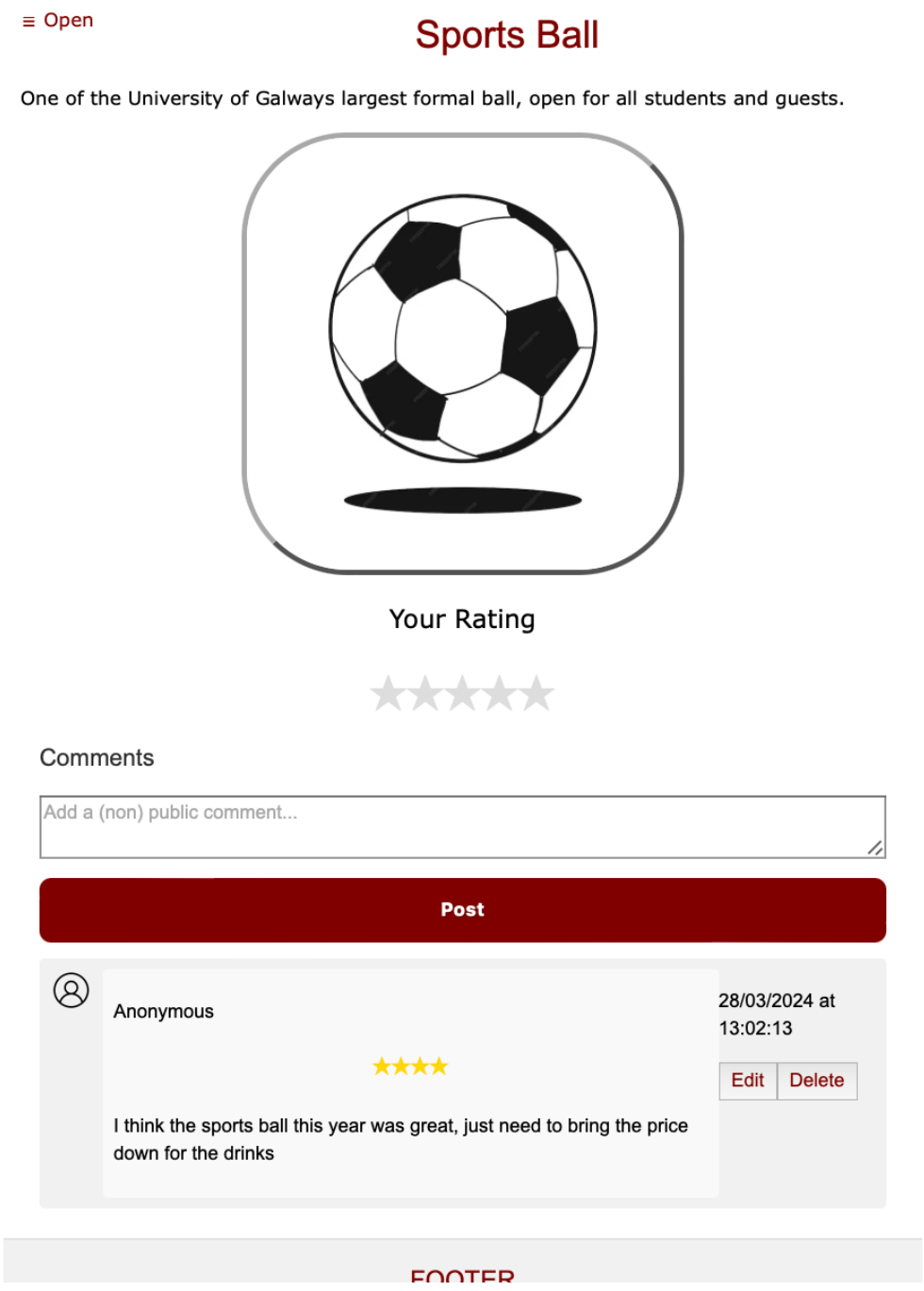
Event Landing Page (Events.vue)

The Events page works very similar to the Food and Beverage Landing Page. It contains links that route to 'popular events' such as university balls and their individual pages.



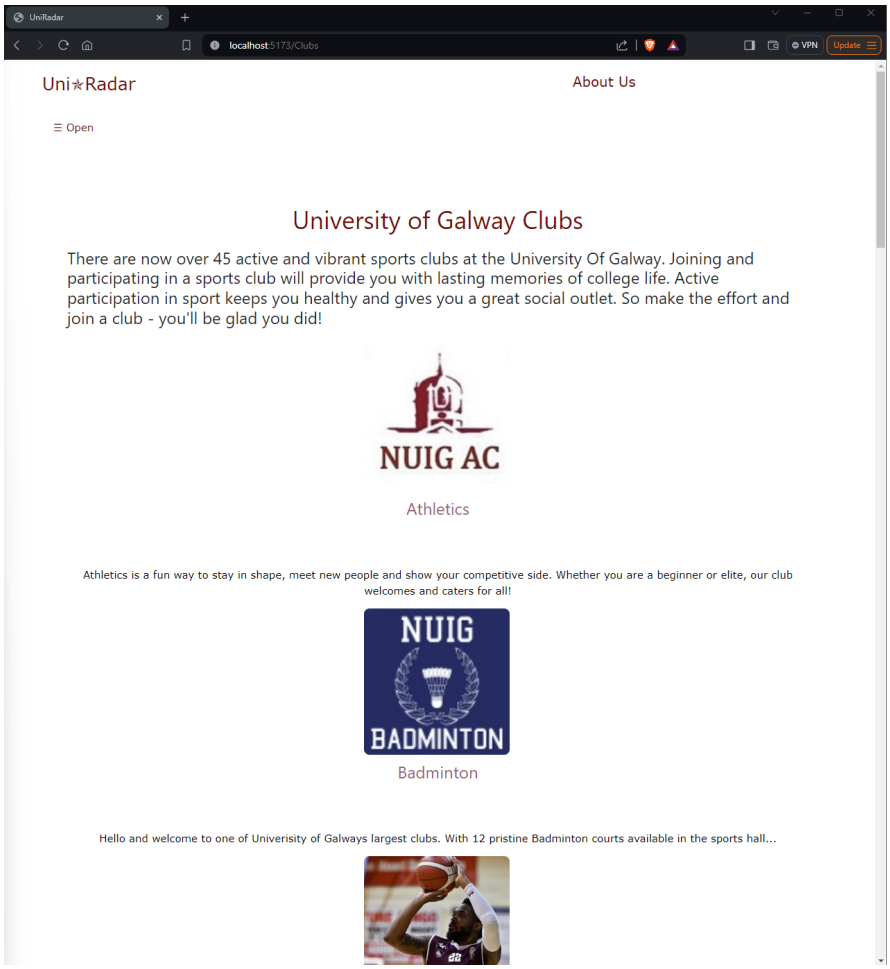
Individual Events pages

Functionality works the same as the Individual Food and Beverage pages where a user inputs their comments and ratings. When the post button is pressed the user's comments are sent to our database.



