University of Belize Clubs Management

ADA Designs

University of Belize

Human Computer Interface 2025-1

Mr. Manuel Medina

Abner B., Dair A., Angie H.

Table of Contents

Introduction

|	
Project Name	University of Belize Club Management System (UBCMS)
Team Members	Angie Hoare (Project Manager) Dair Aban (Programming Lead) Abner Bobadilla (Design Lead)
Value Proposition	*Swift and Easy Processing!*

Problem/ Solution Overview

Students currently face a challenge in accessing club-related information, and the reliance on a paper-based application process creates further inconvenience by making the process time-consuming. Our goal is to provide an online platform that caters to online applications, allows public viewing, and enables administrators to keep track easily.

Tasks & Final Interfaces

Task 1: Posting Club Events - *Simple*

This feature was designed to accommodate the dissemination of information, either within a single club or from a club to the public. This allows a club member to log in and create a post, easily choosing whether to post it publicly or internally within their club.
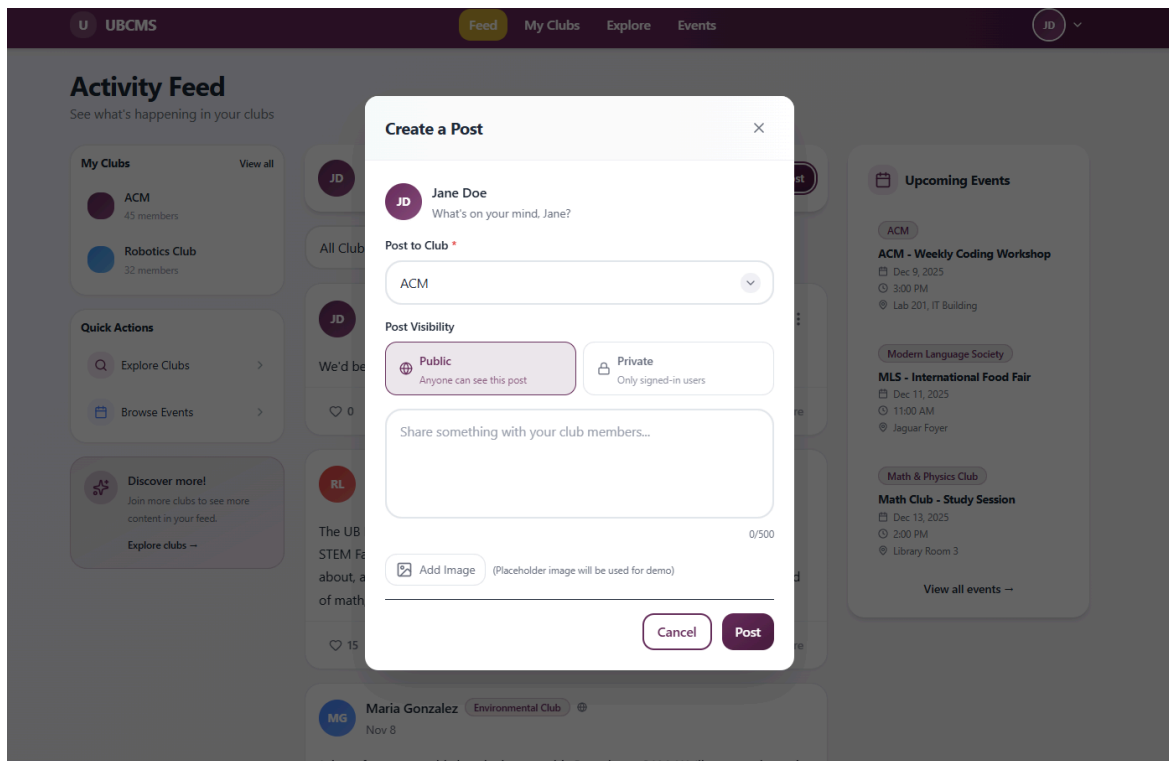
Task 2: Online Applications - *Moderate*

This feature enables UB students to submit their applications to join a club online. It also allows for faster processing by eliminating paper-based applications and providing an immediate response to the applicant from the Dean of Student Affairs once the application is reviewed online and the status is changed.

