CSE471: System Analysis and Design

Project Report

Project Title: University Student Activity Management System

Group No: 10, CSE471 Lab Section: 06, Spring: 2025	
ID	**Name**
21201590	Md. Jadid Hossain Sanim
21201327	MD. Kawsar Habib
21301188	Md. Rakib Hossain
21201009	Md. Nafizur Rahman Bhuiya

Submission Date: 04-08-2025

Table of Content

1. System Request

Business need:

In an effort to provide a centralized and interactive platform, the University Student Activity Management System(SAM Portal) facilitates the linkage between students, clubs and university staff by using event management, club involvement, communication and sharing of resources. This system offers a platform to manage student profile, subscribe to student organizations, register to activities and to access academic and non-academic resources. It can be used as an online resource where students are able to participate in extracurricular activities, cooperate better and be informed by notifications and schedules which eventually enriches the student life and efficiency of the administration.

Business requirements:

The business requirements of the University Student Activity Management System(SAM Portal) are related to providing a holistic solution comprising a secure role based authentication to students, club administrators and university employees. The system should enable the user to create an account, log in and work with their profiles including academic and extracurricular information. Searchable directories, subscription and multi stage membership applications make club management easy. The creation, registration and calendar integration of events provide adequate coordination of university programs. The platform has content interaction through announcements, timelines and discussion forums, as well as real time messaging and multi channel notifications. Other features of the career such as job advertisements and updates on opportunities also give students access to future opportunities. Governance becomes lean with a centralized administration dashboard and knowledge sharing among the community becomes easier with resource repository.

Business value:

The University Student Activity Management System(SAM Portal) is highly valuable in that it automates and digitizes the process of managing student activities at the university. It reduces operational costs to administrators by reducing manual operations and improves

student engagement with clubs, events and resources. The platform promotes teamwork which enhances communication and creates a greater university community. In the long run, the system can produce meaningful data on student engagement and interest that can guide institutions to make informed decisions about extra curricular activities based on the data. Combined with career opportunities and networking characteristics, it may enhance student professional growth over the long run. This is what makes the platform not only an instrument to organize student affairs but also a tactical enabler of student developments and institutional progress.

Special issues or constraints:

The University Student Activity Management System(SAM Portal) is faced with a number of essential concerns and limitations that must be resolved to achieve effective implementation and maintenance of the system. It must be scalable and it must be capable of handling thousands of students, events and interactions at any given time when there are peak periods like registration weeks. Its significance to the security and privacy cannot be overemphasized and should be highly encrypted with role based access control and adherence to the organizational data policies to facilitate integrity of the sensitive student data. Another important aspect is performance because the system should be able to respond quickly and be available at all times. Besides that, the system should also be designed responsively in a way that would not only be compatible with the gadgets but also provide useful monitoring and content management applications that would bring professional and credible interaction. This is supported and makes the service more usable in the long run by the future integration with the external platform. Lastly, it has to be enabled by a long term support and feedback process to allow the system to adapt to new student and institutional needs.

2. Functional Requirements

The University Student Activity Management System will consist of three main modules like Authentication & Core Management, Club & Event Management and Social Media, Communication & Resource Management. The following are the detailed functional requirements:

Module 1: Authentication & Core Management

Multi role User Registration: The system must allow registration of students, club administrators, and university staff with role-based access control.

Profile Management: Users must be able to view and update their profiles, including academic information, interests, contact details, and profile pictures.

Lost and Found System**:** Allows students and staff to report and track lost or found items with details like description and location.

Central Admin Dashboard: The system must provide a central dashboard for administrators to manage clubs, assign admins, and analyze user activities.

Multi channel Notification System: The system must support email and in-website notifications for events, deadlines, club updates, and personal messages.

Module 2: Club & Event Management

Club Directory, Discovery & Subscription: The system must maintain a searchable catalogue of all university clubs with descriptions and member counts. Students should be able to subscribe to specific clubs for targeted notifications.

Club Membership Application & Approval Workflow: Students must be able to apply to join clubs using custom application forms with file upload. The system must support a multi-stage approval process including application review, viva scheduling, and confirmation.

Club Admin Dashboard: The system must provide tools for club administrators to manage member lists, approve applications, and organize club activities.

Event Creation & Management: University staff must be able to create, schedule, and manage events, competitions, and programs.

Event Registration & Calendar Integration: Students must be able to register for events and inter-university competitions through form submissions and deadline tracking. A centralized calendar must display all university activities, deadlines, and important dates with filtering options.

Module 3: Social Media, Communication & Resource Management

Club Activity Timeline: Each club must have a dedicated page displaying upcoming announcements, achievements, and events.

Job Opportunity Update & Career Discussion System: The system must provide a section for job postings, internships, guidelines, and career opportunities.

Course Resource Repository: The system must support a digital library for non-academic course materials and educational resources.

Thread-based Discussion Forums: The system must allow users to create and participate in topic-based discussions related to clubs, extracurricular activities, and academic subjects.

Real-time Messaging System: Users must be able to send direct messages and participate in group chats for clubs, event management teams, and special groups (e.g., CP-only groups).

Social Feed, Community Groups & News Updates: The system must provide a university-wide social feed for posts, announcements, and student activity updates. Users must also be able to create informal, interest-based groups beyond official clubs

3. Technology (Framework, Languages)

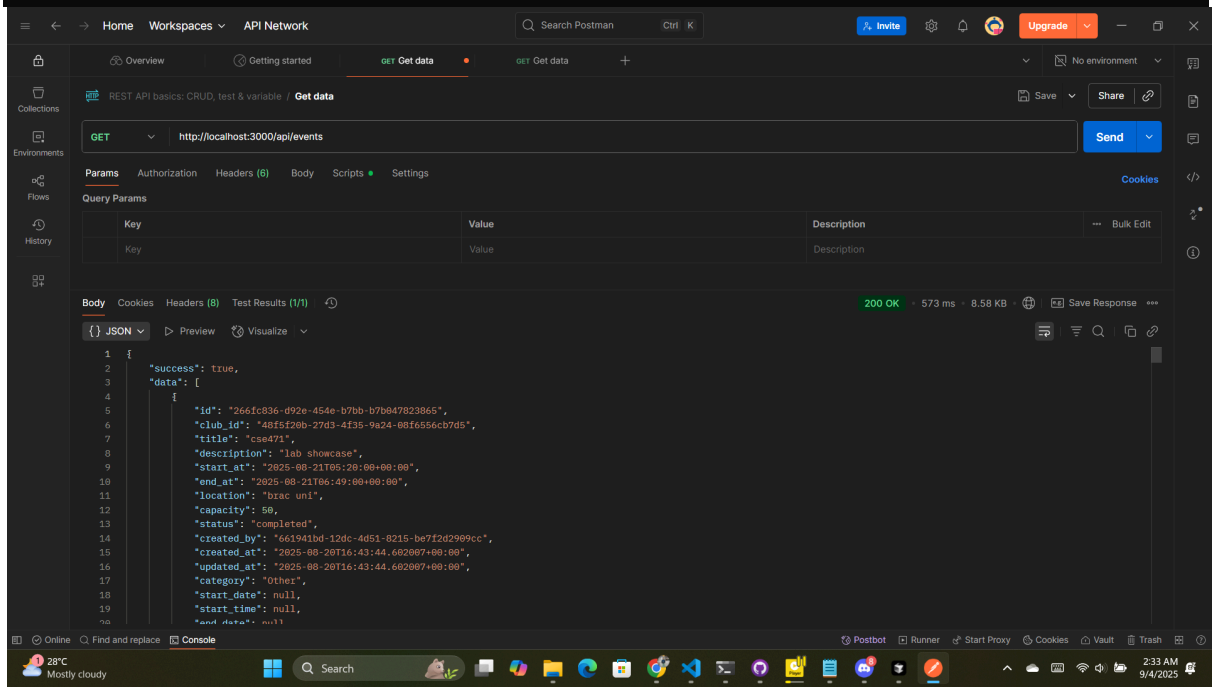
TypeScript, NextJS, TailwindCSS, PostgreSQL

4. Backend Development

1. The Events API retrieves event data from your Supabase database, organizing events by start time. It allows filtering by category and public status, with pagination support for efficient data loading. The API returns event details including titles, descriptions, timing, and location information for calendar and event management applications.

```
// 1. Events API
app.get('/api/events', async (req, res) => {
  try {
    const { category, is_public, limit = 50, offset = 0 } = req.query;

    let query = supabase
      .from('events')
      .select('*')
      .order('start_at', { ascending: true });

    // Apply filters
    if (category) {
      query = query.eq('category', category);
    }

    if (is_public !== undefined) {
      query = query.eq('is_public', is_public === 'true');
    }

    // Apply pagination
    query = query.range(parseInt(offset), parseInt(offset) +
parseInt(limit) - 1);

    const { data, error, count } = await query;

    if (error) {
      console.error('Database error:', error);
      return res.status(500).json({ error: 'Failed to fetch events' });
    }

    res.status(200).json({
      success: true,
```

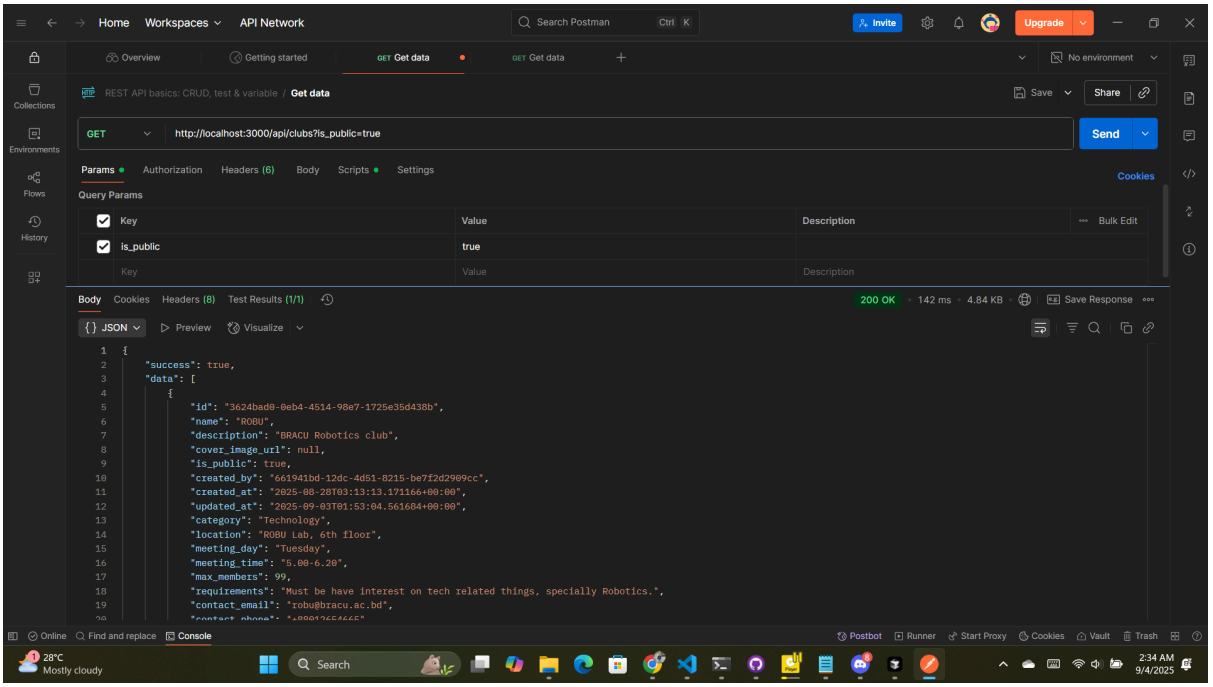
```
      data: data || [],
      count: count || 0,
      pagination: {
        limit: parseInt(limit),
        offset: parseInt(offset),
        hasMore: (data || []).length === parseInt(limit)
      }
    });

  } catch (error) {
    console.error('API error:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```



2. The Clubs API fetches club information from your database, displaying clubs sorted by creation date. Users can filter results by public visibility and category to find specific types of clubs. The response includes club names, descriptions, categories, and associated images, making it ideal for club directories and membership systems.

```
// 2. Clubs API
app.get('/api/clubs', async (req, res) => {
  try {
    const { is_public, category, limit = 50, offset = 0 } = req.query;

    let query = supabase
```

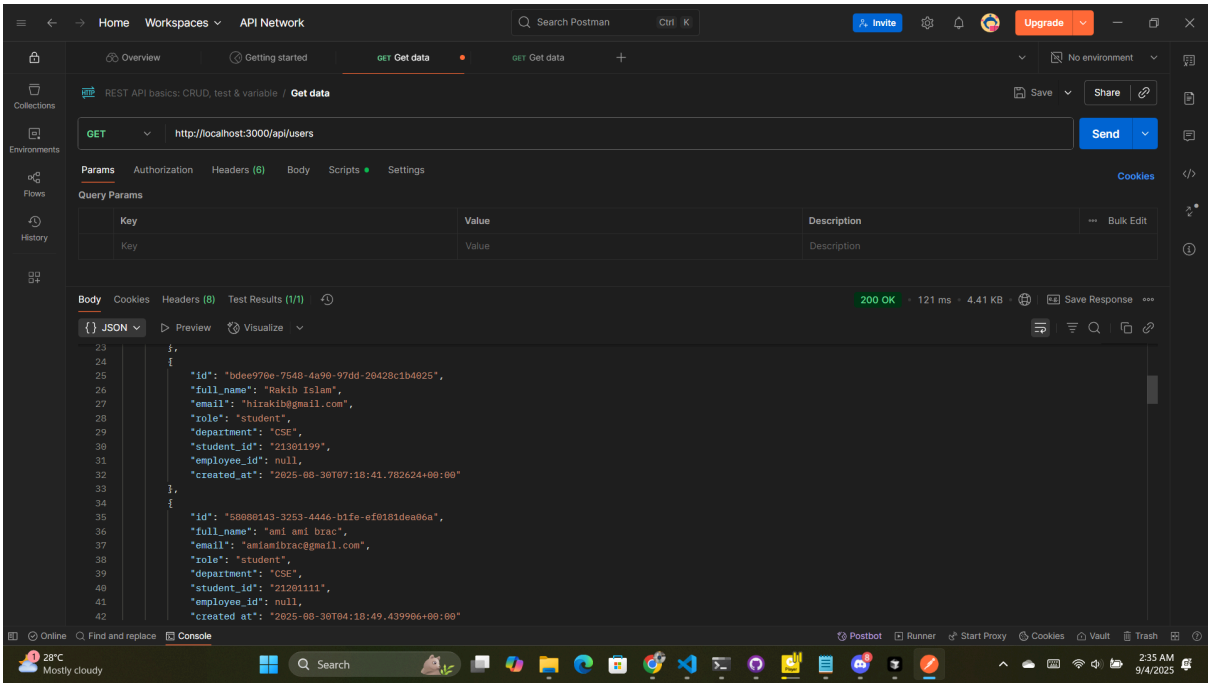
```
      .from('clubs')
      .select('*')
      .order('created_at', { ascending: false });

    // Apply filters
    if (is_public !== undefined) {
      query = query.eq('is_public', is_public === 'true');
    }

    if (category) {
      query = query.eq('category', category);
    }

    // Apply pagination
    query = query.range(parseInt(offset), parseInt(offset) +
parseInt(limit) - 1);

    const { data, error, count } = await query;

    if (error) {
      console.error('Database error:', error);
      return res.status(500).json({ error: 'Failed to fetch clubs' });
    }

    res.status(200).json({
      success: true,
      data: data || [],
      count: count || 0,
      pagination: {
        limit: parseInt(limit),
        offset: parseInt(offset),
        hasMore: (data || []).length === parseInt(limit)
      }
    });

  } catch (error) {
    console.error('API error:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```