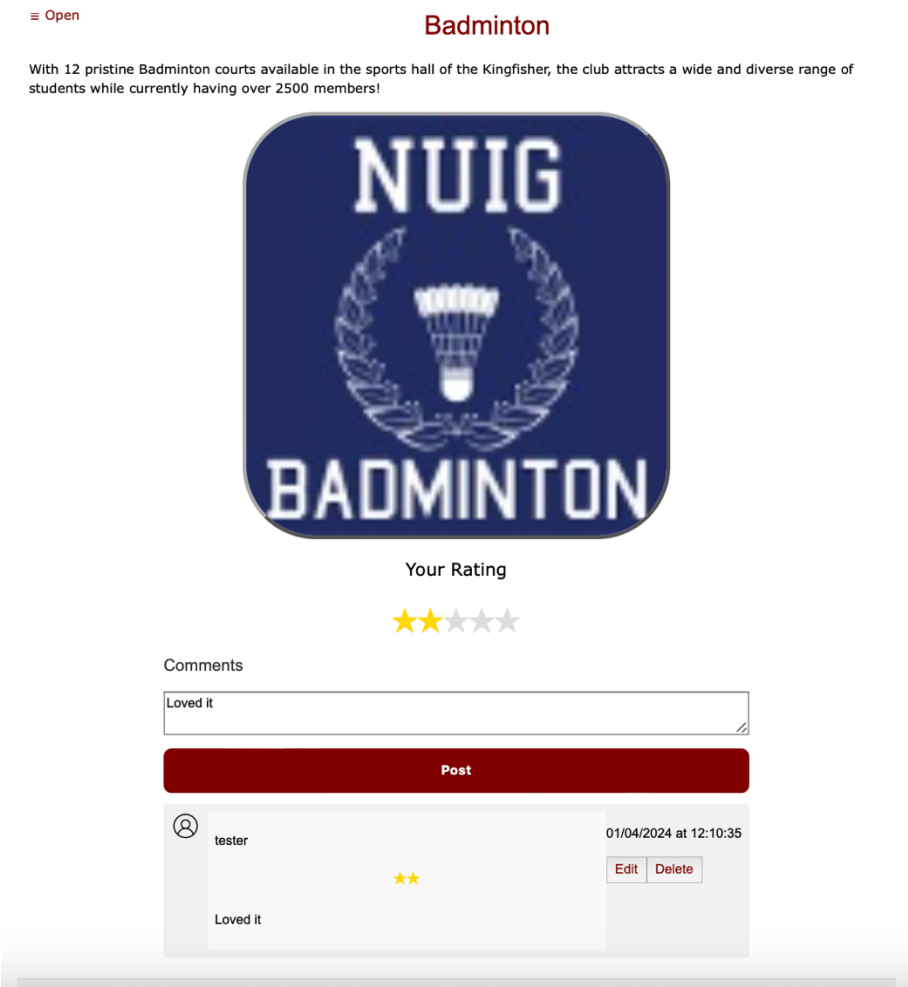
Clubs Landing Page (ClubsPage.vue)

The Events page works very similar to the Food and Beverage Landing Page. It contains links to clubs in the university, a brief description, and their individual pages.



Individual Clubs pages

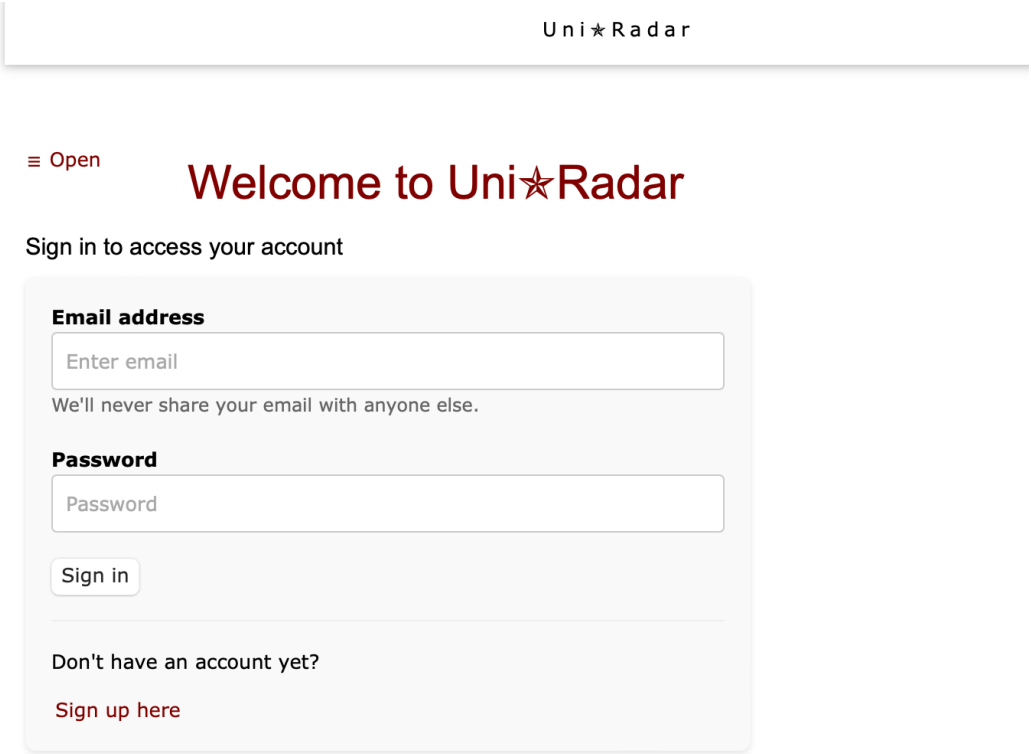
The functionality of the club pages works the same as the functionality of the individual food pages.

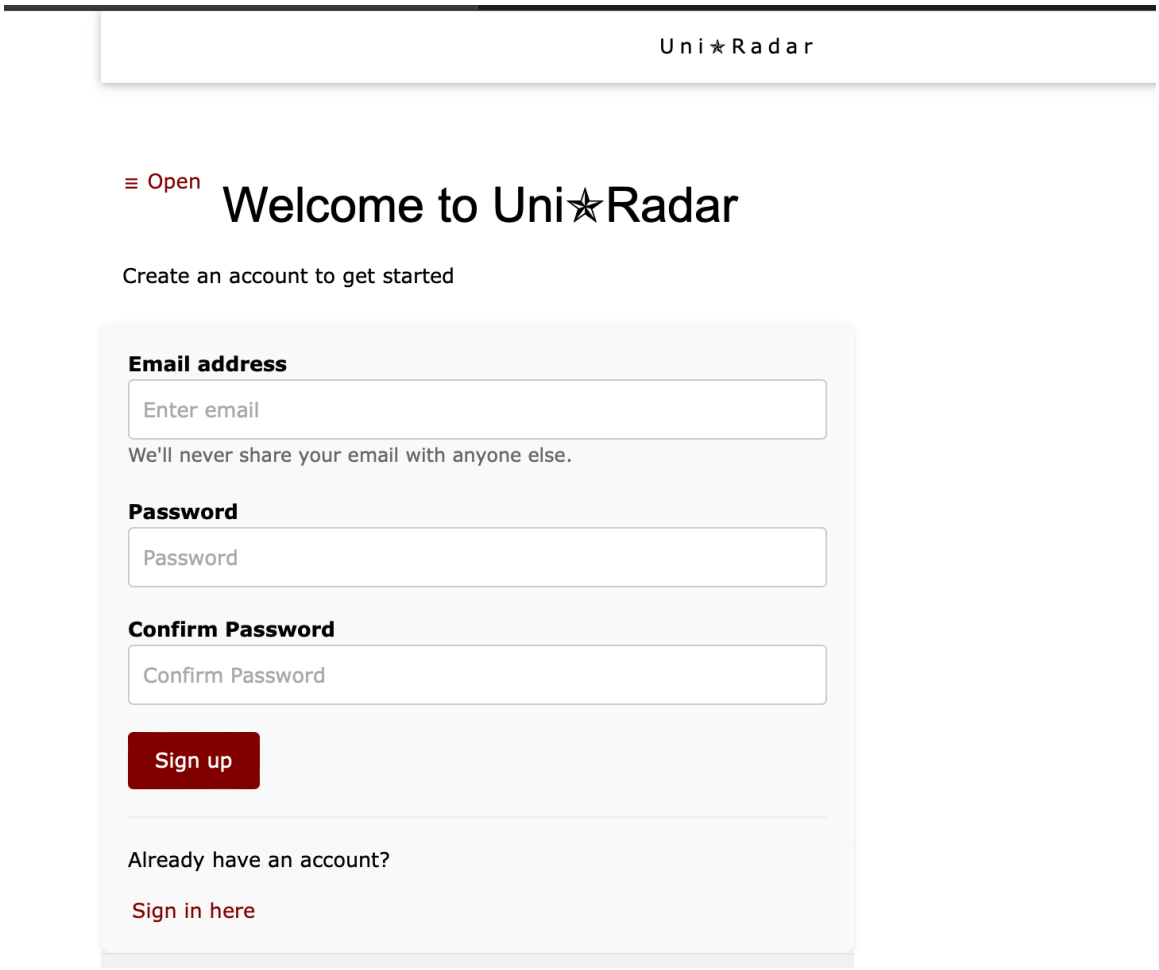


Registration Page (Registration.vue) and Sign in Page

Our sign-in page provides a seamless experience for returning users and newcomers alike. If you're already part of our community, you can simply sign into your account by entering your details and logging in. If you do not have an account you can