Task 3: Admin Tracking - *Complex*

This feature enables the Dean of Student Affairs to stay up-to-date with all the events each club is hosting, while also allowing for easy approval or denial of requests for all UB clubs.

Design Evolutions

Posting Club Events
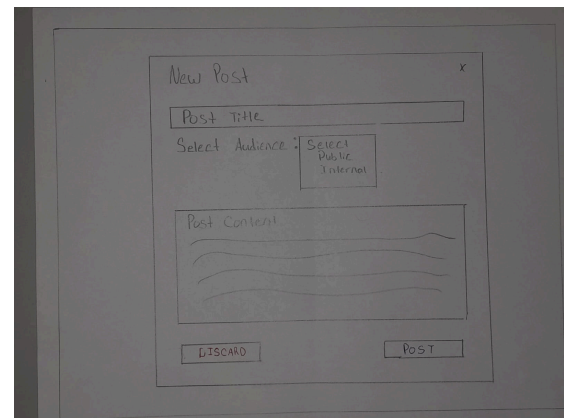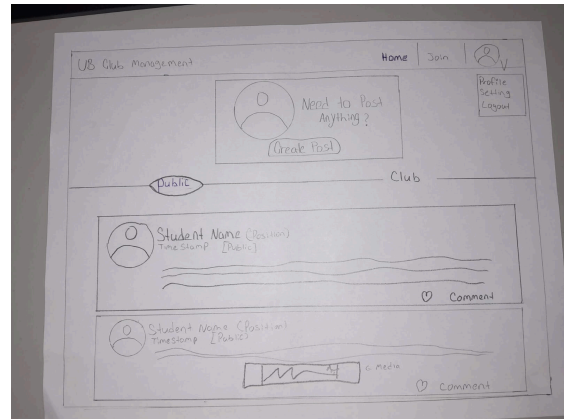
Low-Fi	
Medium-Fi	
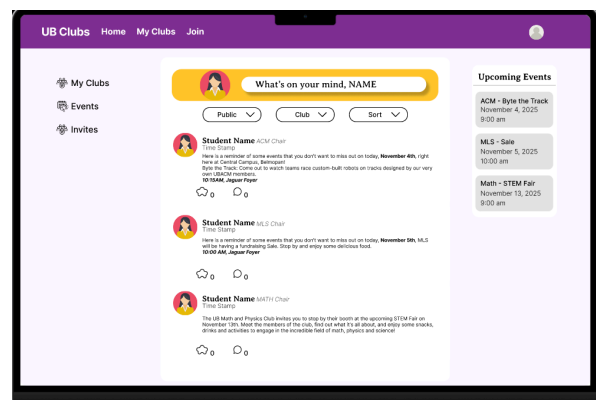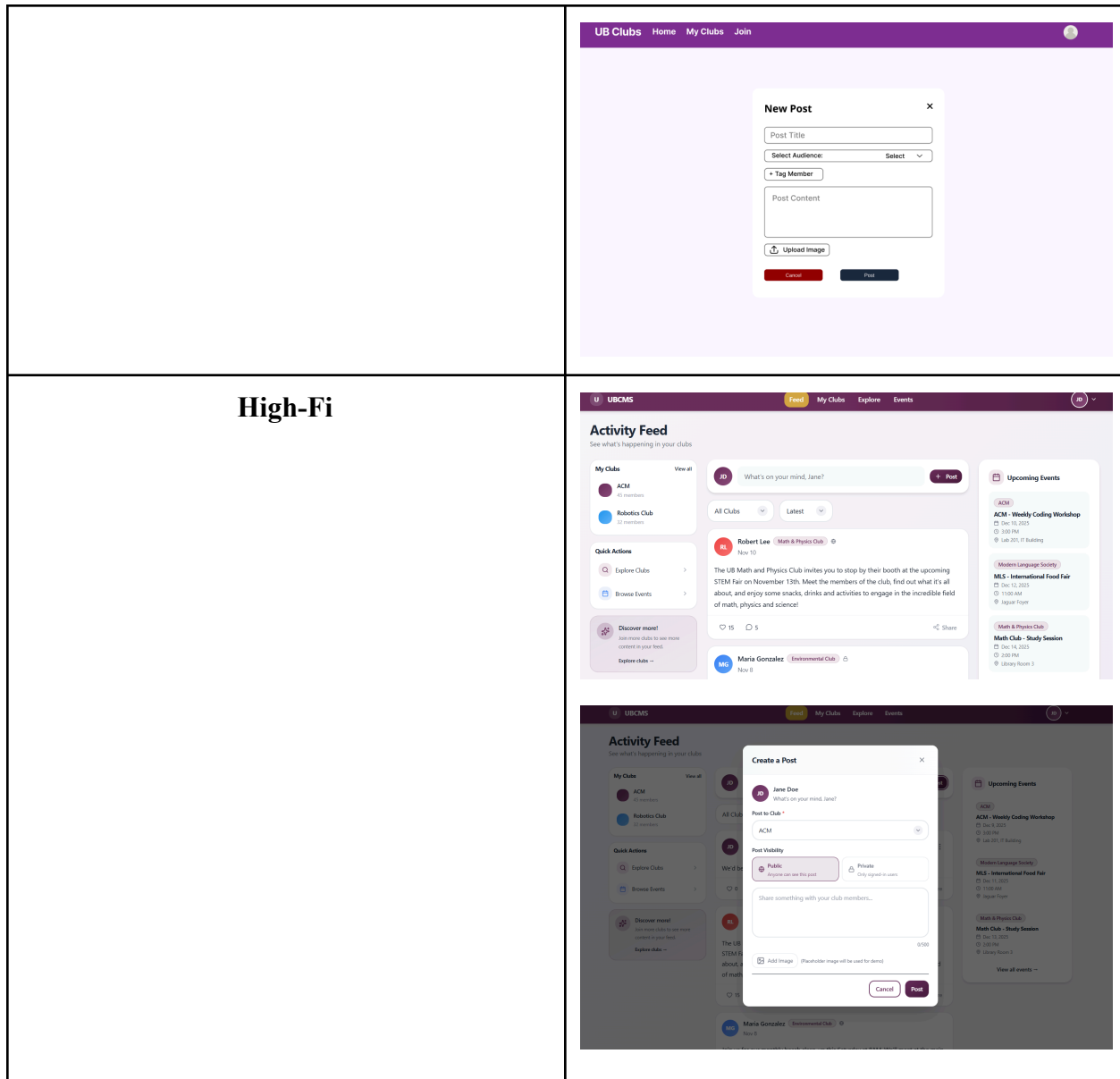
High-Fi	