3. The Users API provides access to user data with role-based filtering for students, faculty, and administrators. It supports department and status filtering, plus search functionality across names, emails, and student IDs. The API returns essential user information while maintaining privacy by excluding sensitive data.

```
// 3. Users API
app.get('/api/users', async (req, res) => {
  try {
    const { role, department, status, search, limit = 50, offset = 0 }
= req.query;

    let query = supabase
      .from('users')
      .select('id, full_name, email, role, department, student_id,
employee_id, created_at')
      .order('created_at', { ascending: false });

    // Apply filters
    if (role) {
      query = query.eq('role', role);
    }

    if (department) {
      query = query.eq('department', department);
    }
```

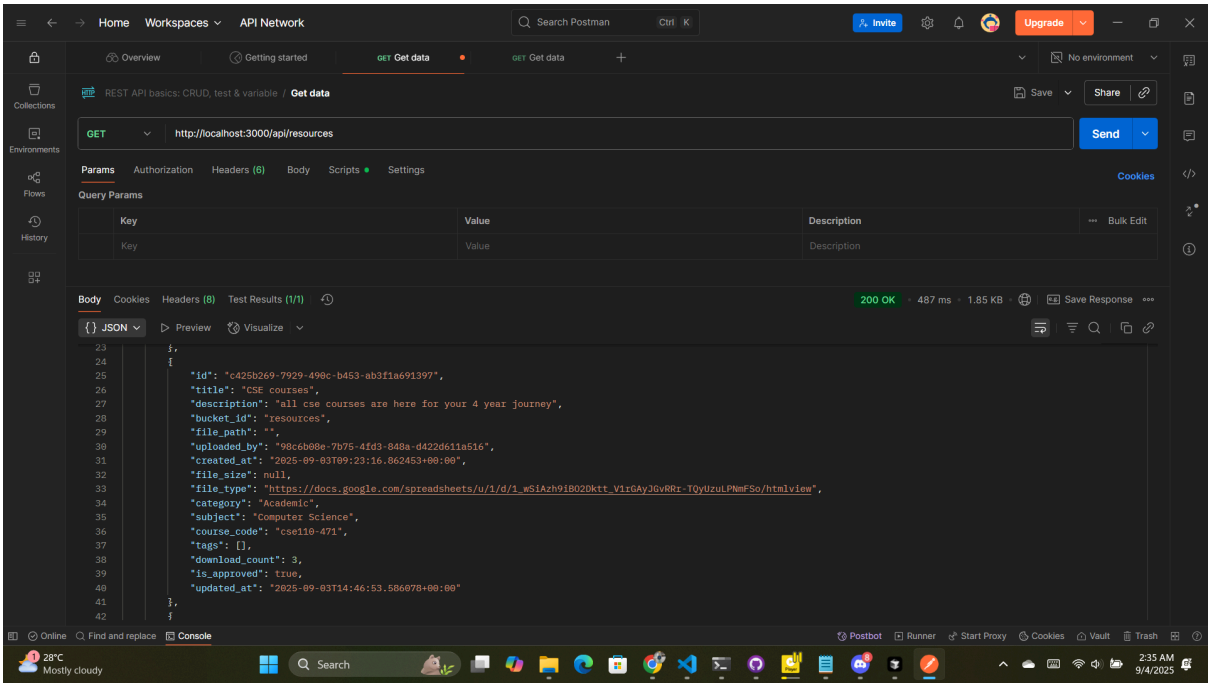
```
    if (status) {
      query = query.eq('status', status);
    }

    if (search) {
      query = query.or(
`full_name.ilike.%${search}%,email.ilike.%${search}%,student_id.ilike.%${search}%`
      );
    }

    // Apply pagination
    query = query.range(parseInt(offset), parseInt(offset) + parseInt(limit) - 1);

    const { data, error, count } = await query;

    if (error) {
      console.error('Database error:', error);
      return res.status(500).json({ error: 'Failed to fetch users' });
    }

    res.status(200).json({
      success: true,
      data: data || [],
      count: count || 0,
      pagination: {
        limit: parseInt(limit),
        offset: parseInt(offset),
        hasMore: (data || []).length === parseInt(limit)
      }
    });

  } catch (error) {
    console.error('API error:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```



4. The Resources API manages educational content by retrieving resources from your database with category, subject, and course code filtering. It includes search capabilities for finding specific materials and handles special cases like "All" categories. The API returns resource metadata, file information, and approval status for digital library systems.

```
// 4. Resources API
app.get('/api/resources', async (req, res) => {
  try {
    const { category, subject, course_code, search, limit = 50, offset
= 0 } = req.query;

    let query = supabase
      .from('resources')
      .select('*')
      .order('created_at', { ascending: false });

    // Apply filters
    if (category && category !== 'All') {
      query = query.eq('category', category);
    }

    if (subject && subject !== 'All') {
      query = query.eq('subject', subject);
    }
```

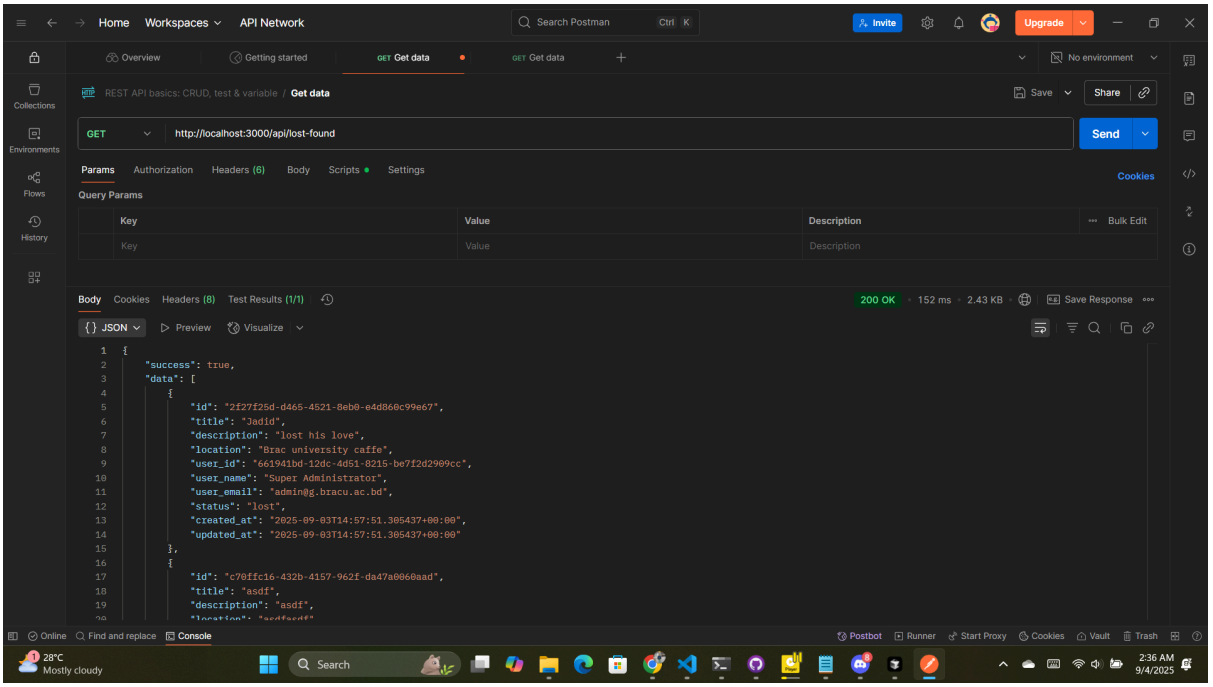
```
    if (course_code && course_code !== 'All') {
      query = query.eq('course_code', course_code);
    }

    if (search) {
      query = query.or(
        `title.ilike.%${search}%,description.ilike.%${search}%`
      );
    }

    // Apply pagination
    query = query.range(parseInt(offset), parseInt(offset) +
parseInt(limit) - 1);

    const { data, error, count } = await query;

    if (error) {
      console.error('Database error:', error);
      return res.status(500).json({ error: 'Failed to fetch resources'
});
    }

    res.status(200).json({
      success: true,
      data: data || [],
      count: count || 0,
      pagination: {
        limit: parseInt(limit),
        offset: parseInt(offset),
        hasMore: (data || []).length === parseInt(limit)
      }
    });

  } catch (error) {
    console.error('API error:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```



5. The Resources API manages educational content by retrieving resources from your database with category, subject, and course code filtering. It includes search capabilities for finding specific materials and handles special cases like "All" categories. The API returns resource metadata, file information, and approval status for digital library systems.

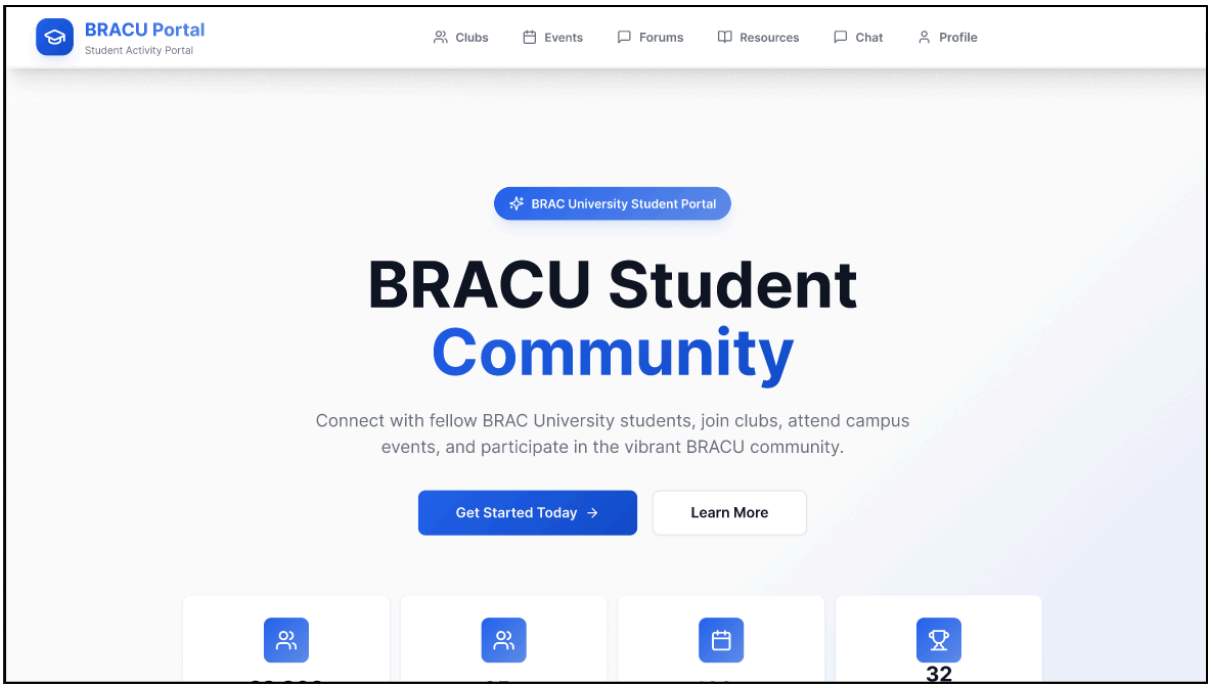
```
// 5. Lost & Found API
app.get('/api/lost-found', async (req, res) => {
  try {
    const { status, search, limit = 50, offset = 0 } = req.query;

    let query = supabase
      .from('lost_found_items')
      .select('*')
      .order('created_at', { ascending: false });

    // Apply filters
    if (status) {
      query = query.eq('status', status);
    }

    if (search) {
      query = query.or(
```

```
`title.ilike.%${search}%,description.ilike.%${search}%,location.ilike.%
${search}%`
      );
    }

    // Apply pagination
    query = query.range(parseInt(offset), parseInt(offset) +
parseInt(limit) - 1);

    const { data, error, count } = await query;

    if (error) {
      console.error('Database error:', error);
      return res.status(500).json({ error: 'Failed to fetch lost and
found items' });
    }

    res.status(200).json({
      success: true,
      data: data || [],
      count: count || 0,
      pagination: {
        limit: parseInt(limit),
        offset: parseInt(offset),
        hasMore: (data || []).length === parseInt(limit)
      }
    });

  } catch (error) {
    console.error('API error:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```

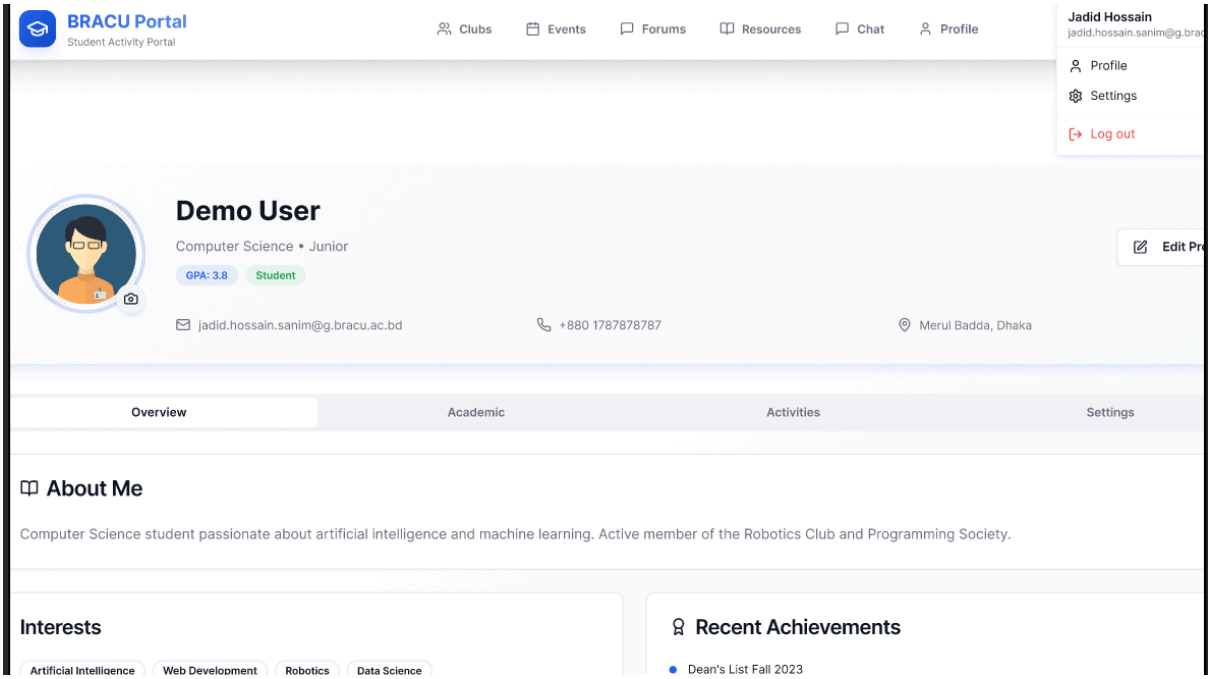


5. User Interface Design

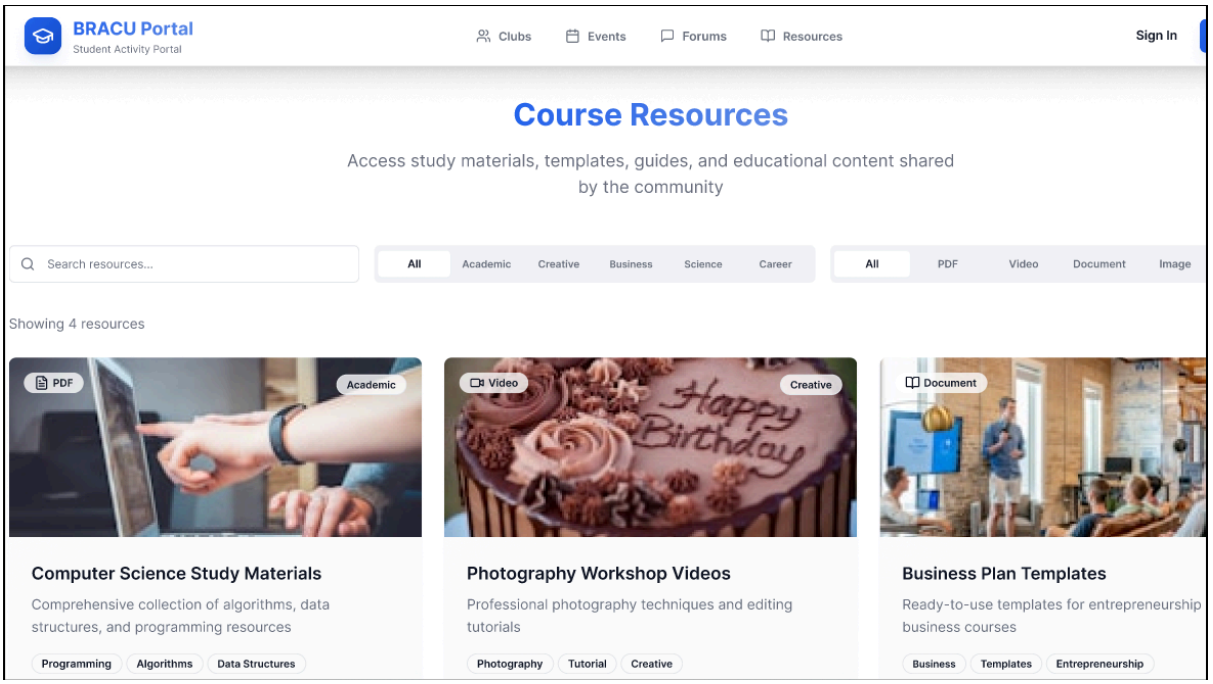
Home page:



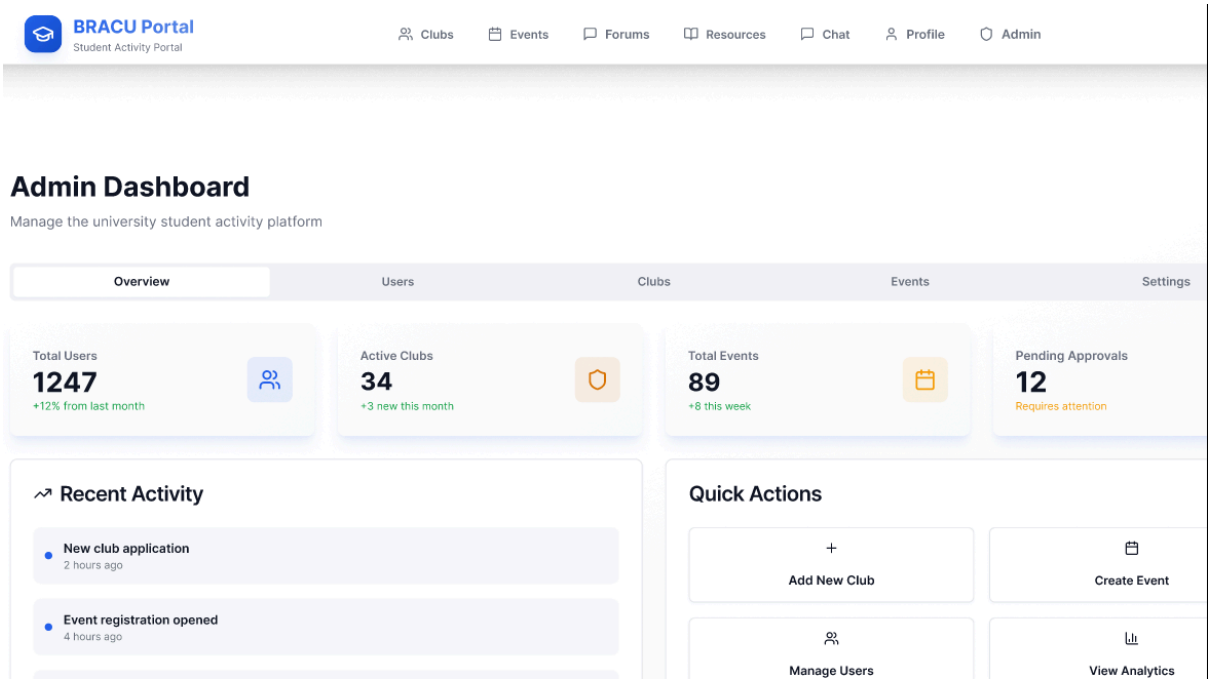
Profile:



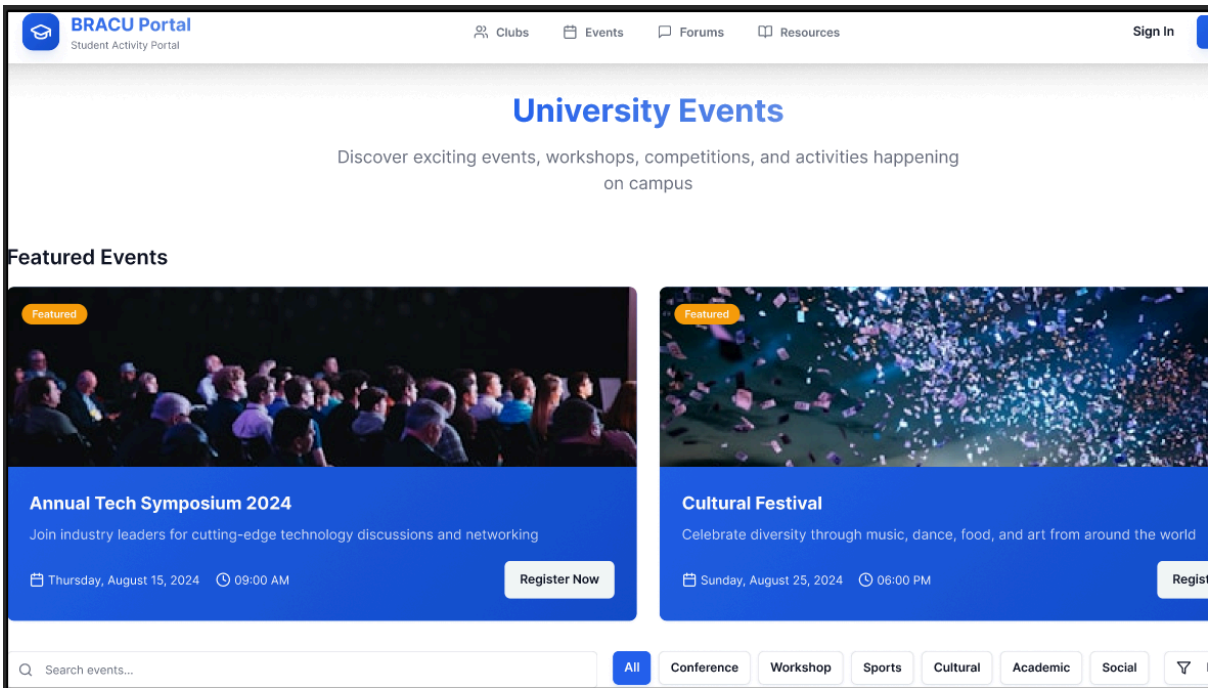
Resources:



Admin Dashboard:



Events:



Figma Prototype link: [Figma Link]

6. Frontend Development

Club page: In the club page, the users can see club details, panel, achievements, upcoming events and awards.