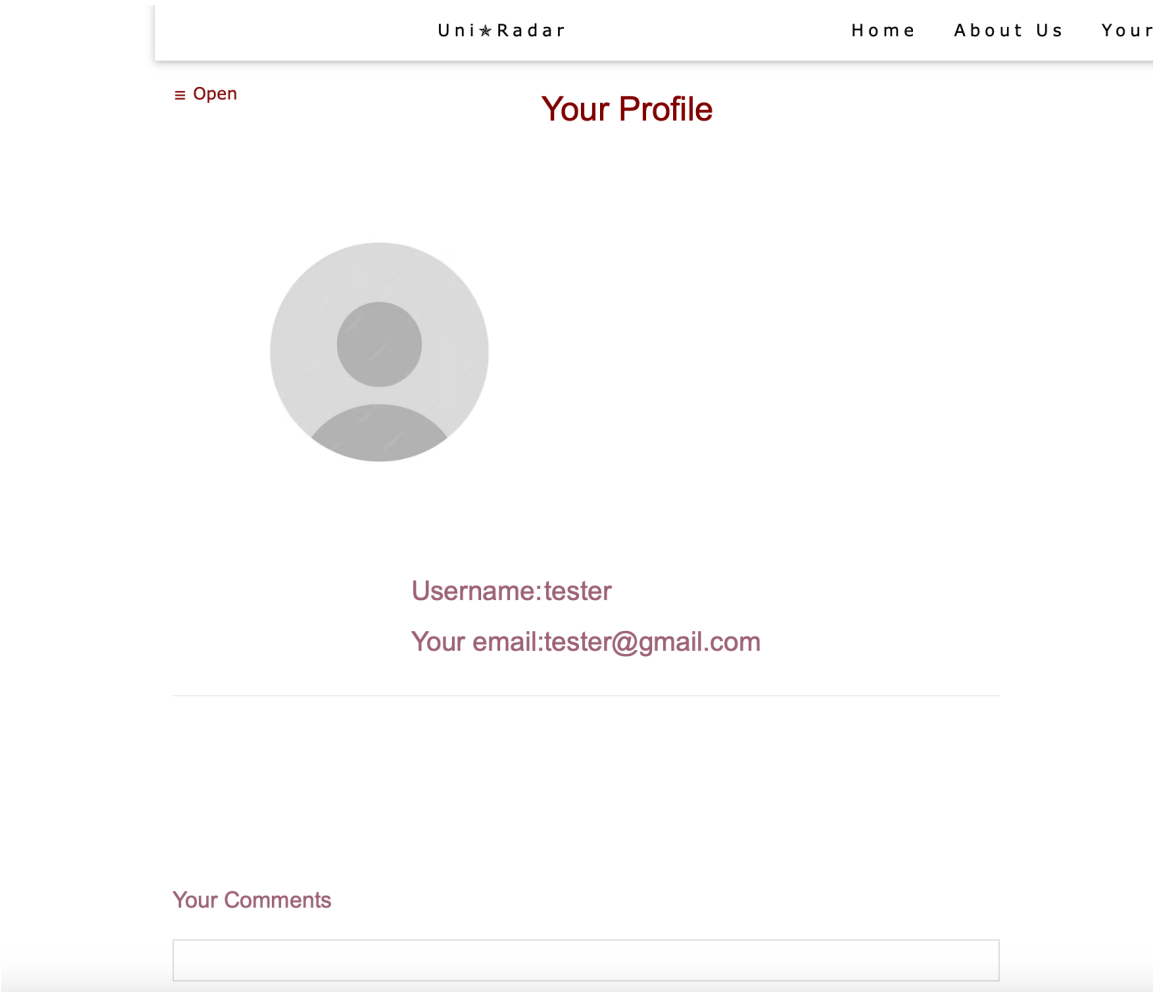
simply create one on the spot by clicking ‘don’t have an account’ which sends you to the registration page where you can sign up straight away.





Profile Page (Profile.vue)

The Profile page lets the user, after registration, view their username, email address, and all the comments and ratings they have made across UniRadar.



CSS Styles (homeCSS.css and main.css)

UniRadar's styles combines a combination of original CSS styles, in combination with W3.CSS, a free to use style sheet found on [www.w3schools.com.](http://www.w3schools.com)

W3 CSS offers a 'modern, responsive, mobile first CSS framework'[1]

[1] W3 Schools. What is W3.CSS?. [Online]. w.w3schools.com. Available at: https://www.w3schools.com/whatis/whatis_w3css.asp [Accessed 23 March 2024].

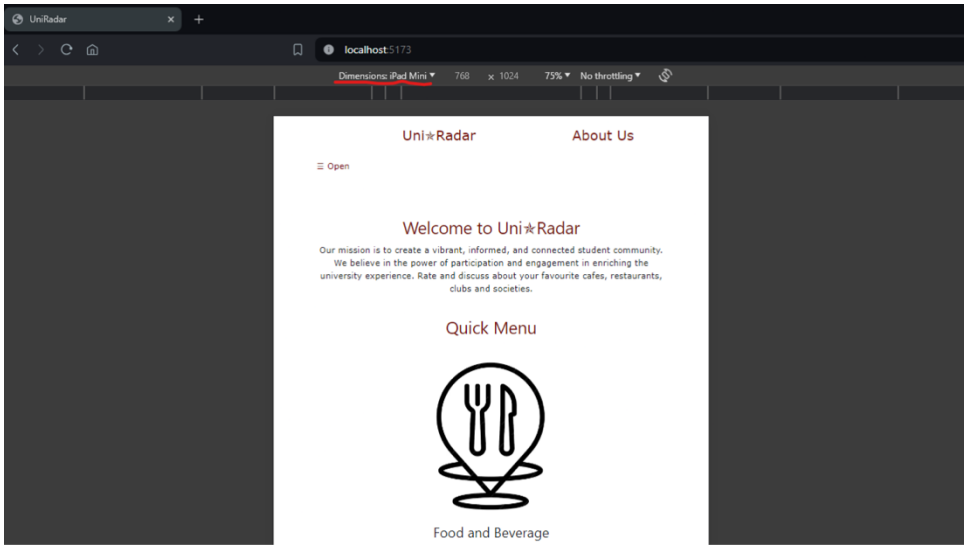
Media Queries (CSS):
To handle the application of UniRadar's CSS styles on various device's media types and other features, such as orientation, resolution, browser viewport width or height etc., the implementation of media queries was used in addition to the responsive W3 CSS implementation.