Change(s):
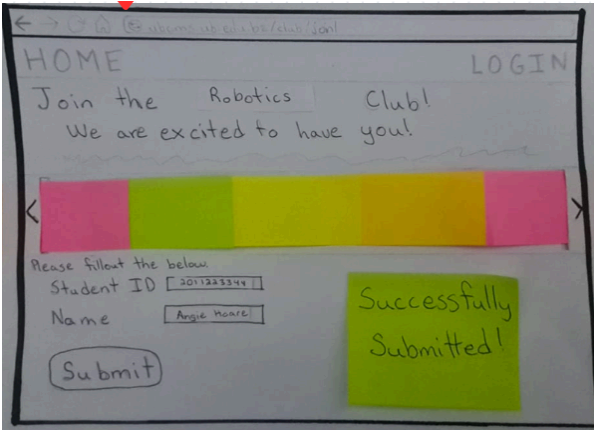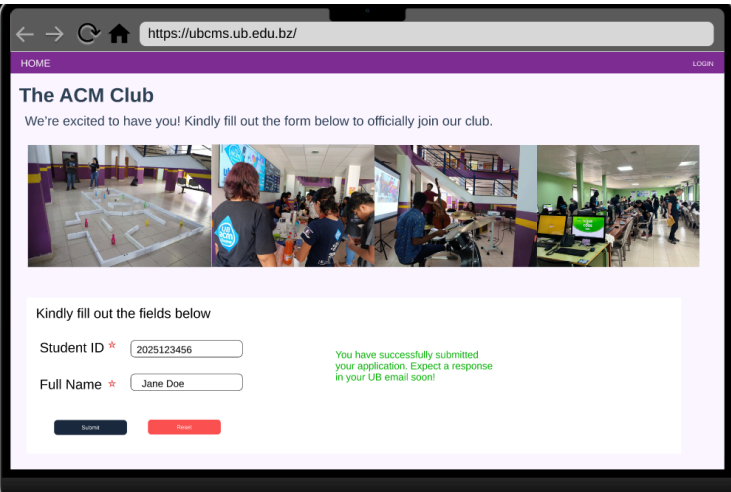
1. UI design of 'Create a Post', 'Select Audience', and sorting features.