```
import { useState, useEffect } from "react";
import { Search, Filter, Users, Calendar, MapPin, Star, Bell, BellOff } from
"lucide-react";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Card } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Link } from "react-router-dom";
import Layout from "@/components/Layout";
import { useClubs } from "@/hooks/useDatabase";
import { useClubSubscriptions } from "@/hooks/useClubSubscriptions";
import { LoadingSpinner } from "@/components/LoadingSpinner";
import { useAuth } from "@/context/AuthContext";
return (
   <Layout>
     <div className="container mx-auto px-4 py-8">
       {/* Header Section */}
       <div className="text-center mb-12">
         <h1 className="text-4xl font-bold bg-gradient-primary bg-clip-text
text-transparent mb-4">
           Discover University Clubs
         </h1>
         <p className="text-xl text-muted-foreground max-w-2xl mx-auto">
           Find your passion, connect with like-minded students, and make
           lasting memories
         </p>
       </div>

       {/* Search and Filter */}
       <div className="flex flex-col md:flex-row gap-4 mb-8">
         <div className="relative flex-1">
           <Search className="absolute left-3 top-1/2 transform -translate-y-1/2
text-muted-foreground h-4 w-4" />
           <Input
             placeholder="Search clubs by name or interests..."
             value={searchTerm}
             onChange={(e) => setSearchTerm(e.target.value)}
             className="pl-10"
           />
```

```
        </div>
        <div className="flex gap-2 flex-wrap">
          {categories.map((category) => (
            <Button
              key={category}
              variant={selectedCategory === category ? "default" : "outline"}
              size="sm"
              onClick={() => setSelectedCategory(category)}
            >
              {category}
            </Button>
          ))}
        </div>
        <Button variant="outline" size="sm">
          <Filter className="h-4 w-4 mr-2" />
          Filters
        </Button>
      </div>

      {/* Results Count */}
      <div className="flex justify-between items-center mb-6">
        <p className="text-muted-foreground">
          Showing {filteredClubs.length} clubs
        </p>
        {/* Temporary test button */}
        {/* <Button onClick={createTestClub} variant="outline" size="sm">
          Create Test Club
        </Button> */}
      </div>

      {/* Loading State */}
      {loading && <LoadingSpinner />}

      {/* Clubs Grid */}
      {!loading && (
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
          {filteredClubs.length === 0 ? (
            <div className="col-span-full text-center py-12">
              <Users className="h-12 w-12 mx-auto text-muted-foreground mb-4" />
              <h3 className="text-lg font-semibold mb-2">No clubs found</h3>
              <p className="text-muted-foreground">
                {searchTerm
                  ? "Try adjusting your search terms"
                  : "No clubs have been created yet"}
              </p>
```

```
            </div>
          ) : (
            filteredClubs.map((club) => (
              <Card
                key={club.id}
                className="group hover:shadow-lg transition-all duration-300
border-0 bg-gradient-card"
              >
                <div className="relative">
                  <div className="w-full h-48 rounded-t-lg overflow-hidden">
                    {club.club_image_url ? (
                      <>
                        <img
                          src={club.club_image_url}
                          alt={`${club.name} club image`}
                          className="w-full h-full object-cover"
                          onError={(e) => {
                            // Fallback to placeholder if image fails to load
                            const target = e.target as HTMLImageElement;
                            target.style.display = 'none';
                            target.nextElementSibling?.classList.remove('hidden');
                          }}
                        />
                        <div className="w-full h-full bg-gradient-to-br
from-green-500 to-blue-600 flex items-center justify-center hidden">
                          <Users className="h-12 w-12 text-white" />
                        </div>
                      </>
                    ) : (
                      <div className="w-full h-full bg-gradient-to-br
from-green-500 to-blue-600 flex items-center justify-center">
                        <Users className="h-12 w-12 text-white" />
                      </div>
                    )}
                  </div>
                  {/* Club Logo - Small display in top-left */}
                  {club.club_logo_url && (
                    <div className="absolute top-4 left-4 w-12 h-12 rounded-lg
overflow-hidden border-2 border-background/80 bg-background/80 shadow-sm">
                      <img
                        src={club.club_logo_url}
                        alt={`${club.name} logo`}
                        className="w-full h-full object-cover"
                        onError={(e) => {
                          const target = e.target as HTMLImageElement;
```

```
                        target.style.display = 'none';
                      }}
                    />
                  </div>
                )}

                <div className="absolute top-4 right-4">
                  <Badge variant="secondary" className="bg-background/80">
                    {club.is_public ? "Public" : "Private"}
                  </Badge>
                </div>

                {/* Subscription indicator */}
                {user && (() => {
                  const isSubscribed = isSubscribedToClub(club.id);
                  return isSubscribed ? (
                    <div className={`absolute top-4 ${club.club_logo_url ?
'left-20' : 'left-4'}`}>
                      <Badge variant="default" className="bg-green-600
hover:bg-green-700">
                        <Bell className="h-3 w-3 mr-1" />
                        Subscribed
                      </Badge>
                    </div>
                  ) : null;
                })()}
              </div>

              <div className="p-6">
                <div className="flex items-start justify-between mb-3">
                  <h3 className="text-xl font-semibold group-hover:text-primary
transition-colors">
                    <Link
                      to={`/club-details/${club.id}`}
                      className="hover:underline cursor-pointer"
                    >
                      {club.name}
                    </Link>
                  </h3>
                </div>

                <p className="text-muted-foreground mb-4 line-clamp-2">
                  {club.description || "No description available"}
                </p>
```

```
                    <div className="flex items-center justify-between mb-4 text-sm
text-muted-foreground">
                      <div className="flex items-center gap-1">
                        <Calendar className="h-4 w-4" />
                        <span>
                          Created{" "}
                          {new Date(club.created_at).toLocaleDateString()}
                        </span>
                      </div>
                    </div>

                    {/* Club Stats */}
                    <div className="flex items-center justify-between mb-4 text-sm
text-muted-foreground">
                      <div className="flex items-center gap-4">
                        <div className="flex items-center gap-1">
                          <Users className="h-4 w-4" />
                          <span>{clubStats[club.id]?.members || 0} members</span>
                        </div>
                        <div className="flex items-center gap-1">
                          <Bell className="h-4 w-4" />
                          <span>{clubStats[club.id]?.subscribers || 0}
subscribers</span>
                        </div>
                      </div>
                    </div>

                    <div className="flex gap-2">
                      <Button asChild className="flex-1">
                        <Link
                          to={`/join-club/${club.id}`}
                          onClick={() => {
                            console.log("Clubs - Navigating to club:", {
                              clubId: club.id,
                              clubName: club.name,
                              fullPath: `/join-club/${club.id}`,
                            });
                          }}
                        >
                          Join Club
                        </Link>
                      </Button>

                      {/* Subscribe/Unsubscribe Button */}
                      {user && (() => {
```

```
                      const isSubscribed = isSubscribedToClub(club.id);
                      return (
                        <Button
                          variant={isSubscribed ? "default" : "outline"}
                          size="icon"
                          onClick={() => handleSubscriptionToggle(club.id)}
                          disabled={loadingStats[club.id]}
                          className={isSubscribed ? "bg-green-600
hover:bg-green-700" : ""}
                          title={isSubscribed ? "Unsubscribe from club updates" :
"Subscribe to club updates"}
                        >
                          {loadingStats[club.id] ? (
                            <div className="h-4 w-4 animate-spin rounded-full
border-2 border-current border-t-transparent" />
                          ) : isSubscribed ? (
                            <BellOff className="h-4 w-4" />
                          ) : (
                            <Bell className="h-4 w-4" />
                          )}
                        </Button>
                      );
                    })()}

                    <Button variant="outline" size="icon">
                      <Star className="h-4 w-4" />
                    </Button>
                  </div>
                </div>
              </Card>
            ))
          )}
        </div>
      )}



      {/* Call to Action */}
      <div className="text-center mt-12">
        <Card className="p-8 bg-gradient-accent border-0">
          <h2 className="text-2xl font-bold text-accent-foreground mb-4">
            Can't find what you're looking for?
          </h2>
          <p className="text-accent-foreground/80 mb-6">
            Start your own club and bring together students with similar
```

```
            interests
          </p>
          <Button variant="secondary" size="lg">
            Contact Us
          </Button>
        </Card>
      </div>
    </div>
  </Layout>
 );
};

export default Clubs;
```

Events: In the events page, the user can see the upcoming events in detail like when and where it will start, when it will end etc.