Media Queries in main.css:

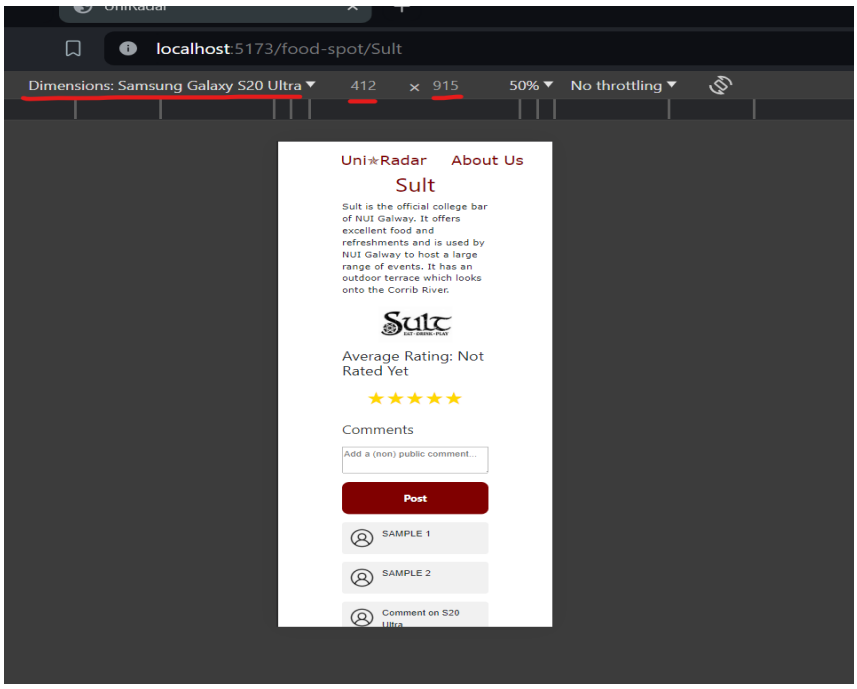
```
@media (hover: hover) {
  a:hover {
    background-color: hsla(325, 100%, 25%, 0.2);
  }
}

@media (min-width: 1024px) {
  body {
    display: flex;
    place-items: center;
  }
```

Example UI for the dimensions of:

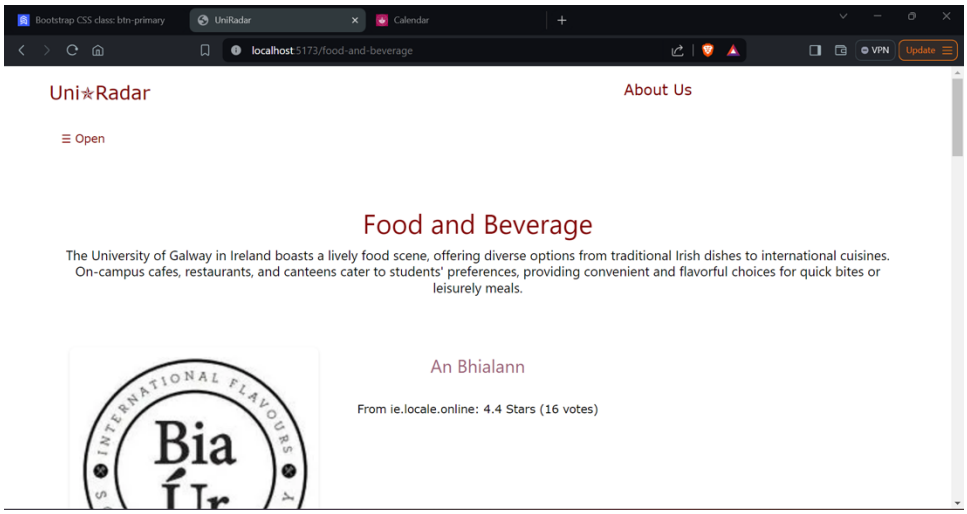
iPad Mini:



Samsung Galaxy S20 Ultra:



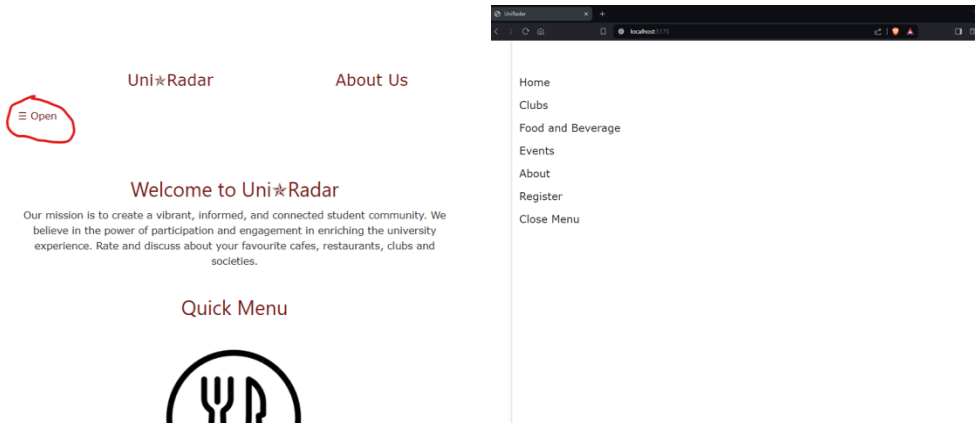
ThinkPad T480 on Brave Browser:



Sidebar Navigation (Sidebar.vue)

For easy navigation on any page of the web app, a sidebar was added. It offers a sleek transition when clicked to show or close the sidebar navigation menu.

The sidebar, like the links found in the Food and Beverage Landing Page and the top menu, utilizes the router (index.js) for fast, single page loading.




```vue
<div :class="['backdrop', { 'show': showSidebar }]" @click="toggleSidebar(false)"></div>

<!-- Sidebar navigation -->
<nav :class="['w3-sidebar', 'w3-bar-block', 'w3-card', 'w3-top', 'w3-xlarge', 'w3-animate-left', { 'w3-show': showSidebar }]" id
  <a href="javascript:void(0)" @click="toggleSidebar(false)" class="w3-bar-item w3-button" style="color: white;">Close Menu</a>
  <router-link :to="`/`" @click="toggleSidebar(false)" class="w3-bar-item w3-button" style="padding-top: 25px;">Home</router-lin
  <router-link :to="`/food-and-beverage`" @click="toggleSidebar(false)" class="w3-bar-item w3-button">Food and Beverage</router-
  <router-link :to="`/about`" @click="toggleSidebar(false)" class="w3-bar-item w3-button">About</router-link>
  <router-link :to="`/registration`" @click="toggleSidebar(false)" class="w3-bar-item w3-button">Register</router-link>
  <a href="javascript:void(0)" @click="toggleSidebar(false)" class="w3-bar-item w3-button">Close Menu</a>
</nav>

<!-- Sidebar toggle button -->
<div @click="toggleSidebar(true)" class="w3-button w3-padding-16 w3-left" style="color: #800000;">☰ Open</div>
</div>
/template>

script>
xport default {
 name: 'Sidebar',
 data() {
  return {
   showSidebar: false, // Controls the visibility of the sidebar
  };
 },
 methods: {
oggleSidebar() {
 this.showSidebar = !this.showSidebar;
},

/script>

style scoped>
* Base styles for the sidebar, initially off-screen */
w3-sidebar {
 transform: translateX(-150%);
 transition: transform 0.75s ease;
 display: block !important;
}
```

The sidebar and all routes /links

(screenshot 2)

The button for users to press to toggle sidebar visibility

Toggles sidebar when clicked (@click)

7. Data Requirements

Thanks to the free cloud credits awarded to us by Google, we were able to use Firebase Firestore to manage and store essential data from our web application. The data includes the ratings, review text, source of data and timestamp from the various webpages for our events, societies and clubs as well as userId and user's email handle.

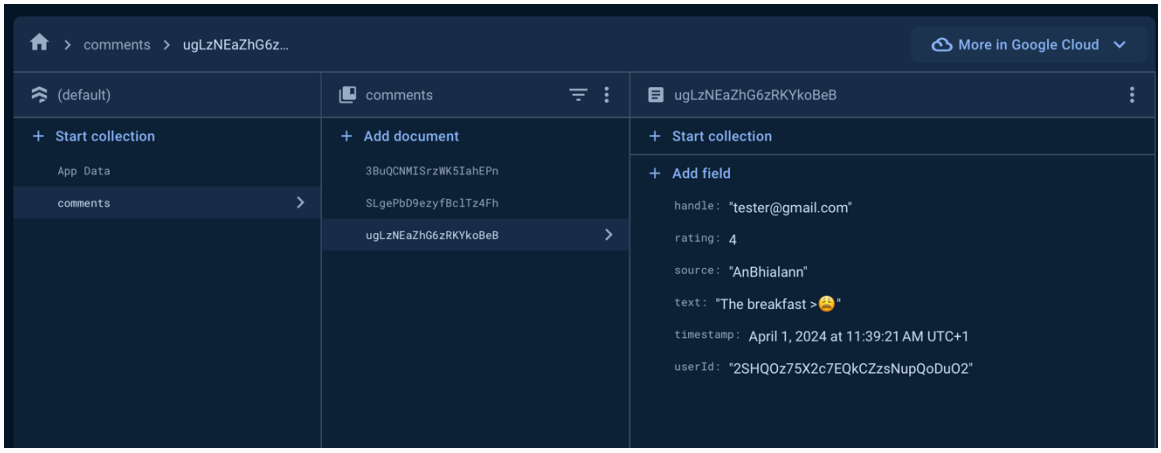


Fig: Structure of our database

User-Generated Data

As displayed above, the format for storage of user-generated data was ratings->source->text->timestamp.