2. Removal of the tag member option.

Reason:

1. Users were confused when asked to simulate creating a post in the medium-fidelity prototype, which is why the high-fidelity version includes a "Post" button next to the input modal. The "Select Audience" UI also appeared somewhat abstract and outdated. Additionally, the sorting feature made more sense with two options rather than three.

2. The "tag a member" feature was somewhat redundant, as the "Select Audience" options would be sufficient; different club members would already be able to see what is going on.

Online Applications

Low-Fi	
Medium-Fi	

High-Fi	

Change(s): Success Message

Reason: Success Message was adjusted to provide the user with more useful information after submission.

Admin Tracking

Low-Fi	
Medium-Fi	
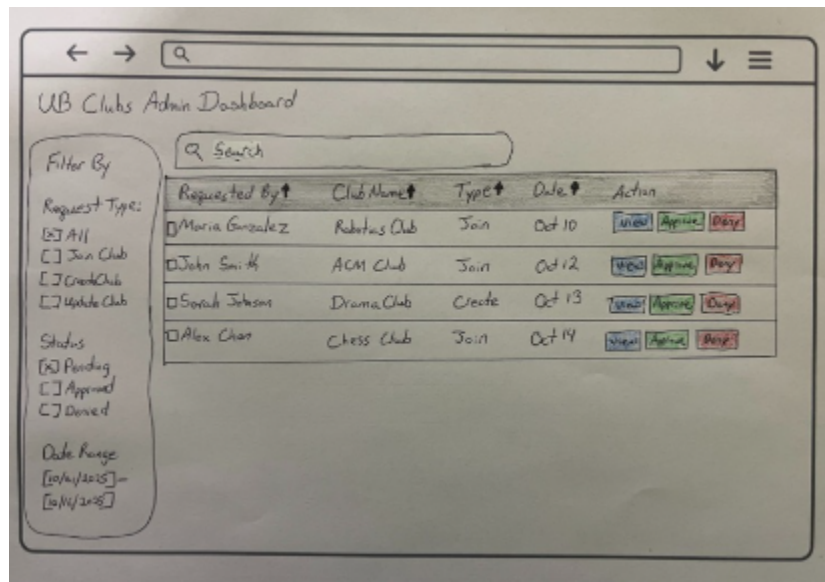
High-Fi	
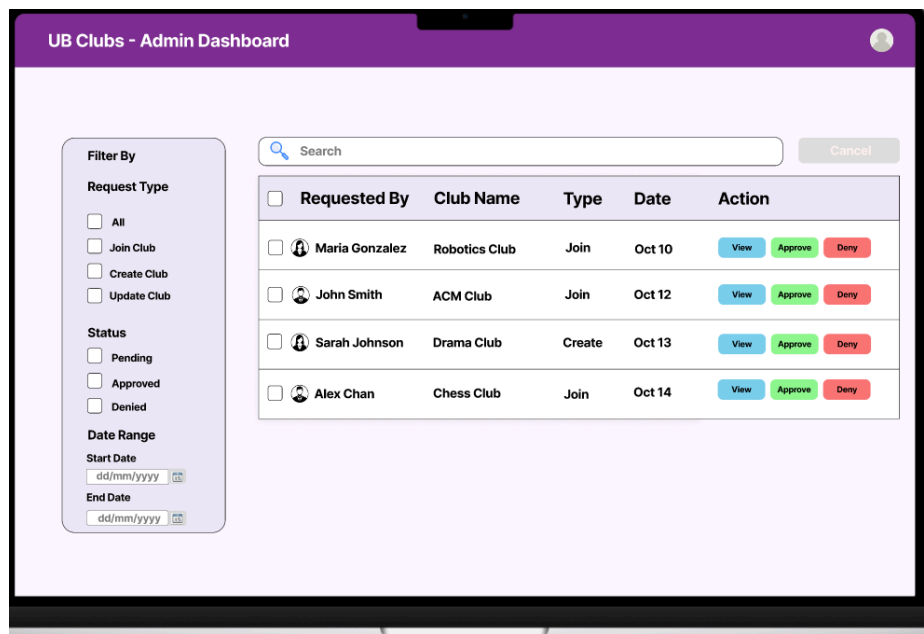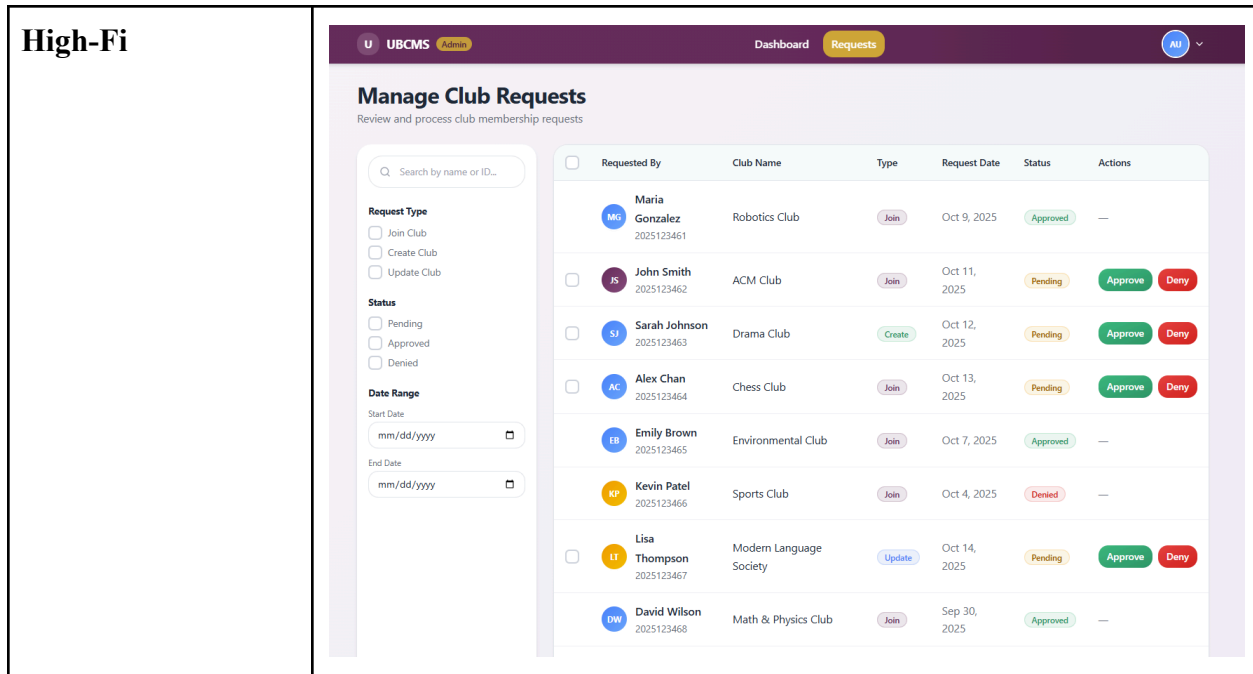
Change(s):

1. Added page header with descriptive subtitle ("Manage Club Requests" / "Review and process club membership requests").

2. Enhanced search functionality with specific placeholder text ("Search by name or ID..." instead of generic "Search").

3. Included student ID numbers below each requester's name.

4. Created a dedicated "Status" column with badge-style indicators (Approved, Pending, Denied) separate from the Actions column.