```
import { useState } from "react";
import { Calendar, Clock, MapPin, Users, Filter, Search, Star } from
"lucide-react";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Card } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Link } from "react-router-dom";
import Layout from "@/components/Layout";
import EventManagement from "@/components/EventManagement";
import { useAuth } from "@/context/AuthContext";
import { useEvents } from "@/hooks/useDatabase";
import { LoadingSpinner } from "@/components/LoadingSpinner";

  return (
    <Layout>

      <div className="container mx-auto px-4 py-8">
        {/* Staff Event Management */}
        {user?.role === 'staff' && (
          <div className="mb-12">
            <EventManagement
              events={events}
              onEventCreate={handleEventCreate}
```

```
              onEventUpdate={handleEventUpdate}
              onEventDelete={handleEventDelete}
            />
          </div>
        )}

        {/* Header Section */}
        <div className="text-center mb-12">
          <h1 className="text-4xl font-bold bg-gradient-primary
bg-clip-text text-transparent mb-4">
            {user?.role === 'staff' ? 'All Campus Events' : 'University
Events'}
          </h1>
          <p className="text-xl text-muted-foreground max-w-2xl
mx-auto">
            {user?.role === 'staff'
              ? 'View and manage all campus events, workshops, and
activities'
              : 'Discover exciting events, workshops, competitions, and
activities happening on campus'
            }
          </p>
        </div>

        {/* Event Management for Staff */}
        {user?.role === 'faculty' && (
          <div className="mb-8">
            <EventManagement
              onEventCreate={handleEventCreate}
              onEventUpdate={handleEventUpdate}
              onEventDelete={handleEventDelete}
            />
          </div>
        )}

        {/* Search and Filter */}
        <div className="flex flex-col md:flex-row gap-4 mb-8">
          <div className="relative flex-1">
            <Search className="absolute left-3 top-1/2 transform
-translate-y-1/2 text-muted-foreground h-4 w-4" />
            <Input
              placeholder="Search events..."
              value={searchTerm}
```

```
              onChange={(e) => setSearchTerm(e.target.value)}
              className="pl-10"
            />
          </div>
          <div className="flex gap-2 flex-wrap">
            {eventTypes.map((type) => (
              <Button
                key={type}
                variant={selectedType === type ? "default" : "outline"}
                size="sm"
                onClick={() => setSelectedType(type)}
              >
                {type}
              </Button>
            ))}
          </div>
          <Button variant="outline" size="sm">
            <Filter className="h-4 w-4 mr-2" />
            More Filters
          </Button>
        </div>


        {/* Call to Action */}
        <div className="text-center mt-12">
          <Card className="p-8 bg-gradient-accent border-0">
            <h2 className="text-2xl font-bold text-accent-foreground
mb-4">
              Have an event idea?
            </h2>
            <p className="text-accent-foreground/80 mb-6">
              Submit your event proposal and engage the university
community
            </p>
            <Button variant="secondary" size="lg">
              Submit Event Proposal
            </Button>
          </Card>
        </div>
      </div>
    </Layout>
  );
};
```

```
export default Events;
```

Sign in: In the sign in page, the user will input their credentials like username and password to sign in to the SAM Portal

```
import { useState } from "react";
import { Eye, EyeOff, Mail, Lock, ArrowRight } from "lucide-react";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Card } from "@/components/ui/card";
import { Label } from "@/components/ui/label";
import { Checkbox } from "@/components/ui/checkbox";
import { Link, useNavigate } from "react-router-dom";
import { useAuth } from "@/context/AuthContext";
import { useToast } from "@/hooks/use-toast";
import Layout from "@/components/Layout";

  return (
    <Layout>
      <div className="min-h-screen flex items-center justify-center
bg-gradient-to-br from-background via-background to-accent/10 p-4">
      <div className="w-full max-w-md">
        {/* Header */}
        <div className="text-center mb-8">
          <Link to="/" className="inline-block">
            <h1 className="text-3xl font-bold bg-gradient-primary
bg-clip-text text-transparent">
              BRACU Student Activity Portal
            </h1>
          </Link>
          <p className="text-muted-foreground mt-2">
            Welcome back to BRAC University Student Activity Portal
          </p>
        </div>

        {/* Sign In Form */}
        <Card className="p-8 border-0 bg-gradient-card shadow-lg">
          <form onSubmit={handleSubmit} className="space-y-6">
            {/* Email Field */}
            <div className="space-y-2">
              <Label htmlFor="email">Email Address</Label>
```

```
              <div className="relative">
                <Mail className="absolute left-3 top-1/2 transform
-translate-y-1/2 text-muted-foreground h-4 w-4" />
                                  <Input
                    id="email"
                    type="email"
                    placeholder="your.email@g.bracu.ac.bd"
                    value={email}
                    onChange={(e) => setEmail(e.target.value)}
                    className="pl-10"
                    required
                  />
              </div>
            </div>

            {/* Password Field */}
            <div className="space-y-2">
              <Label htmlFor="password">Password</Label>
              <div className="relative">
                <Lock className="absolute left-3 top-1/2 transform
-translate-y-1/2 text-muted-foreground h-4 w-4" />
                <Input
                  id="password"
                  type={showPassword ? "text" : "password"}
                  placeholder="Enter your password"
                  value={password}
                  onChange={(e) => setPassword(e.target.value)}
                  className="pl-10 pr-10"
                  required
                />
                <button
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="absolute right-3 top-1/2 transform
-translate-y-1/2 text-muted-foreground hover:text-foreground"
                >
                  {showPassword ? <EyeOff className="h-4 w-4" /> : <Eye
className="h-4 w-4" />}
                </button>
              </div>
            </div>

          {/* Sign Up Link */}
```

```
          <div className="mt-6 text-center">
            <p className="text-sm text-muted-foreground">
              Don't have an account?{" "}
              <Link
                to="/signup"
                className="text-primary hover:text-primary/80
font-medium"
              >
                Sign up for free
              </Link>
            </p>
          </div>
        </Card>



        {/* Footer */}
        <div className="text-center mt-8">
          <p className="text-xs text-muted-foreground">
            By signing in, you agree to our{" "}
            <Link to="/terms" className="text-primary
hover:text-primary/80">
              Terms of Service
            </Link>{" "}
            and{" "}
            <Link to="/privacy" className="text-primary
hover:text-primary/80">
              Privacy Policy
            </Link>
          </p>
        </div>
      </div>
      </div>
    </Layout>
  );
};

export default SignIn;
```

AI assistant: In this part, the user can interact with an AI assistant, the user can ask about everything about the SAM Portal happenings and also can explore outside of the portal things.

```
import { useEffect, useRef, useState } from "react";
// Remove ChatCompletionMessage import, not compatible with HuggingFace
router
import { Card, CardContent, CardHeader, CardTitle } from
"@/components/ui/card";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { Badge } from "@/components/ui/badge";
import { useToast } from "@/hooks/use-toast";
import { useAuth } from "@/context/AuthContext";
import { supabase } from "@/lib/supabase";
import { Loader2, Send, Bot, User } from "lucide-react";
import { OpenAI } from "openai";

import Layout from "@/components/Layout";

  return (
    <Layout>
      <div className="container mx-auto px-4 py-6">
        <div className="mb-4 flex items-center gap-2">
          <Bot className="h-6 w-6" />
          <h2 className="text-2xl font-semibold">AI Assistant</h2>
          {contextLoading ? (
            <Badge variant="secondary">Loading context…</Badge>
          ) : (
            <Badge variant="outline">Context ready plus OpenAI</Badge>
          )}
          {usingAI && (
            <Badge variant="default" className="bg-blue-500">
              🤖 AI Powered
            </Badge>
          )}
        </div>
        <Card>
          <CardHeader>
            <CardTitle>Ask about clubs, events, and careers</CardTitle>
          </CardHeader>
          <CardContent>
            <div
              ref={listRef}
              className="h-[50vh] overflow-y-auto space-y-4 pr-2"
            >
              {messages.length === 0 ? (
```

```
                <div className="space-y-4">
                  <div className="flex items-start gap-2">
                    <Bot className="h-5 w-5 mt-1 text-muted-foreground"
/>
                    <div className="rounded-md px-3 py-2 text-sm
bg-accent/50">
                      👋 Hello! I'm your university assistant. I can
help you
                      with:
                      <br />
                      <br />• <strong>Clubs:</strong> Find and join
student
                      organizations
                      <br />• <strong>Events:</strong> Discover
upcoming
                      activities and workshops
                      <br />• <strong>Careers:</strong> Explore job
                      opportunities and internships
                      <br />• <strong>Resources:</strong> Access study
materials
                      and templates
                      <br />
                      <br />
                      What would you like to know about?
                    </div>
                  </div>
                </div>
              ) : (
                messages.map((m) => (

              <div className="flex gap-2">
                <Input
                  placeholder="Ask me about a club, event, or job…"
                  value={input}
                  onChange={(e) => setInput(e.target.value)}
                  onKeyDown={(e) => {
                    if (e.key === "Enter") handleSend();
                  }}
                />
                <Button onClick={handleSend} disabled={loading}>
                  <Send className="h-4 w-4" />
                </Button>
              </div>
```

```
            </div>
          </CardContent>
        </Card>
      </div>
    </Layout>
  );
};

export default Assistant;
```

Resources: In this page, the user will find resources of the courses that are taught in university.

```
import { useState } from "react";
import {
  Search,
  Download,
  Eye,
  BookOpen,
  FileText,
  Video,
  Image,
  Upload,
  Plus,
  Archive,
  Hash,
  Calendar,
  ExternalLink,
  Edit,
  Trash2,
  MoreVertical,
} from "lucide-react";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Card } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Tabs, TabsContent, TabsList, TabsTrigger } from
"@/components/ui/tabs";
import {
  Select,
  SelectContent,
  SelectItem,
  SelectTrigger,
```

```
  SelectValue,
} from "@/components/ui/select";
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";
import Layout from "@/components/Layout";
import UploadResourceDialog from "@/components/UploadResourceDialog";
import EditResourceDialog from "@/components/EditResourceDialog";
import ResourcePreviewDialog from "@/components/ResourcePreviewDialog";
import { LoadingSpinner } from "@/components/LoadingSpinner";
import { useResources } from "@/hooks/useDatabase2";
import { useAuth } from "@/context/AuthContext";
import type { Resource } from "@/lib/supabase";

  return (
    <Layout>
      <div className="container mx-auto px-4 py-8">
        {/* Header Section */}
        <div className="text-center mb-12">
          <div className="flex items-center justify-center gap-3 mb-4">
            <BookOpen className="h-10 w-10 text-primary" />
            <h1 className="text-4xl font-bold bg-gradient-primary
bg-clip-text text-transparent">
              Course Resource Repository
            </h1>
          </div>
          <p className="text-xl text-muted-foreground max-w-3xl mx-auto
mb-6">
            Digital library for non-academic course materials and
educational
            resources. Share knowledge and access resources contributed
by your
            fellow students.
          </p>
          {isAuthenticated && (
            <Button
              onClick={() => setUploadDialogOpen(true)}
              size="lg"
              className="gap-2"
            >
```

```
              <Plus className="h-5 w-5" />
              Contribute Resource
            </Button>
          )}
        </div>

        {/* Search and Filters */}
        <div className="space-y-4 mb-8">
          <div className="relative">
            <Search className="absolute left-3 top-1/2 transform
-translate-y-1/2 text-muted-foreground h-4 w-4" />
            <Input
              placeholder="Search resources by title or description..."
              value={searchTerm}
              onChange={(e) => setSearchTerm(e.target.value)}
              className="pl-10"
            />
          </div>

          <div className="grid grid-cols-1 md:grid-cols-4 gap-4">
            <Select
              value={selectedCategory}
              onValueChange={setSelectedCategory}
            >
              <SelectTrigger>
                <SelectValue placeholder="Category" />
              </SelectTrigger>
              <SelectContent>
                {categories.map((category) => (
                  <SelectItem key={category} value={category}>
                    {category}
                  </SelectItem>
                ))}
              </SelectContent>
            </Select>

            <Select value={selectedSubject}
onValueChange={setSelectedSubject}>
              <SelectTrigger>
                <SelectValue placeholder="Subject" />
              </SelectTrigger>
              <SelectContent>
                {subjects.map((subject) => (
```

```
                <SelectItem key={subject} value={subject}>
                  {subject}
                </SelectItem>
              ))}
            </SelectContent>
          </Select>

          <Select value={selectedType}
onValueChange={setSelectedType}>
            <SelectTrigger>
              <SelectValue placeholder="File Type" />
            </SelectTrigger>
            <SelectContent>
              {resourceTypes.map((type) => (
                <SelectItem key={type} value={type}>
                  {type}
                </SelectItem>
              ))}
            </SelectContent>
          </Select>

          <Button
            variant="outline"
            onClick={() => {
              setSearchTerm("");
              setSelectedCategory("All");
              setSelectedSubject("All");
              setSelectedType("All");
            }}
          >
            Clear Filters
          </Button>
        </div>
      </div>

      {/* Statistics Section */}
      <div className="grid grid-cols-1 md:grid-cols-4 gap-4 mb-6">
        <Card className="p-4 text-center">
          <div className="text-2xl font-bold text-primary">
            {resources.length}
          </div>
          <div className="text-sm text-muted-foreground">Total
Resources</div>
```

```
          </Card>
          <Card className="p-4 text-center">
            <div className="text-2xl font-bold text-green-600">
              {resources.reduce(
                (total, r) => total + (r.download_count || 0),
                0
              )}
            </div>
            <div className="text-sm text-muted-foreground">Total
Downloads</div>
          </Card>
          <Card className="p-4 text-center">
            <div className="text-2xl font-bold text-blue-600">
              {formatFileSize(
                resources.reduce((total, r) => total + (r.file_size ||
0), 0)
              )}
            </div>
            <div className="text-sm text-muted-foreground">Total
Storage</div>
          </Card>
          <Card className="p-4 text-center">
            <div className="text-2xl font-bold text-purple-600">
              {resources.filter((r) =>
r.file_type?.includes("pdf")).length}
            </div>
            <div className="text-sm text-muted-foreground">PDF
Documents</div>
          </Card>
        </div>


        {/* Call to Action Section */}
        {isAuthenticated && (
          <div className="text-center mt-12">
            <Card className="p-8 bg-gradient-accent border-0">
              <Upload className="h-12 w-12 mx-auto
text-accent-foreground mb-4" />
              <h2 className="text-2xl font-bold text-accent-foreground
mb-4">
                Share Your Knowledge
              </h2>
```

```
              <p className="text-accent-foreground/80 mb-6 max-w-2xl
mx-auto">
                Help your fellow students succeed by contributing your
study
                materials, project reports, useful tools, and
educational
                resources to our community library.
              </p>
              <Button
                variant="secondary"
                size="lg"
                onClick={() => setUploadDialogOpen(true)}
                className="gap-2"
              >
                <Plus className="h-5 w-5" />
                Upload Resource
              </Button>
            </Card>
          </div>
        )}
      </div>

      {/* Dialogs */}
      <UploadResourceDialog
        open={uploadDialogOpen}
        onOpenChange={setUploadDialogOpen}
        onUploadSuccess={() => {
          console.log("DEBUG: Upload success, triggering refetch...");
          refetch();
        }}
      />

      <ResourcePreviewDialog
        resource={previewResource}
        open={previewDialogOpen}
        onOpenChange={setPreviewDialogOpen}
        onDownload={handleDownload}
        getPreviewUrl={getResourcePreviewUrl}
      />

      <EditResourceDialog
        open={editDialogOpen}
        onOpenChange={setEditDialogOpen}
```

```
        resource={editingResource}
        onEditSuccess={() => {
          console.log("DEBUG: Edit success, triggering refetch...");
          refetch();
        }}
      />
    </Layout>
  );
};

export default Resources;
```

Chat: In this part, the user can communicate with another person through personal text in real time.