- The ratings from our various web pages were stored as integers in our database but on displaying them on the web app they were looped over, and stars were displayed instead. This gave the front-end a better display and more aesthetic look.

- The source of the review and ratings were also stored in the database, the source of each review was used in the back-end code to filter out what page each review was sent from and display accordingly.
- The text just contained the user's review and would be displayed whenever the page was loaded.
- The timestamp was stored to aid in the sorting arrangement of the data when it being displayed, displaying the most recent comments first.

User Data

- Before accessing certain web pages, the user is asked to create an account with UniRadar. This account creation consists of the user inputting their password and email to be stored in our database.
- Whenever the user tries to sign in with the credentials, their input is checked against the ones stored in our database to ensure proper authentication.

Data Integrity and Security

- Firebase ensures the user's credentials are safe, it blocks us from viewing user passwords and only allows the user email to be seen.
- Firebase also stored the last log in of each user that has an account with UniRadar.

8. Functional Requirements

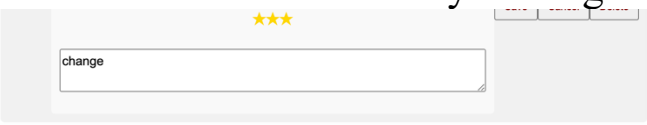
Back-End

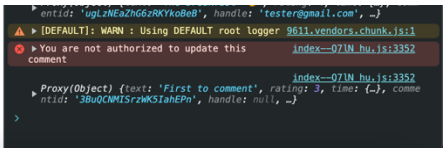
- All functions used in the back end are written in node.js. These functions are responsible for sending and receiving data to our Firestore cloud hence why we refer to them as cloud-based functions.
- For this project 5 serverless functions were developed and deployed to firebase.
- These functions run on the cloud whenever they are called from our web app when data is needed to be retrieved or stored.

➢ Getcontent- This function is responsible for returning all comments posted on our web app. This function then filters these comments based on the source page from which they were posted. The result of this filtration is the comments posted about the individual web page whether it be a society, event or café on our web app.

➢ GetUserComments- This function is very similar to the getContent function. Instead of filtering comments based on the source of the comment, this function filters comments based on the user ID currently logged in. This function is invoked when viewing the user's profile where all the comments the user has posted will be displayed.

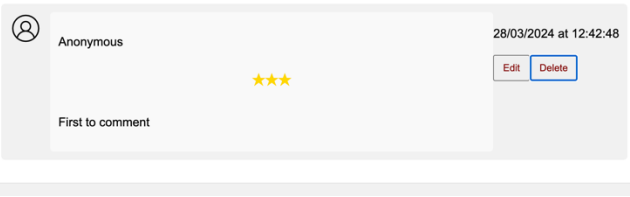
➢ UpdateComments- The update function is responsible for making changes to already posted comments. This functionality is only available to users with the same userID as the posted comment. This means a user can only edit their comment and no one else's, this makes the data from UniRadar more trustworthy and legitimate.





➢ DeleteComments- deleteComments like updatComments checks if the userId of the current user is the same as the userId stored in the comments data field. If it is then the

user's comment is deleted from UniRadar's database. Otherwise an error message is shown to the screen.



- PostComments- The post comments function creates an array of data from the front-end being inputted by the user and stores it in our database. Part of the data being stored include the userId and the source of the data, both of which could be used for data filtering.

```
data() {
  return {
    rating: 0,
    averageRating: 'In progress',
    newComment: '',
    comments: [],
    time: '',
    source:'Smokeys',
    handle:null,
    commentid:'',
    userId: null,
```

Fig: Array of data to be stored in database.

- These functions were used to implement CRUD functionality into our web app (Create, Read, Update and Delete).

Front-end Functional Requirements

For UniRadar's front-end, the main requirements that needed to be met were the development of a functional router, a working star rating and comment posting functions, and a responsive, easy-to-use and visually appealing user interface.

The router is an integral part of UniRadar, as we wish to build it as a Single Page Application (SPA). The implementation of a router will ensure that pages don't need to load every time the route changes, instead it essentially changes what components the user sees after everything is loaded, greatly improving a user's experience on UniRadar. In addition, the sidebar component will contain a router link to every page of UniRadar (apart from the individual event/clubs/food spots) and will be accessible on every page, allowing users to navigate through the web app quickly and easily.

The star rating and comment posting functions are a key part of UniRadar. As our goal is to allow users to rate and share opinions on various amenities on campus, this would not be achievable without such systems in place. The requirements to be met in building such systems include dynamically changing the HTML and CSS of the component being rated through JavaScript (like the posting of comments, the changing of start ratings shown when rated etc.), assigning the users rating dynamically when changed, storing the stars as a variable so it can be implemented by the backend team etc.

When a user makes a review, i.e. rating an event/clubs/food spot a certain amount stars, and adds text to their comment, this data is added to an array. Then when the user presses the ‘post’ button below the comment box, the post comment function is called. The post comment function then takes the data of that array and puts it on the database by calling the backend functions.
Other miscellaneous requirements include the development of components such as footer, top menu etc. As this will greatly reduce the constant repetition of code throughout all UniRadar’s components.

III. Testing

5a. Features Tested

Frontend Testing

Using the localhost address given, when running *npm run dev* in the VS code terminal, changes and developments made in the projects code were tested in real time.
Frontend testing was primarily focused on delivering well-designed, styled, user-friendly webpages which delivered on their respective functions. The following list describes the main problems encountered, and how I tested the problems to resolve these for the frontend.

1. Vue Conversion: After converting our original HTML, CSS and JavaScript files into separate Vue Components, assets etc., and correctly setting up firebase & firestore for the new Vue project, many problems arose.

1a. Firebase and Firestore: Issues arose to connecting the new project to Firebase and Firestore after conversion to Vue