5. Implemented contextual actions - action buttons (Approve/Deny) only appear for pending requests.

6. Added user avatars with color-coded initials for each requester.

Reason:

1. The descriptive page header and subtitle provide immediate context about the page's purpose, helping administrators understand their task at a glance without relying on memory (Recognition over Recall).

2. The specific search placeholder ("Search by name or ID...") guides administrators on exactly what they can search for, reducing confusion and improving task completion speed.

3. Student ID integration allows administrators to quickly verify requester identity and cross-reference with university records, improving efficiency and reducing errors.

4. The separate Status column with badge styling makes it immediately clear which requests have been processed and which require action, improving scanability and reducing cognitive load.

5. Contextual actions (showing Approve/Deny only for pending requests) prevent errors by eliminating the possibility of re-approving or re-denying already processed requests, while also reducing visual clutter.

6. User avatars with color-coded initials provide quick visual identification and make the interface more user-friendly and modern.

Major Usability Problems Addressed

Recognition Not Recall (Severity 3)

During the heuristic evaluation, it was found that no description was provided to the user about each club on the club listing page, leaving the user to decipher what the club was about based solely on the image and name. To address this issue and further assist the user, a brief description of each club was included on the club listing page, improving efficiency and reducing the need for users to recall information. (image below shows implemented fix)
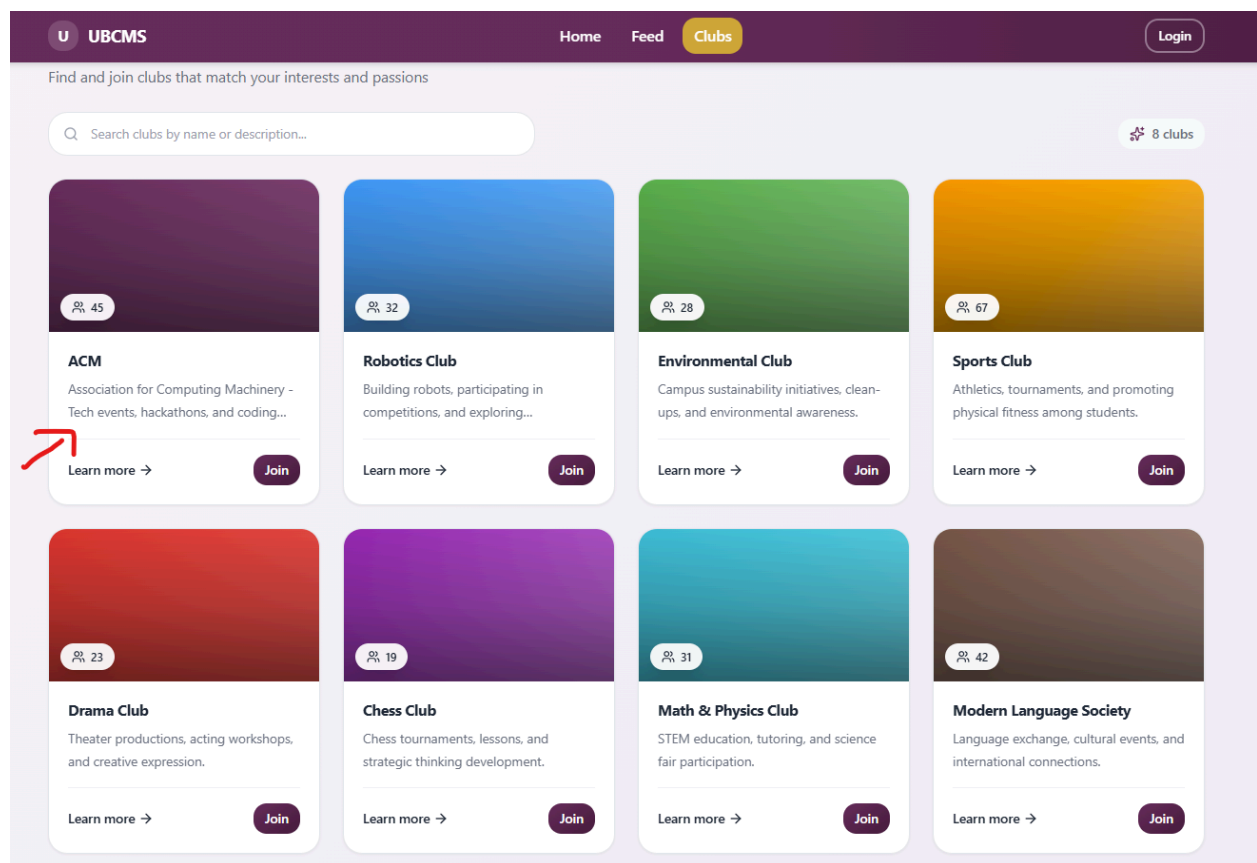
User Control & Freedom (Severity 4)