```
import { useState, useRef, useEffect } from "react";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Card, CardContent, CardHeader, CardTitle } from
"@/components/ui/card";
import { Avatar, AvatarFallback, AvatarImage } from
"@/components/ui/avatar";
import { Badge } from "@/components/ui/badge";
import { ScrollArea } from "@/components/ui/scroll-area";
import { Separator } from "@/components/ui/separator";
import {
  Send,
  MoreVertical,
  Search,
  Plus,
  Users,
  Hash,
  Paperclip,
  Image,
  MessageSquare,
  X,
  Upload
} from "lucide-react";
import Layout from "@/components/Layout";
import { useAuth } from "@/context/AuthContext";
import { useChat2, type Conversation, type Message } from
"@/hooks/useChat2";
```

```
import { supabase } from "@/lib/supabase";
import { uploadImageToCloudinary } from "@/lib/cloudinary";
import {
  DropdownMenu,
  DropdownMenuTrigger,
  DropdownMenuContent,
  DropdownMenuItem,
} from "@/components/ui/dropdown-menu";
import {
  Dialog,
  DialogContent,
  DialogHeader,
  DialogTitle,
  DialogFooter,
} from "@/components/ui/dialog";
```

```
  return (
    <Layout>
      <div className="h-screen bg-background pt-16">
        <div className="h-full flex">
          {/* Sidebar */}
          <div className="w-80 border-r border-border bg-card flex
flex-col">
            {/* Header */}
            <div className="p-4 border-b border-border">
              <div className="flex items-center justify-between mb-4">
                <h2 className="text-xl font-semibold">Messages</h2>
                <Button
                  size="icon"
                  variant="ghost"
                  onClick={() => setShowNewConversation(true)}
                  aria-label="Start new chat"
                >
                  <Plus className="h-5 w-5" />
                </Button>
                <Button
                  size="sm"
                  variant="outline"
                  className="ml-2"
                  onClick={() => setShowGroupDialog(true)}
                >
                  New Group
```

```
                </Button>
              </div>


          {/* Chat Area */}
          <div className="flex-1 flex flex-col">
            {selectedConversation && currentConversation ? (
              <>
                {/* Chat Header */}
                <div className="p-4 border-b border-border bg-card">
                  <div className="flex items-center justify-between">
                    <div className="flex items-center space-x-3">
                      <Avatar className="h-10 w-10">
                        <AvatarImage
src={conversationAvatars[currentConversation.id] ||
currentConversation.avatar} />
                        <AvatarFallback className="bg-gradient-primary
text-primary-foreground">
                          {currentConversation.type === 'group' ? (
                            <Users className="h-5 w-5" />
                          ) : (
                            currentConversation.name.split(' ').map(n
=> n[0]).join('')
                          )}
                        </AvatarFallback>
                      </Avatar>
                      <div>
                        <h3 className="font-semibold">
                          {currentConversation.type === 'direct'
                            ? currentConversation.name
                            : currentConversation.name
                          }
                        </h3>
                        <p className="text-sm text-muted-foreground">
                          {currentConversation.type === 'group'
                            ? `${onlineCount} online`
                            : 'Active now'
                          }
                        </p>
                      </div>
                    </div>
                    <div className="flex items-center space-x-2">
                      <DropdownMenu>
```

```
                        <DropdownMenuTrigger asChild>
                          <Button size="icon" variant="ghost"
aria-label="Conversation menu">
                            <MoreVertical className="h-5 w-5" />
                          </Button>
                        </DropdownMenuTrigger>
                        <DropdownMenuContent align="end"
className="w-56">
                          <DropdownMenuItem
                            onClick={async () => {
                              if (!selectedConversation) return;
                              const list = await
fetchParticipantsWithProfiles(selectedConversation);
                              setMembers(list);
                              setShowMembersDialog(true);
                            }}
                          >
                            View/Add members
                          </DropdownMenuItem>
                          <DropdownMenuItem
                            onClick={() => {
                              if (!currentConversation) return;
                              setRenameValue(currentConversation.name);
                              setShowRenameDialog(true);
                            }}
                          >
                            Rename conversation
                          </DropdownMenuItem>
                          <DropdownMenuItem
                            onClick={() => setShowBlockDialog(true)}
                          >
                            Block user
                          </DropdownMenuItem>
                          <DropdownMenuItem
                            className="text-red-600"
                            onClick={() => setShowDeleteDialog(true)}
                          >
                            Remove chat
                          </DropdownMenuItem>
                        </DropdownMenuContent>
                      </DropdownMenu>
                    </div>
                  </div>
```

```
                </div>

                {/* Rename dialog */}
                <Dialog open={showRenameDialog}
onOpenChange={setShowRenameDialog}>
                  <DialogContent>
                    <DialogHeader>
                      <DialogTitle>Rename conversation</DialogTitle>
                    </DialogHeader>
                    <div className="mt-2">
                      <Input
                        autoFocus
                        value={renameValue}
                        onChange={(e) =>
setRenameValue(e.target.value)}
                        placeholder="Enter new name"
                      />
                    </div>
                    <DialogFooter>
                      <Button
                        variant="outline"
                        onClick={() => setShowRenameDialog(false)}
                      >
                        Cancel
                      </Button>
                      <Button
                        onClick={async () => {
                          if (selectedConversation &&
renameValue.trim()) {
                            const ok = await
renameConversation(selectedConversation, renameValue.trim());
                            if (ok) setShowRenameDialog(false);
                          }
                        }}
                      >
                        Save
                      </Button>
                    </DialogFooter>
                  </DialogContent>
                </Dialog>

                {/* Delete conversation dialog */}
```

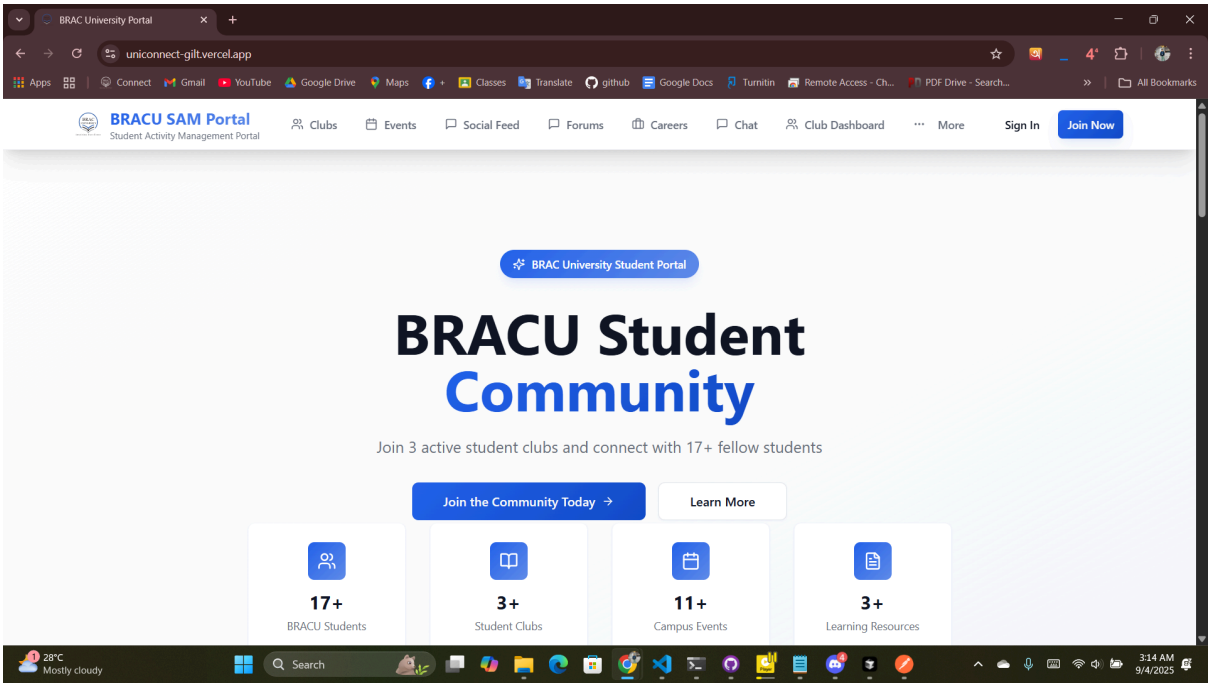
```
                <Dialog open={showDeleteDialog}
onOpenChange={setShowDeleteDialog}>
                  <DialogContent>
                    <DialogHeader>
                      <DialogTitle>Remove chat</DialogTitle>
                    </DialogHeader>
                    <p className="text-sm text-muted-foreground">
                      This will delete the conversation and its
messages. This action cannot be undone.
                    </p>
                    <DialogFooter>
                      <Button variant="outline" onClick={() =>
setShowDeleteDialog(false)}>Cancel</Button>
                      <Button
                        className="bg-red-600 hover:bg-red-700"
                        onClick={async () => {
                          if (selectedConversation) {
                            const ok = await
deleteConversation(selectedConversation);
                            if (ok) setShowDeleteDialog(false);
                          }
                        }}
                      >
                        Delete
                      </Button>
                    </DialogFooter>
                  </DialogContent>
                </Dialog>

                    {/* Image Upload Button */}
                    <div className="relative">
                      <input
                        type="file"
                        accept="image/*"
                        onChange={handleImageSelect}
                        className="absolute inset-0 w-full h-full
opacity-0 cursor-pointer"
                        id="image-upload"
                      />
                      <Button
                        size="icon"
                        variant="ghost"
```

```
                        className={selectedImage ? "text-blue-600" :
""}
                        disabled={isUploadingImage}
                      >
                      <Image className="h-5 w-5" />
                    </Button>
                     </div>

                    <div className="flex-1 relative">
                      <Input
                        placeholder={selectedImage ? "Add a message
(optional)..." : "Type a message..."}
                       value={message}
                       onChange={(e) => setMessage(e.target.value)}
                       onKeyDown={(e) => e.key === 'Enter' &&
!e.shiftKey && handleSendMessage()}
                       className="pr-10"
                        disabled={isUploadingImage}
                      />
                    </div>

                     <Button
                       onClick={handleSendMessage}
                       className="bg-gradient-hero"
                       disabled={isUploadingImage || (!message.trim()
&& !selectedImage)}
                      >
                       {isUploadingImage ? (
                         <div className="h-4 w-4 animate-spin
rounded-full border-2 border-white border-t-transparent" />
                       ) : (
                      <Send className="h-4 w-4" />
                       )}
                    </Button>
                  </div>
                </div>
              </>
            ) : (
              <div className="flex-1 flex items-center justify-center">
                <div className="text-center">
                  <MessageSquare className="h-16 w-16
text-muted-foreground mx-auto mb-4" />
```

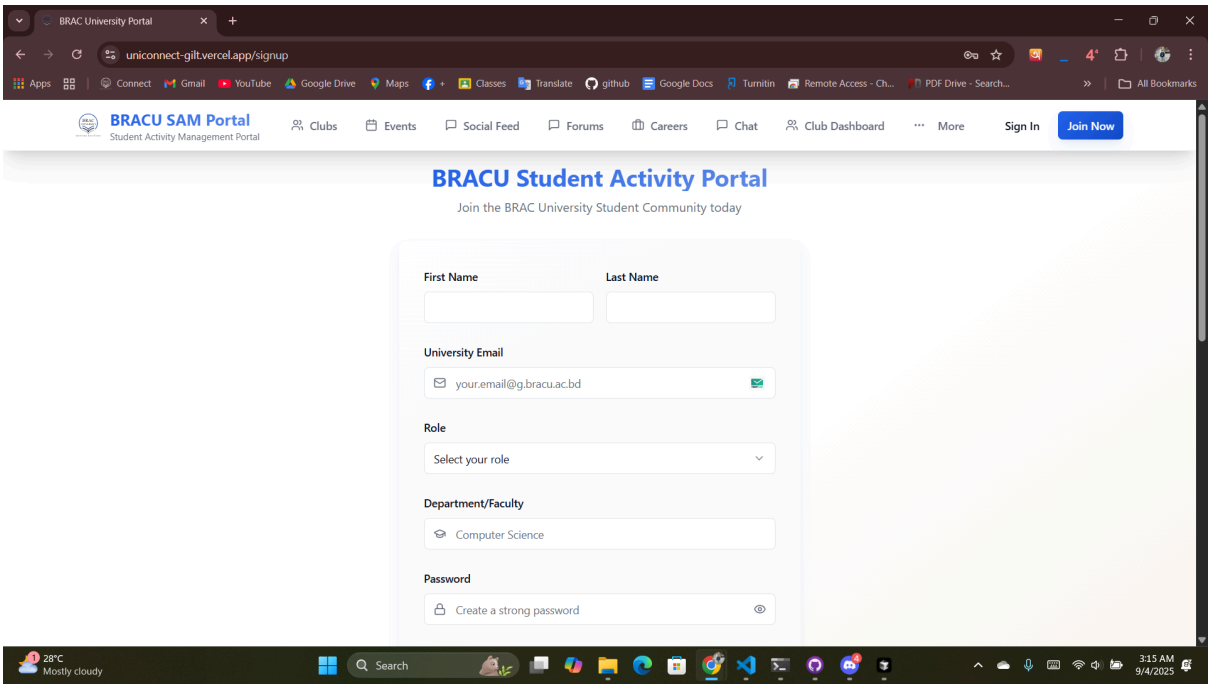
```
                  <h3 className="text-lg font-semibold mb-2">Welcome to
Chat!</h3>
                  <p className="text-muted-foreground mb-4">
                    {conversations.length === 0
                      ? "Create your first conversation to start
messaging"
                      : "Select a conversation from the sidebar to
start messaging"
                    }
                  </p>
                  {conversations.length === 0 && (
                    <Button
                      onClick={() => setShowNewConversation(true)}
                      className="bg-gradient-hero"
                    >
                      <Plus className="h-4 w-4 mr-2" />
                      Start New Chat
                    </Button>
                  )}
                </div>
              </div>
            )}
          </div>
        </div>
      </div>
    </Layout>
  );
};