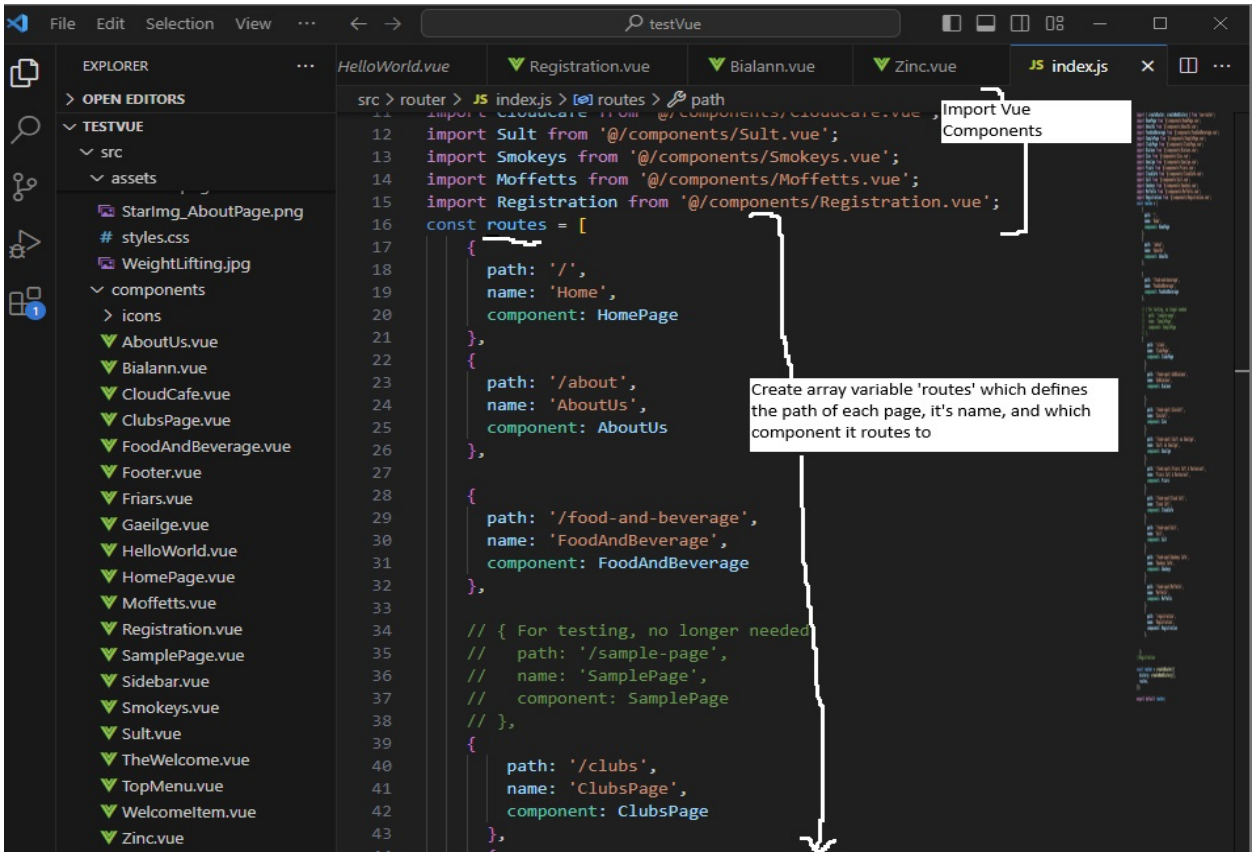
1b. Incorrect CSS, HTML, JS: The styles of the original project were not consistently seen throughout the new Vue project. For example, the background had changed from black to white, the headers and footer components had different styles to other components, and the images on all component pages were incorrectly styled, such as the image grid in FoodandBeverage.vue

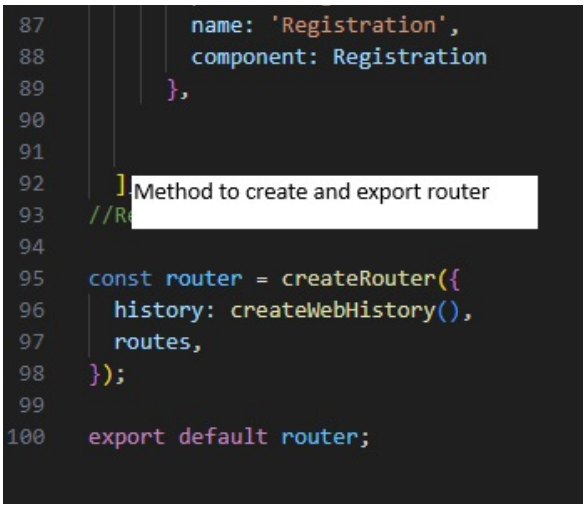
1c. The header component also had issues with formatting. The 'About Us' router link the upper right corner was barely visible as it was 'pushed' to the right. The Uni✯Radar logo on the header was also righter than it should've been.

1d. The sidebar, the key navigation tool for the website did not function correctly. The HTML and CSS side of the sidebar was incorrect, with the '☰ Open' div in the header, which should toggle the visibility of the sidebar, would not trigger the `"toggleSidebar(true)"` method.

2. Routing

After the problems that occurred after the conversion were resolved, the next problem to solve and test was the correct implementation of routing. Routing in this project is controlled by the JavaScript file *index.js* in the folder *router*, which itself is in the *src* folder. For each *href* link in the original project, a *router-link* replaces it. These links will be found in, and tested for the components HomePage.vue, Sidebar.vue, FoodandBeverage.vue and Header.vue. The following screenshots show snippets of the *index.js* code.





3. rateClub and postComment methods

These methods, found within the individual Food and Beverage pages, control the users' star rating and their ability to post comments respectively. These methods will provide the foundation for the backend to save the users' comments and ratings.

rateClub method

```
rateClub(star){
  this.rating=star;
  console.log("Rating = "+star);
},
```

postComment method

```
async postNewComment(){
  console.log(this.newComment);
  console.log(this.rating);
  try{
    const response= await fetch('https://postcontent-3b74umvm6q-uc.a.run.app',{
      method: 'POST',
      headers:{
        'Content-Type': 'application/json', // Corrected typo here
      },
      body: JSON.stringify({
        text: this.newComment,
        rating: this.rating,
        source: this.source,
        handle: this.handle,
        userId: this.userId,
      }),
    });
    const data= await response.json();
    console.log('Response from post comment:',data);
    this.fetchComments(); // Corrected function call here
  }
  catch(error){
    console.error('Error Posting comment: ',error);
  }
},
```

Back-end Testing

Postman and Firebase Emulators

The back-end functions were tested using Postman and Firebase emulators. In rigorously checking the back-end functionality of my web application, I employed strategic implementation of Postman and Firebase emulators to simulate real-time database queries and requests.

Testing functions in Postman before deploying :

Testing get Function

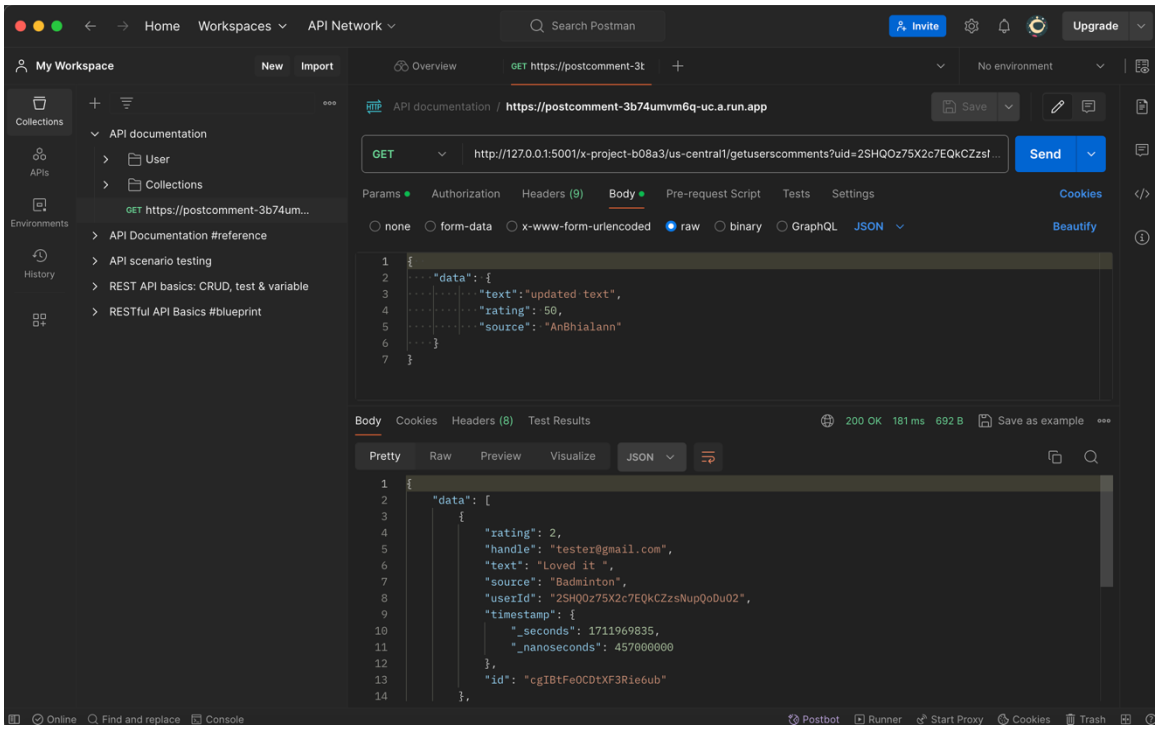


Fig: testing getUserComments

Expected output: return comments with userid= 2SHQOz75X2c7EQkCZzsNupQoDuO2

Result: as expected.