The major issue found during the heuristic evaluation was the lack of providing the user with the option to edit or delete a post once it was created. This took away the user's control and freedom to edit their own entries or remove them entirely from public/internal view. This was addressed by implementing both features, allowing users to edit or delete any post they have created. (see image below for the implemented fix)
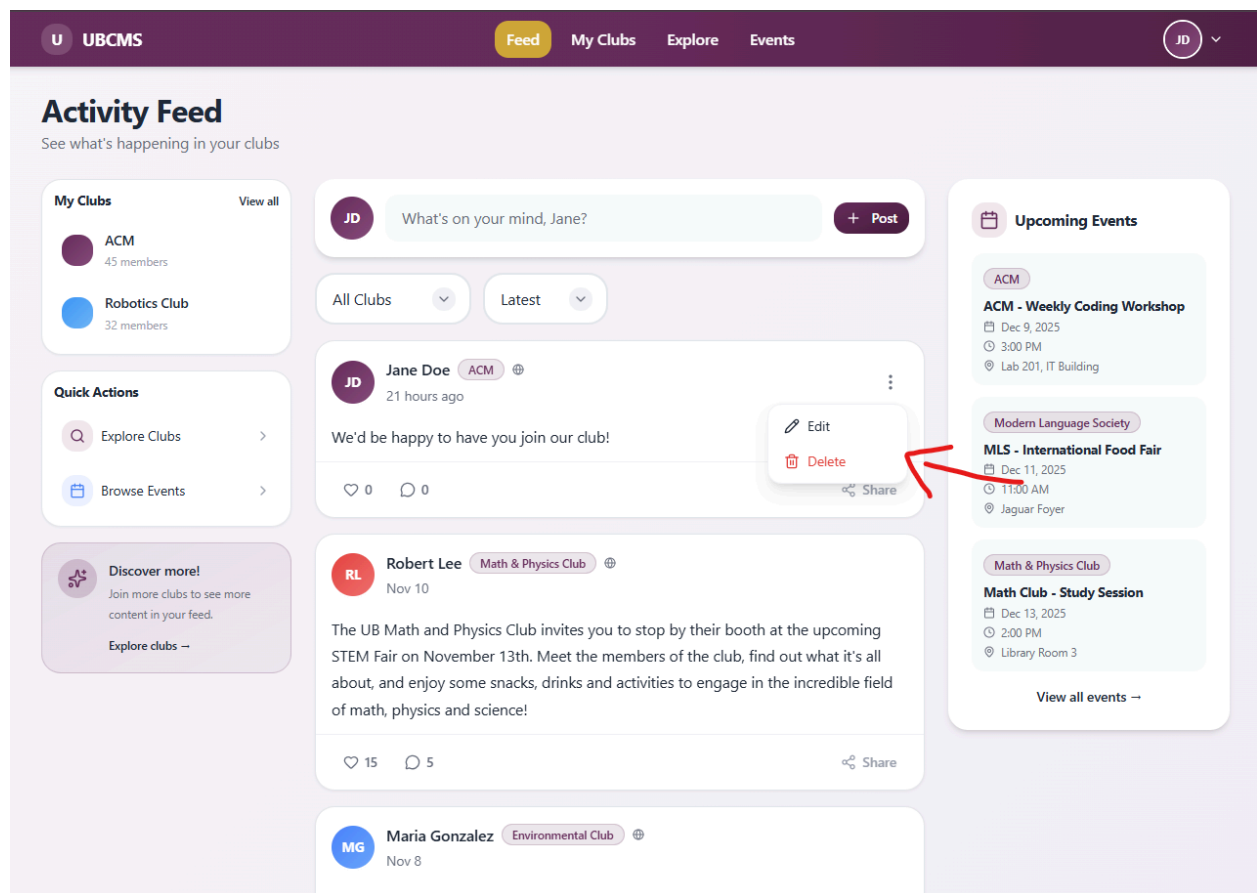
Prototype Implementation