export default Chat;
```

7. User Manual

1. Visit the BRACU SAM Portal homepage and click "Join Now" to start. Explore sections like "Clubs", "Campus Events" and "Learning Resources" by clicking the respective tiles. Click "Join the Community Today" or "Learn More" to proceed with registration.

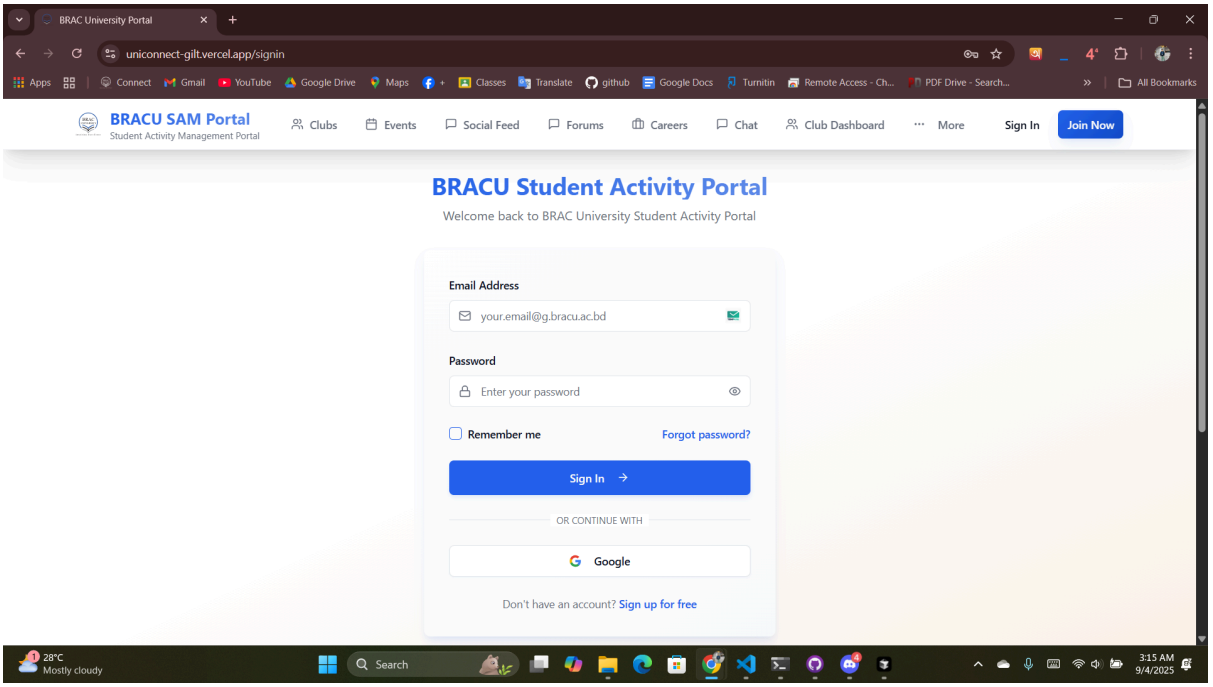


2. Click "Join Now" from the homepage of the BRACU SAM Portal. Fill in your First Name, Last Name, University Email, select your Role and Student/Faculty, give you department name and create a strong Password. Click "Sign Up" to create your account.

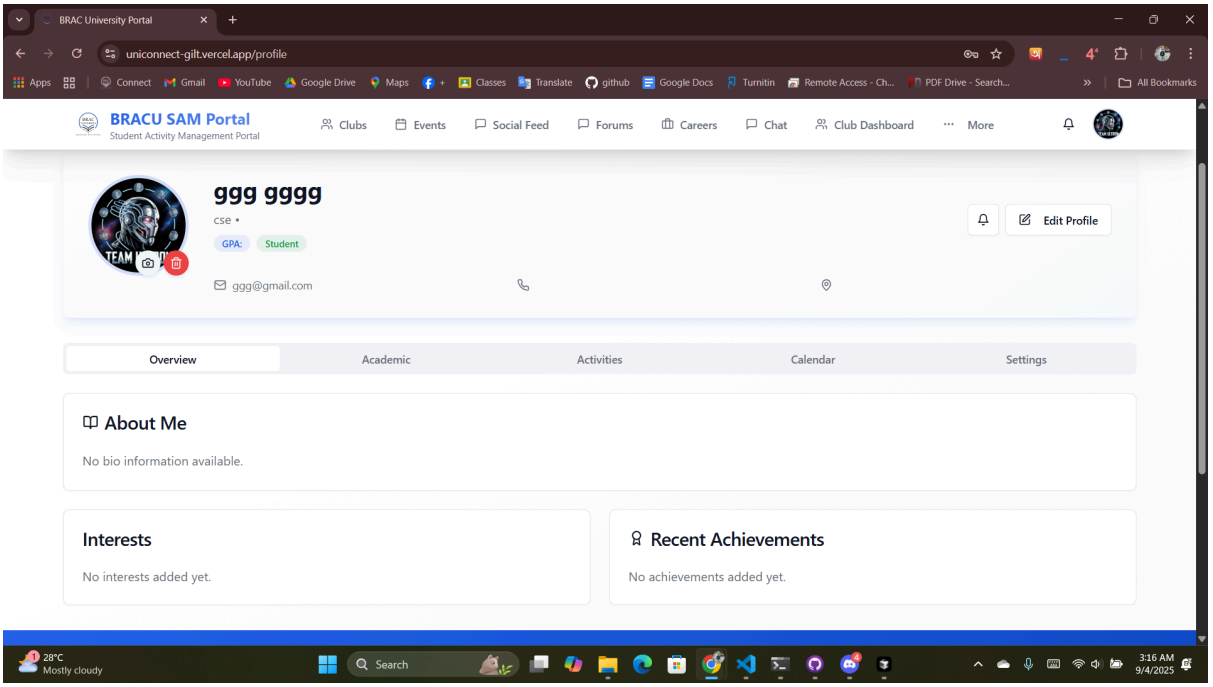


3. Navigate to the BRACU SAM Portal and click "Sign In" from the top menu. Enter your Email Address (e.g., your.email@bracu.ac.bd) and Password. Check "Remember

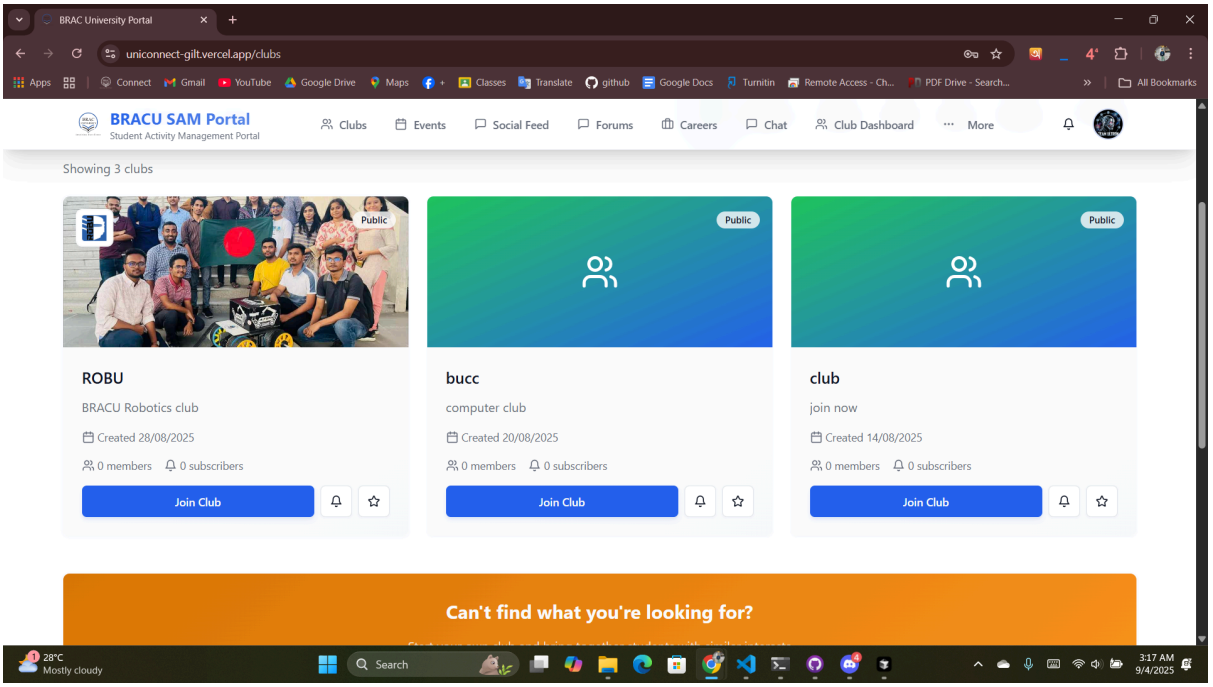
me" to stay logged in. Click "Sign In" to proceed. If new, click "Sign up for free" to register.



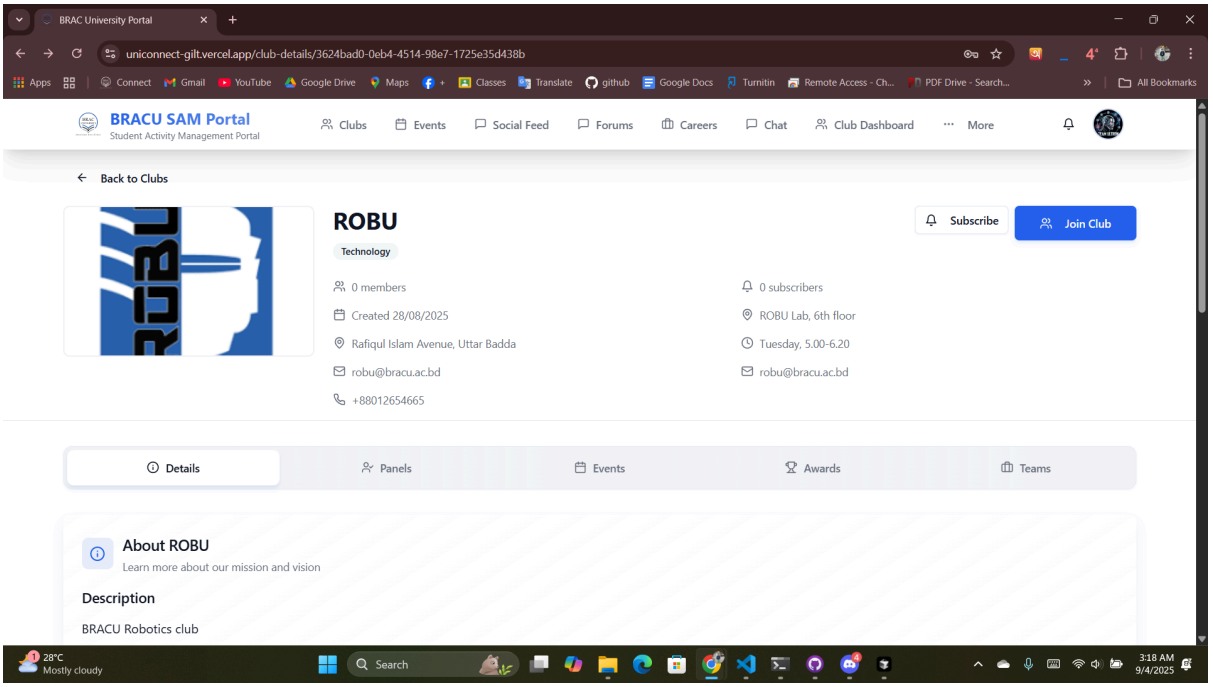
4. Access your profile by clicking your profile icon or "Profile" from the top menu on the BRACU SAM Portal. View or edit details such as your name ("Anik"), email (anik@gmail.com), and GPA under the "Overview" tab. Add information in "About Me" or "Interests" sections. Use the "Academic," "Activities," "Calendar," or "Settings" tabs to manage other profile aspects.



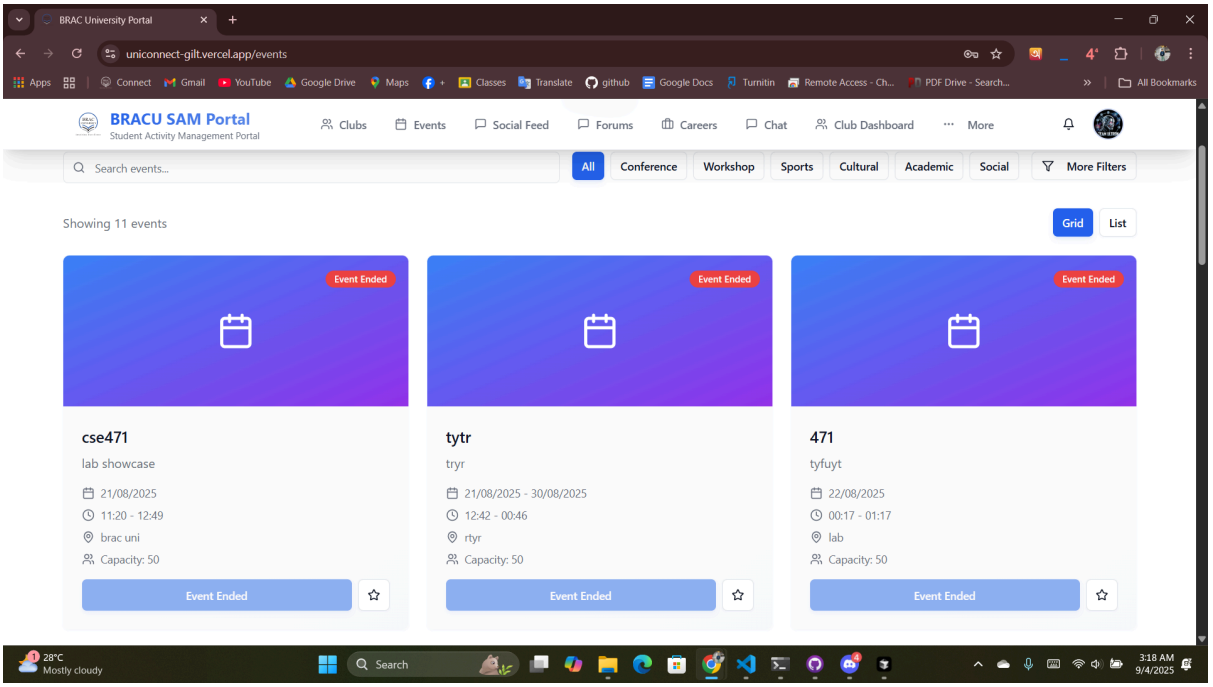
5. Go to "Clubs" from the top menu on the BRACU SAM Portal to view available clubs like "ROBU" "bucc"etc. Each club tile shows the name, creation date and member/subscriber count. Click "Join Club" on a tile (e.g., "ROBU") to become a member. If the user is not willing to join the club then they can just subscribe to the desired club to get notifications.



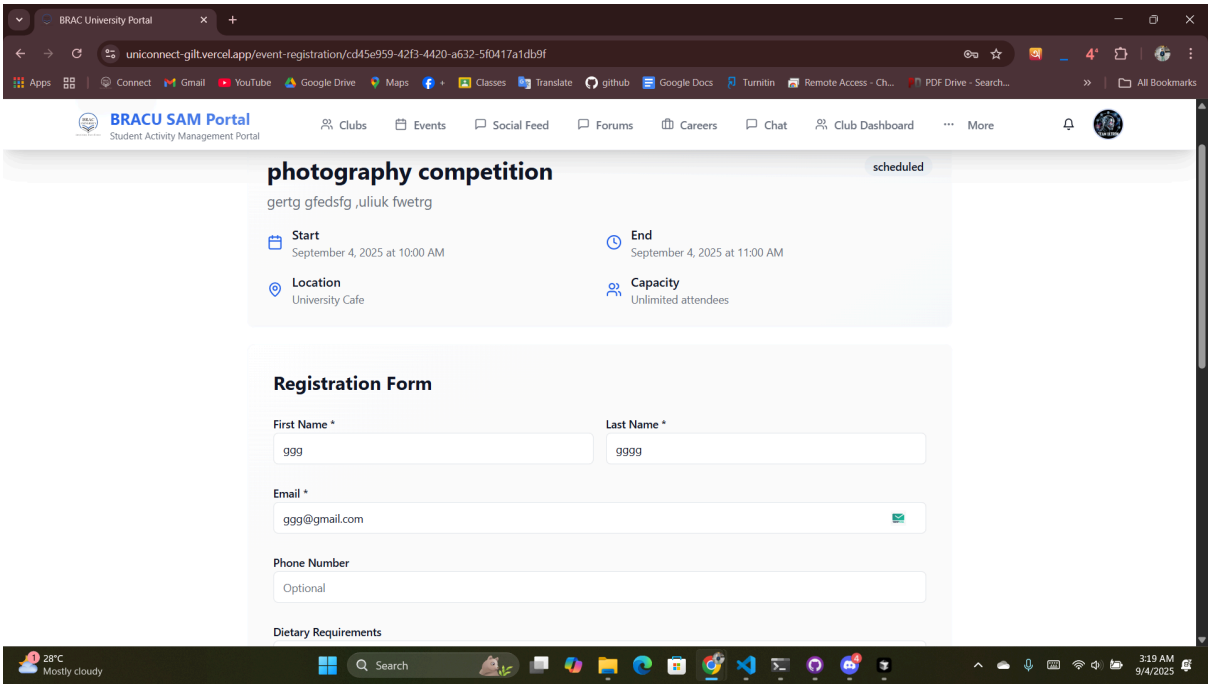
6. Navigate to "Clubs" select "ROBU" and click "Details" on the BRACU SAM Portal. Learn about the club's mission and vision under "About ROBU," and read the description ("BRACU Robotics club"). Note the location (ROBU Lab, 6th Floor), meeting time (Tuesday, 5:00 PM), and contact info (email: robu@bracu.ac.bd, phone: +88012654665). Click "Join Club" or "Subscribe" to get involved.



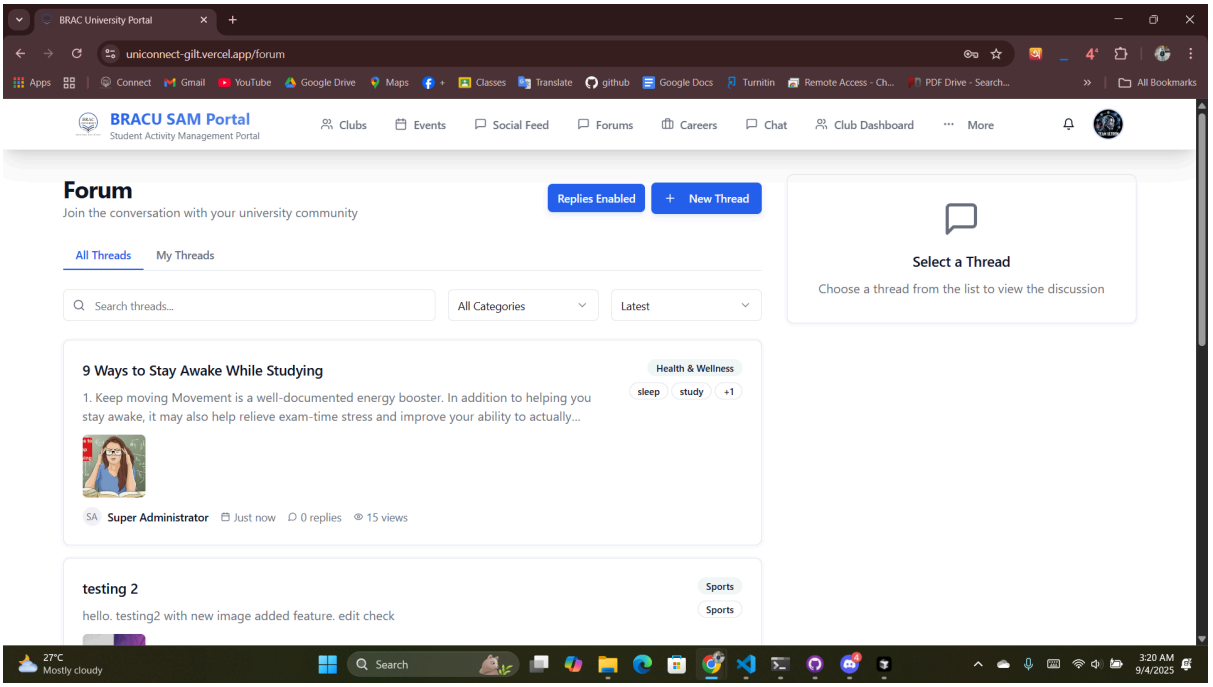
7. Navigate to "Events" from the top menu on the BRACU SAM Portal. Use the search bar to find specific events or filter by categories like Conference, Workshop, Sports, Cultural, Academic, or Social. The page lists events with details like dates, times, locations, and capacity(default at 50). Users can't register that are marked "Event Ended" or "Cancelled". Users can only register "Upcoming" or "Ongoing" events.



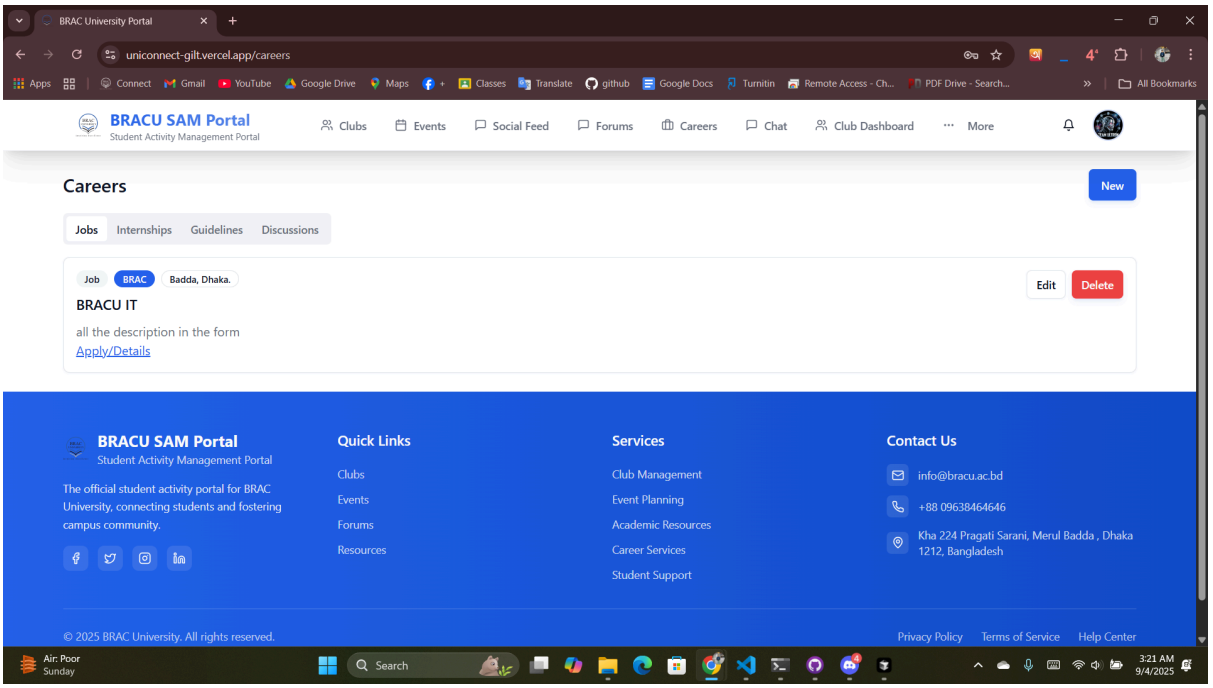
8. Navigate to the "Events" section from the top menu on the BRACU SAM Portal. Select the "photography competition" event to view details such as the start time (September 4, 2025, 10:00 AM) and end time (September 4, 2025, 11:00 AM), location (University Cafe). To participate, fill out the "Registration Form" by entering your First Name, Last Name, Email, Phone Number (optional), and any Dietary Requirements, then submit the form to register.