Testing Update Function

Fig: data before calling update function

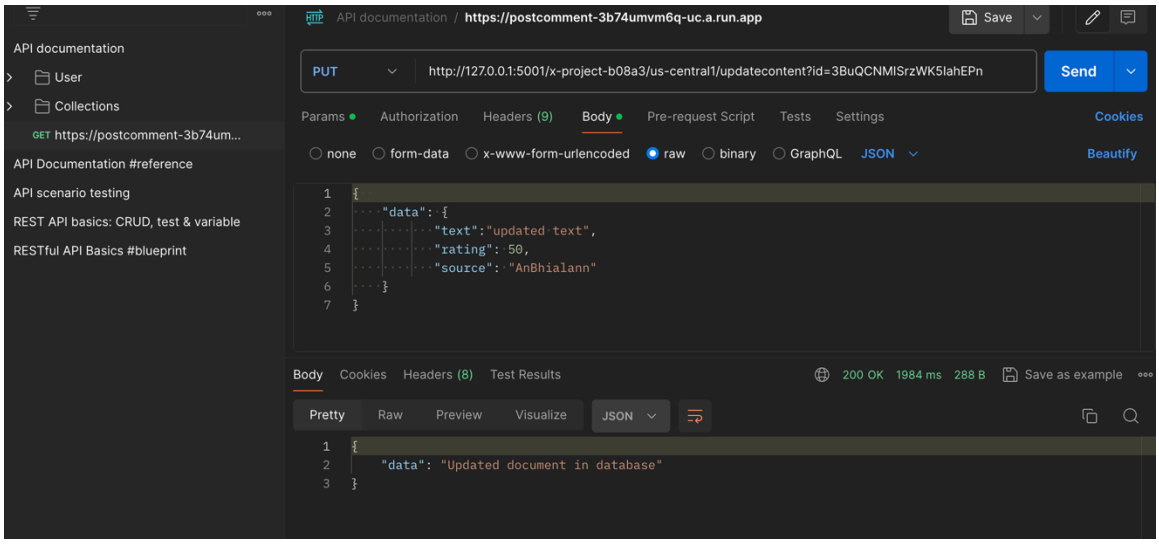


Fig: Testing updateContent

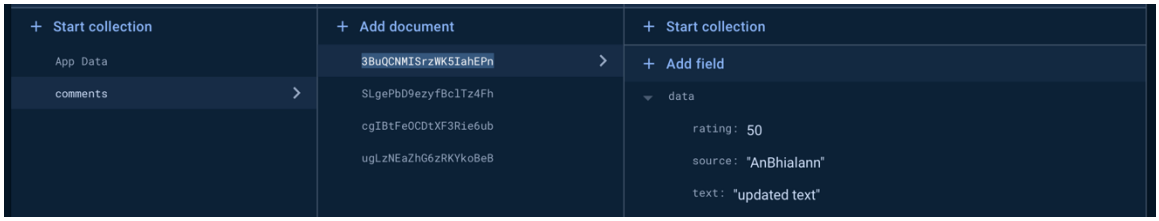
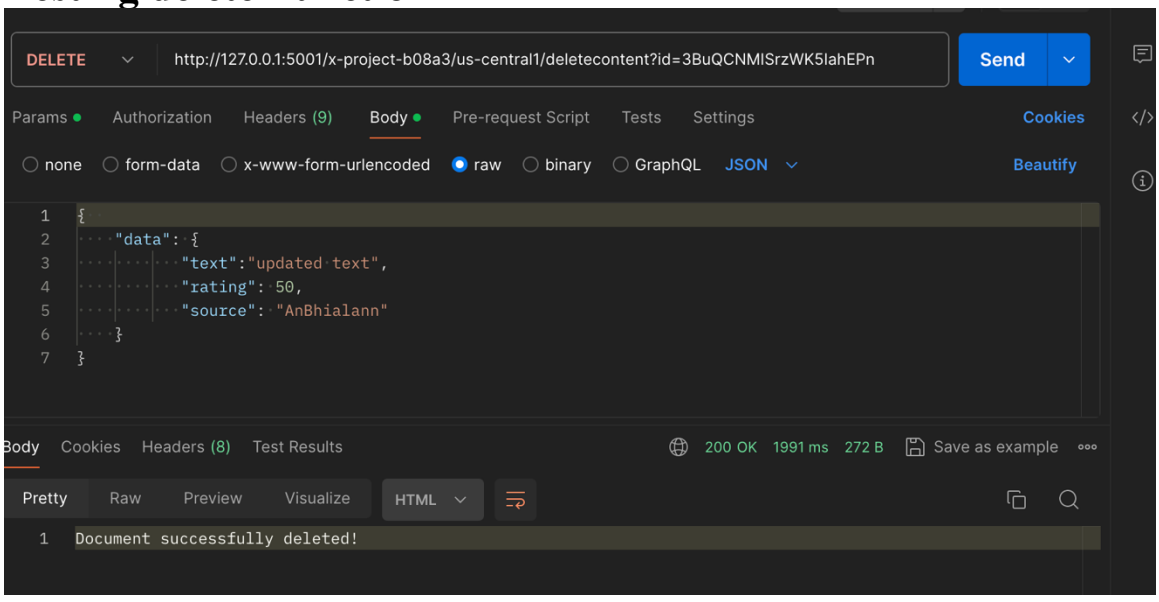
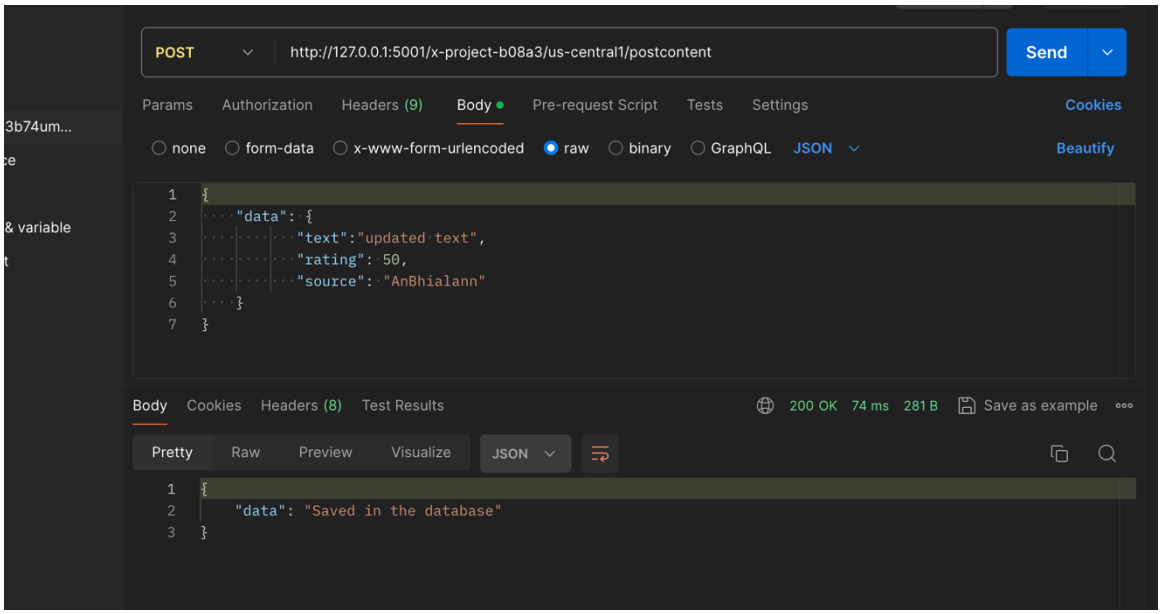