Category	Technology
IDE	Visual Studio Code
Programming Language	Typescript
Framework	Next.js (React Framework)
Styling	Tailwind CSS
Data Persistence	LocalStorage API (Browser Native)

How tools helped:

1. **Visual Studio Code**
   - Provided IntelliSense autocomplete for TypeScript types and React components, reducing development time and syntax errors.
   - Version control integration.
   - Real-time error detection through ESLint and TypeScript extensions caught issues during development.

2. **TypeScrip**t
   - Strong type checking caught errors at compile-time, preventing bugs before testing.
   - Interface definitions for User, Club, Request, Post, and Event types created clear data structure contracts.
   - Enhanced IDE autocomplete for component props and function parameters improved development speed.

3. **Next.js Framework**
   - File-based routing system automatically created routes from the app directory structure, eliminating manual route configuration.
   - Built-in TypeScript support with zero configuration provided immediate type checking.

- App Router architecture made it easy to organize pages for different user roles (student, admin).

4. **Tailwind CSS**
   - Utility-first approach enabled rapid UI development without writing custom CSS files.
   - Built-in responsive modifiers (sm:, md:, lg:) facilitated mobile-first design implementation.
   - VS Code extension provided autocomplete for class names, reducing lookup time.

5. **LocalStorage API**
   - Simulated a backend database by persisting user data, clubs, posts, and requests in the browser.
   - Maintained user login state across page refreshes, enabling realistic session management.
   - Allowed seeding of initial data (clubs, users, events) for demonstration purposes.
   - Enabled full prototype functionality without building an actual backend server

The combination of these tools allowed the team to build an interactive, visually representative high-fidelity prototype with realistic user interactions, role-based navigation, and persistent data simulation, all without requiring a fully functioning backend infrastructure.

How tools did not help:

1. **Backend Limitations**
   - TypeScript and Next.js alone cannot persist data permanently beyond browser storage; localStorage was used as a temporary workaround but data is lost if browser cache is cleared.
   - Real user authentication and secure password management could not be implemented without a backend server.

2. **Advanced Feature Limitations**
   - Cannot implement real-time notifications or live updates without WebSocket connections or server-sent events.
   - File upload functionality for club images or user profile pictures requires backend storage solution.

Hardcoded Data:

The data that was hardcoded for the Hi-Fi prototype were elements that would typically be stored in a database and updated dynamically in a fully developed system. These were included solely to simulate real functionality during the demonstration. The hardcoded data consisted of:

- The list of UB available clubs displayed for users to browse and join
- Posts made by club members, used to showcase the platform's activity feed feature
- Club requests submitted by students, which are viewable and manageable by the admin
- One predefined account for a Club Member (Student) to demonstrate student-level access and interactions
- One predefined account for the Dean of Student Affairs (Admin) to demonstrate administrative privileges and dashboard features

By hardcoding these data points in localStorage with initialization logic, the prototype was able to effectively illustrate how the system would function once integrated with a real backend and live database.

Unimplemented Features:

Some of the features that have not been implemented but would play a key role in allowing the application to run smoothly and as intended are to be able to:

- Apply to create a new club
- Reporting feature for admin

<u>Summary</u>

The University of Belize Clubs Management System is a web application facilitating the swift and easy processing of all club-related requests. The three main tasks were successfully completed, allowing the user to *Join a club*, a club member to *Create a Post* and the Dean of Student Affairs to *Track requests and club events*. Based on the heuristic evaluation conducted, the violations of severity 3 and 4 were the lack of information being provided to the user (Recognition not recall) and the inability to edit or delete a post created by that user (User control and freedom). Both violations were addressed in the High-Fidelity prototype.