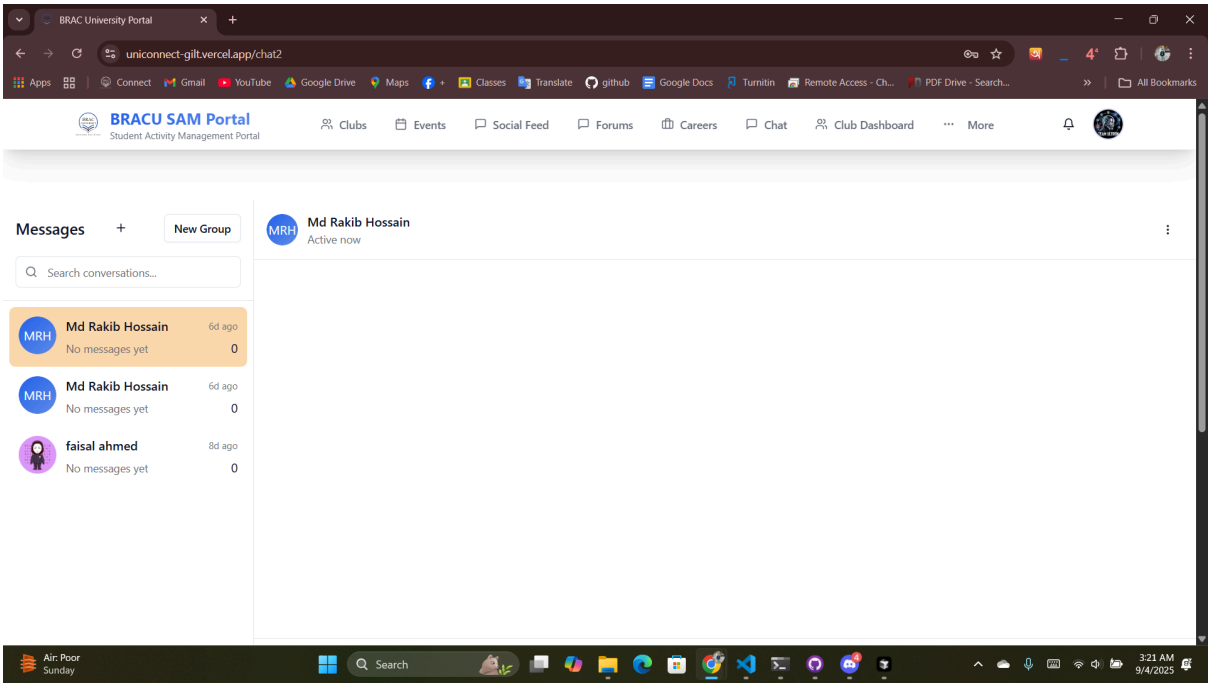
9. Navigate to "Forum" from the top menu on the BRACU SAM Portal. This page shows discussion threads (e.g., "Tips for Photography Competition") with posts from users. Users can create a new thread by clicking "Start Topic," reply to existing posts or use the search bar to find specific discussions, with a moderation note visible.



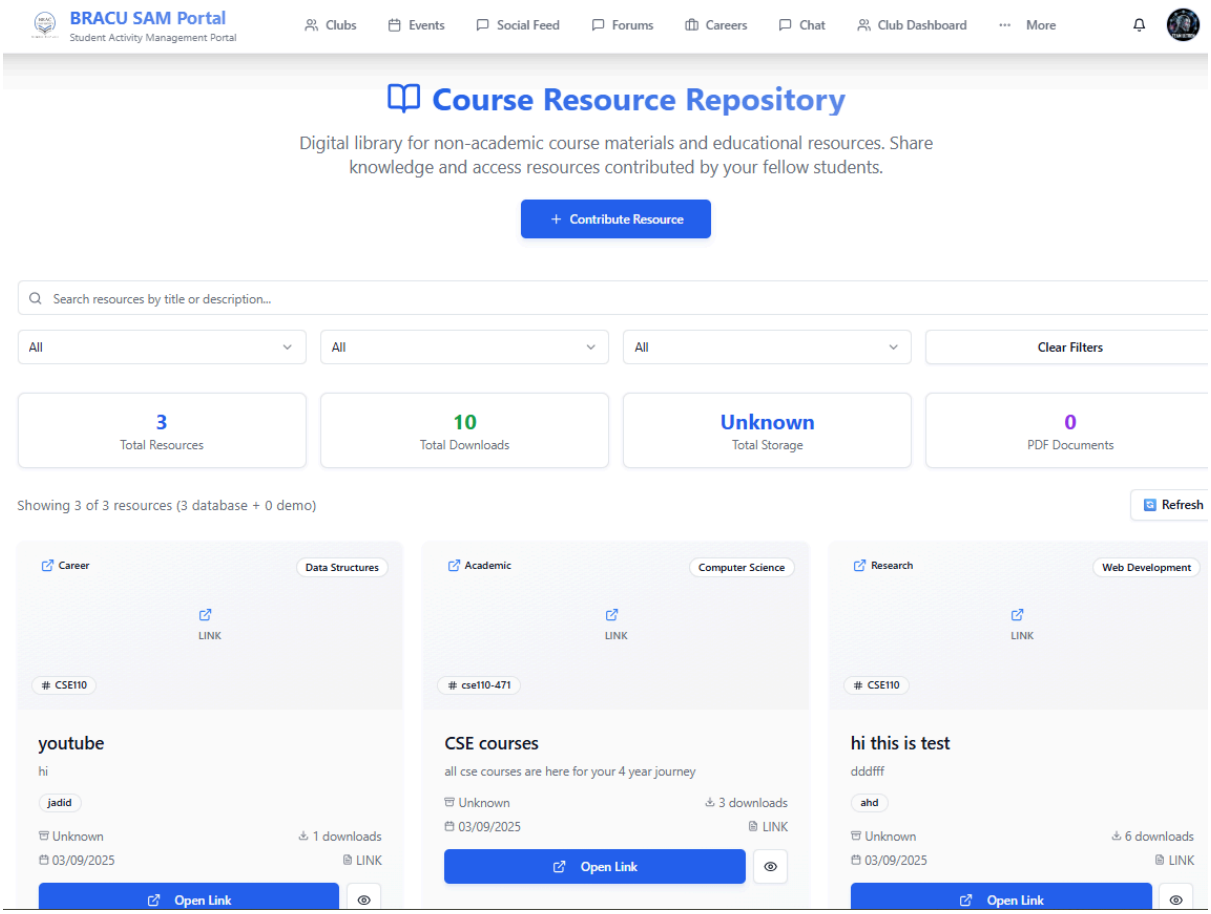
10. Navigate to "Career" from the top menu on the BRACU SAM Portal. This page provides career-related resources, including job listings, internship opportunities and guidelines and discussions. Users can apply for jobs from the given link or post a job with appropriate links.



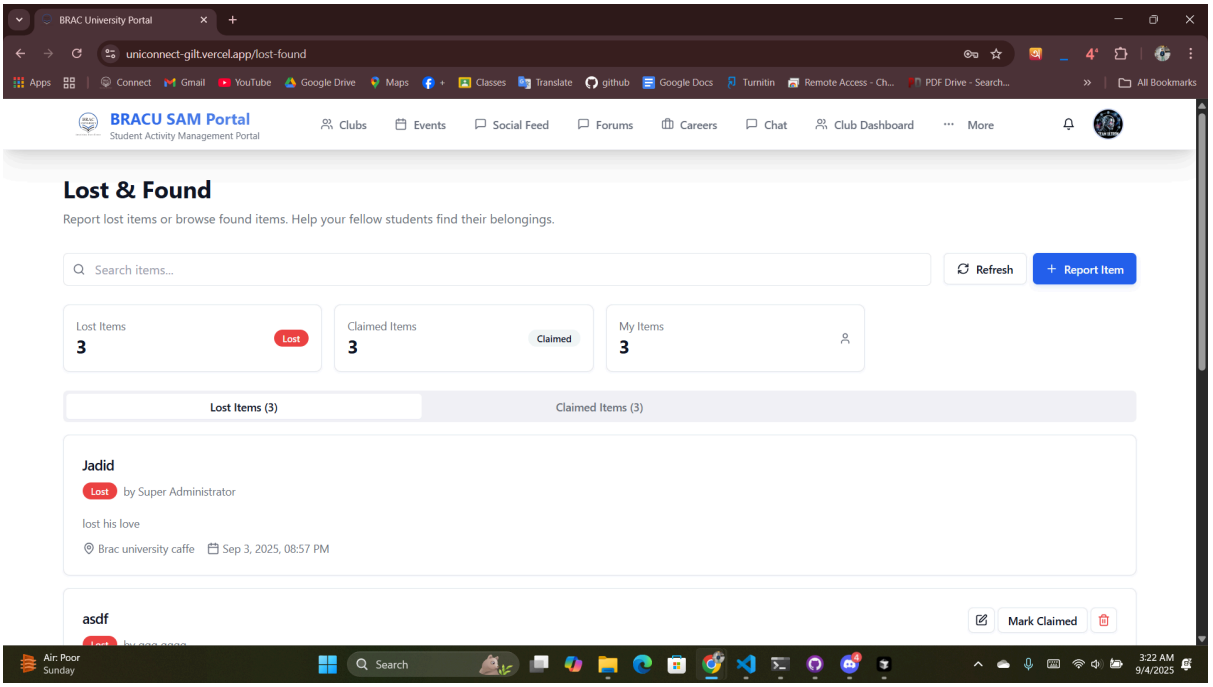
11. Go to " Chat" from the top menu on the BRACU SAM Portal. This page shows previous chats and search options to search for users and start a new chat. Users can join a room, add a user in the room, send messages, upload images, leave the chat and block a user.



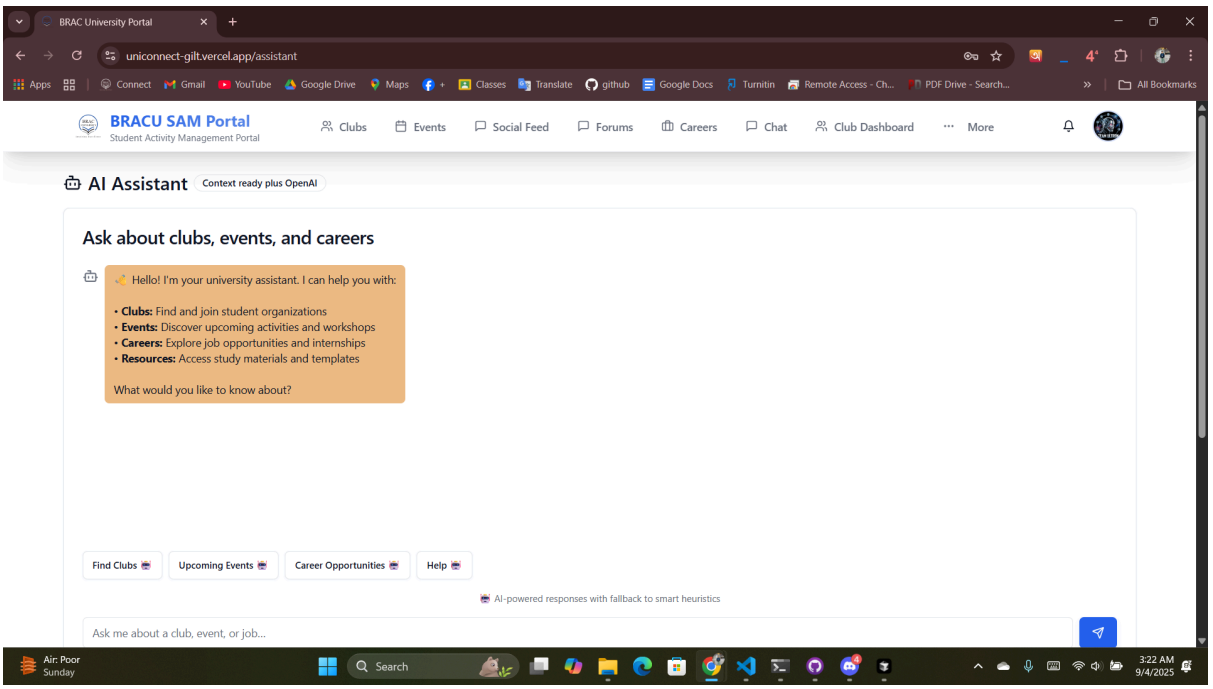
12. Go to "Resources" from the top menu drop down button on the BRACU SAM Portal. This page lists available resources like lecture notes, e-books, and tutorials links, categorized by department. Users can search by keyword, filter by category and click "Open link" or "the eye button" to preview the resources information with additional description.



13. Navigate to "Lost and Found" from the drop down of the top menu on the BRACU SAM Portal. This page allows users to report lost items by entering details such as item description, date last seen and location. Users can search by the item name or location and also claim them and mark them as claims.

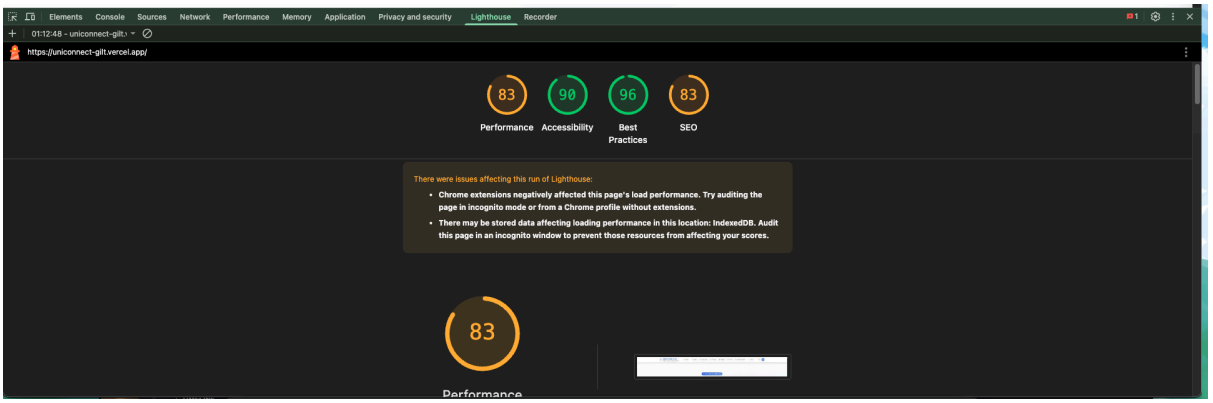


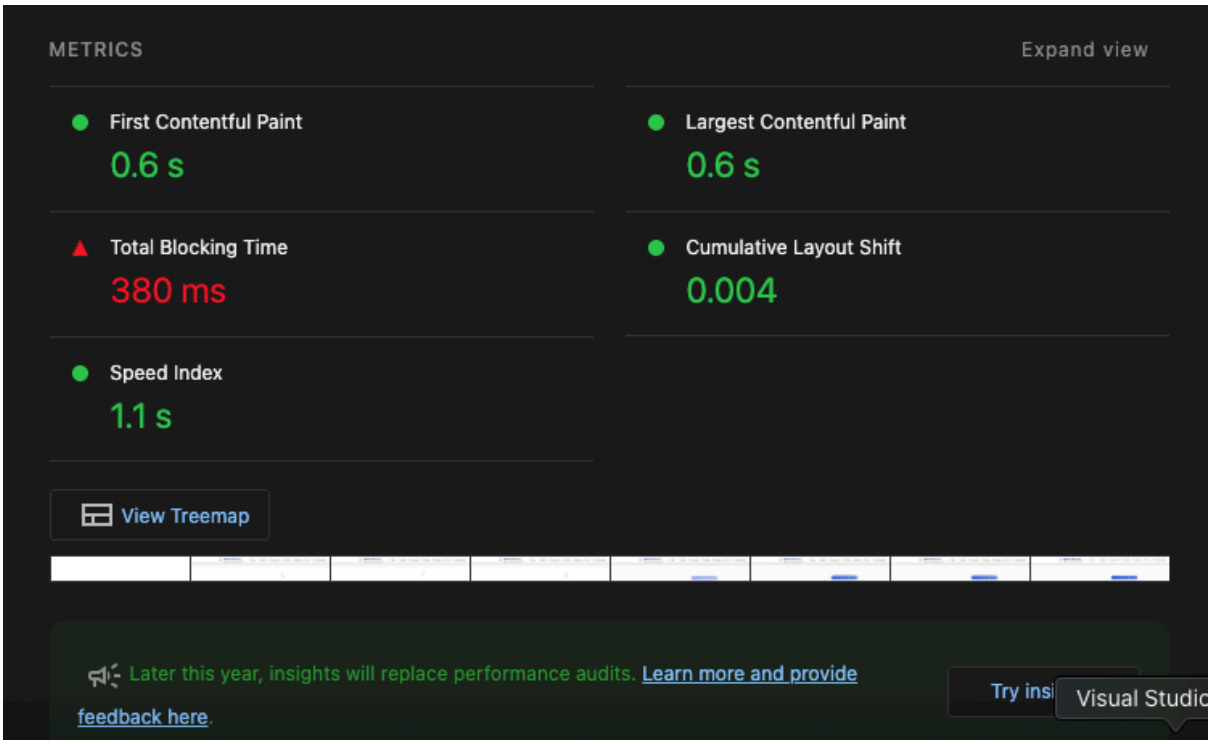
14. Navigate to "AI Assistant" from the drop down of the top menu on the BRACU SAM Portal. This page offers an AI-powered tool available 24/7, including right now, to assist with queries like club details, event schedules, job opening and so on. Users can type questions, receive responses instantly in one place. Here users can decide which club they want to join or in which event they want to register etc.

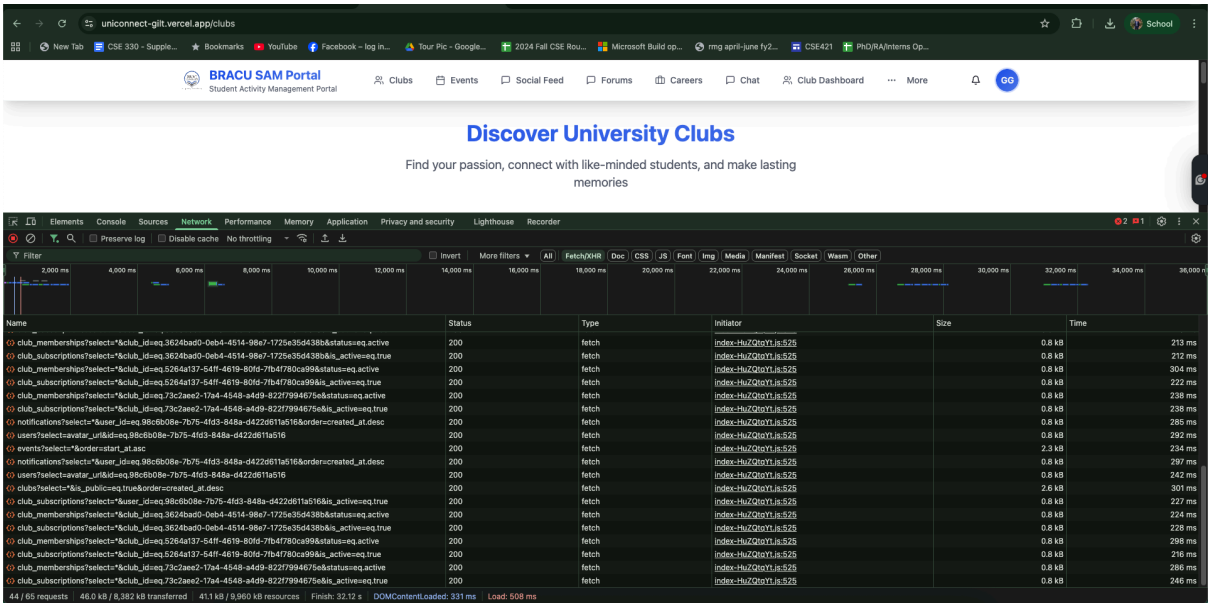


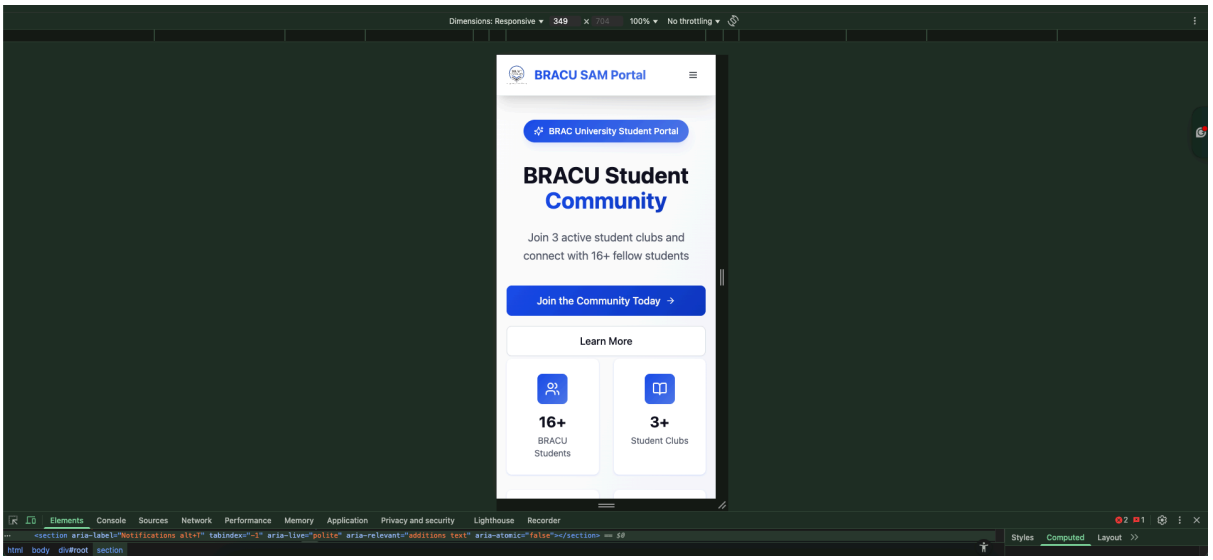
8. Performance and Network Analysis

Performance and Network analysis report using lighthouse DevTool. screenshots of the report. Also a screenshot of my system in the mobile viewport.









9. Github Repo [Public] Link

[Github Repository]

10. Link of Deployed Project

[Uniconnect]

11. Individual Contribution

Group member - 01	
Name: **Md. Jadid Hossain Sanim**	Student ID: **21201590**
Functional Requirements which are developed by this member:	
1. **Profile Management** - Users can view and update comprehensive profiles, including academic info, interests, contact details and profile pictures.	