Fig: data after calling update function

Testing delete Function



Testing Post Function

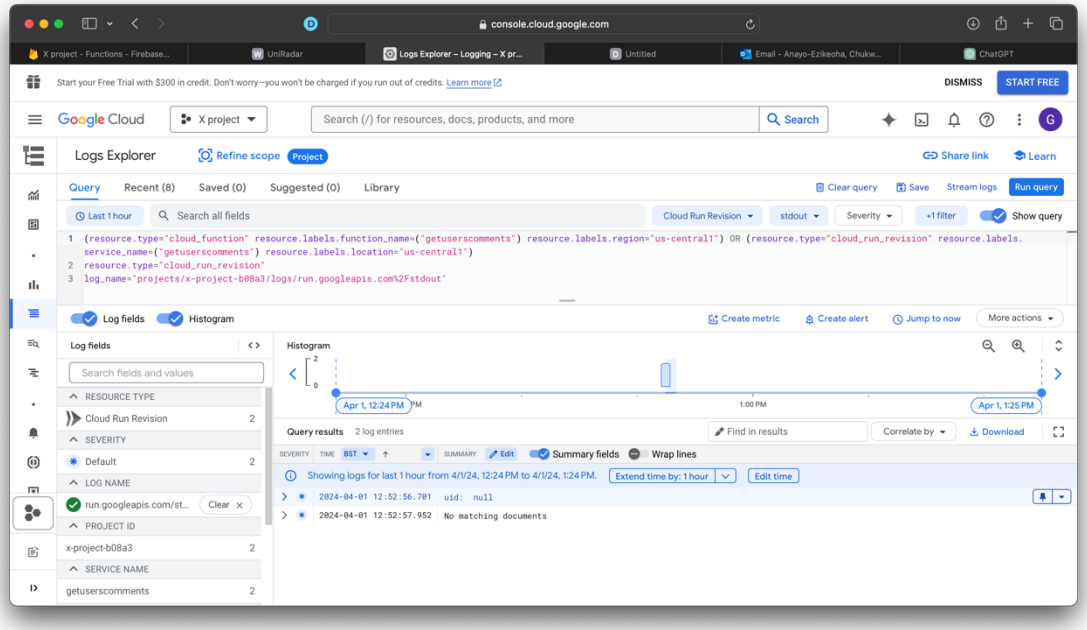


Results:

I was able to fully implement CRUD functionality into Uniradar by thoroughly testing the functions above and deploying them to firebase to be called by the web app. Results were as expected.

- Postman allows different test scenarios including POST, GET, DELETE and PUT as illustrated above.
- Postman also provides insightful error messages and headers which helps in debugging and makes it a very powerful testing tool.
- Postman was incredibly helpful in timesaving and streamlining the testing process, especially when combined with Firebase emulators.

Google Console Logs



- Besides Postman another valuable testing tool is the console log.
- The console log comes with the Firebase package and can be accessed by viewing the logs of your functions.
- The logs keep track of what happens during each invocation of your function and give details of errors or bugs that occurred while trying to invoke your functions.

12. Pass and Fail

1. Vue Conversion: Despite formatting changes, the overall UI of UniRadar was fixed after conversion to Vue files. In addition, some improvements were made to the overall look and feel of the page (see here, which is our first version of UniRadar on GitHub to compare). In addition, with help from the CT216 Lab Tutor, issues with the firebase and firestore were resolved.

2. Routing: Though we ran into some difficulties when routing to individual pages (e.g. when routing to /Clubs/ Badminton). For example, when trying to route using the path

```
{
        // path: '/Events/.Sports Ball'
}
```

I found that the '.' before the individual page was causing the routing issue. Thus, by removing this and routing like:

```
{
        path: '/Clubs/Basketball',
        name: 'Basketball',
        component: Basketball
      },
```

The Issue was resolved, the testing of routing passed with success.

IV. Individual Roles

George Anayo-Ezikeoha- Back-End Developer

As the project head of UniRadar, I played a pivotal role in ensuring the smooth execution of the project from inception to completion. My responsibilities included various aspects of project and people management as well as software development. Firstly, I organized regular meetings to facilitate effective communication and collaboration among team members. These meetings served as forums for discussing progress, addressing challenges, and aligning efforts towards our team goal.

In addition to overseeing the allocation of tasks, I took on the role of lead back-end developer, where I spearheaded the development of the backend infrastructure of UniRadar. This involved coding and deploying Firebase functions to handle data storage and retrieval, as well as setting up Firebase authentication for user registration and login.

Secondly, I collaborated closely with Ciaran (the Front-end engineer) to ensure seamless integration between the front-end and back-end components of UniRadar. By establishing clear communication channels and coordinating our efforts, we were able to optimize the performance and functionality of the web application.

Additionally, I played a pivotal role in implementing tools and processes to enhance project efficiency and productivity. From setting up project management tools like Notion to monitor the progress and ensuring adherence to timelines.

Throughout the project, I held myself and my team members accountable for their tasks and roles, fostering a culture of responsibility and ownership. By prioritizing effective communication, collaboration, and attention to detail, I contributed to the successful development and deployment of UniRadar, ultimately fulfilling our objective of enhancing the student experience at the University of Galway and I hope to take UniRadar to the next level someday.

Ciaran Daly – Lead Front-End Developer

Ciaran's role in our project was to lead the development of the UniRadar's Frontend, aiming to develop a visually appealing website, and flowed smoothly when users interacted with UniRadar's various components. Initially, I coded various sample pages, such as a home page, a landing page for the university's clubs, an about page and a sample club page using HTML and CSS. After liaising with my team, I developed the main theme and UI of the website, such as the layout for each page. After the design and layout process of the website was finalized, I began working on the JavaScript functions for UniRadar, such as handling the sidebars functionality, Star rating and commenting system for each page etc. When complete, I began the process of converting the HTML, CSS and JavaScript files into a new Vue project folder, developing a router to achieve a Single Page Application (SPA) for UniRadar. In addition, I added new sections like an events landing page and individual events pages, frontend pages for registration and user profiles which would implement backend functions developed by George. Moreover, I made various improvements to the UI, such as changing the Home Page to a simpler and more concise design and improving the general look of each component and features (like the comment section) to achieve a more polished, responsive and modern look to UniRadar. The files for final implementation of the

Frontend can be found on the 'Vue-Front-end' branch of our GitHub project repository.

Habel Sebastian- UI Design

Habel made contributions to the UniRadar project, primarily focusing on the initial UI design using HTML and CSS. I crafted the layout of key web pages providing a solid foundation for the project's user interface and ensuring a visually appealing and intuitive experience for users. Additionally, I set up the project on GitHub, establishing an organized repository for collaborative development and version control.

V. References

1. Get Bootstrap. Buttons. [Online]. getbootstrap.com. Available at: https://getbootstrap.com/docs/4.0/components/buttons/ [Accessed 25 March 2024].

2. W3 Schools. W3.CSS Tutorial. [Online]. https://www.w3schools.com/. Available at: https://www.w3schools.com/w3css/defaulT.asp [Accessed 21 March 2024].

3. W3 Schools. How TO - Typical Device Breakpoints. [Online]. https://www.w3schools.com/. Available at: https://www.w3schools.com/howto/howto_css_media_query_breakpoints.asp [Accessed 25 March 2024].

4. Paul Ryley (pryley). (2018). star-rating.js. [Online]. https://github.com/. Last Updated: 5 August, 2022. Available at: https://github.com/pryley/star-rating.js/blob/main/README.md [Accessed 25 March 2024].

5. Freepik. [Online]. https://www.freepik.com/. Available at: https://www.freepik.com/search?format=search&last_filter=query&last_value=icons&query=icons [Accessed 25 March 2024].

6. Vue 3+ Firebase Authentication: https://youtu.be/xceR7mrrXsA?si=p1BxQtPfY8IsyhFw

7. Serve a HTML Website or a Single page with Node js: https://youtu.be/fyc-4YmgLu0?si=E-v8eItJxKCc-SD

8. Stack Overflow: https://stackoverflow.com/questions/70242081/writing-data-to-firebase-using-vue

9. Reintech: https://reintech.io/blog/using-vue-js-with-firebase-tutorial

10. Cloud Functions for Firebase: https://firebase.google.com/docs/functions