2. **Club Directory, Discovery & Subscription** - Searchable catalogue of all university clubs with descriptions, member counts, Club panel, department memebers etc. Also, Students can subscribe to specific clubs for targeted notifications.	
3. **Thread-based Discussion Forums** - Thread based discussions organized by topics, clubs, extracurricular activities and academic subjects with comment and thread filtering features.	
4. **Real-time Messaging System** - Direct messaging between users and group chats for clubs, event management groups and other specified groups, like for CP only groups. Users can share images in the chat also.	

Group member - 02	
Name: **MD. Kawsar Habib**	Student ID: **21201327**
Functional Requirements which are developed by this member:	
1.**Multi-role User Registration** - Students, club administrators and university staff can register with role-based access control.	
2.**Lost and Found System -** Allows students and staff to report and track lost or found items with details like description and location.	

3.**Event Creation & Managemen**t - University staff can create, schedule, manage various events, competitions and programs.
4.**Course Resource Repository** - Digital library for non-academic course materials and educational resources.

Group member - 03	
Name: **Md. Rakib Hossain**	Student ID: **21301188**
Functional Requirements which are developed by this member:	
1. **Multi-channel Notification System** - Email and in-website notifications for events, deadlines, club updates and personal messages.	
2. **Club Membership Application & Approval Workflow** - Students can apply to join clubs through custom application forms with file upload capabilities. Also, a multi-stage approval process including application review, viva scheduling and membership confirmation.	
3. **Club Activity Timeline & AI Features**- Dedicated pages showing the club's upcoming announcements, achievements and events. Also an AI assistant to know about all the clubs and events in one place	
4. **Event Registration & Calendar Integration** - Students can enroll in inter-university competitions through form submissions and deadline tracking. Additionally, a centralized calendar displays all university activities, deadlines and important dates with filtering options.	

Group member - 04	
Name: **21201009**	Student ID: **Md. Nafizur Rahman Bhuiya**
Functional Requirements which are developed by this member:	

1. **Central Admin Dashboard** - Central admin dashboard to control everything like adding new clubs, selecting other admins and user activity analysis.
2. **Club Admin Dashboard** - Tools for club administrators to manage member lists, approve applications and organize club activities.
3. **Job Opportunity Update & Career Discussion System** - Dedicated section for job postings, internships, guidelines and career opportunities.
4. **Social Feed, community groups & News Updates** - University-wide news feed with posts, announcements and student activity updates. Also, users can create interest-based groups beyond official clubs for informal networking.

12. References

N/A