VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum -590014, Karnataka.

FULLSTACK WEB DEVELOPMENT PROJECT REPORT

On

Campus Club Finder – 'Club Compass'

Submitted by

ANSHIKA RAMESH	**(1WA24CS046)**
ANUSHKA BS	**(1WA24CS048)**
APOORVA ANAND BURJI	**(1WA24CS050)**
ARYA SINGH	**(1WA24CS055)**

Under the Guidance of
Srushti C S
Assistant Professor

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

**(Autonomous Institution under
VTU) BENGALURU-560019**

2025 - 2026

<u>CERTIFICATE</u>

This is to certify that the project work entitled **Campus Club Finder – 'Club Compass'** carried out by Anshika Ramesh (1WA24CS046), Anushka BS (1WA24CS048), Apoorva Anand Burji (1WA24CS050), Arya Singh (1WA24CS055) who are bonafide students of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the Academic Year Sep 2025- Jan 2026. The project report has been approved as it satisfies the academic requirements in respect of **Full Stack Web Development (23CS3AEFWD)** work prescribed for the said degree.

Signature of the Guide

Srushti C S Assistant
Professor
BMSCE, Bengaluru

Signature of the HOD

Dr. Kavitha Sooda
Prof & Head of Dept of
CSE BMSCE, Bengaluru

External Viva

Name of the Examiner

Signature with date

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECALARATION

We, Anshika Ramesh (1WA24CS046), Anushka BS (1WA24CS048), Apoorva Anand Burji (1WA24CS050), Arya Singh (1WA24CS055) students of 3rd Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this Full Stack Web Development entitled Campus Club Finder – 'Club Compass' has been carried out by us under the guidance of Srushti C S, Assistant Professor, Department of CSE, B.M.S College of Engineering, Bangalore during the academic semester Sept 2025 - Jan 2026.

We also declare that to the best of our knowledge and belief, the development reported here is not from part of any other report by any other students.

Signature

ANSHIKA RAMESH	**(1WA24CS046)**
ANUSHKA BS	**(1WA24CS048)**
APOORVA ANAND BURJI	**(1WA24CS050)**
ARYA SINGH	**(1WA24CS055)**

TABLE OF CONTENTS

1: INTRODUCTION

1.1 Overview

Club Compass is a centralized digital platform designed for B.M.S. College of Engineering that connects students with the diverse range of campus clubs. It serves as a smart compass for student life, enabling students to discover technical, cultural, and departmental clubs. Club Compass helps students choose clubs where they can grow their skills, explore their passions, and stay informed through a unified announcement and event feed- all in one place.

The platform also features an AI-assisted recommendation tool that suggests clubs based on a student's personality and aspirations, along with administrative tools for club coordinators to manage their digital presence efficiently. By centralizing club information into a modern web application, Club Compass aims to improve student engagement and ensure no opportunity is missed due to lack of awareness.

1.2 Motivation

The Problem We Observed

Despite having a vibrant campus culture with numerous active clubs, students especially freshers face significant challenges:

1. **Information Overload & Fragmentation:** Club recruitment drives, event details, and workshop schedules are scattered across disparate WhatsApp groups, Instagram pages, and physical notice boards.
2. **Discovery Issues:** Students are often unaware of niche clubs (like aerospace, robotics, or literature) that match their specific interests until late in their college journey.
3. **Lack of Centralization:** There is no single "directory" where a student can view all active clubs, their current leadership, or their past achievements.
4. **Decision-Making Complexity**: With over 60 active clubs on campus, students often find it overwhelming to identify which clubs best suit their interests, skills, and long-term goals, leading to missed opportunities or uninformed choices.

Inspiration

The inspiration for Club Compass comes from observing how modern digital platforms simplify discovery and engagement. Professional networking platforms like LinkedIn demonstrate the value of structured information, while services such as Spotify highlight the impact of personalized recommendations. We aimed to bring a similar, user-centric experience to the campus environment.

Our exposure to large college fests such as Phase Shift and Utsav further reinforced this idea, as their dedicated websites successfully centralized announcements, event schedules, and updates, making communication seamless and efficient. These examples inspired us to build Club Compass as a unified platform that simplifies club discovery and ensures smooth information flow across the campus.

1: INTRODUCTION

Why This Matters

Extracurricular involvement plays a crucial role in holistic engineering education. Active participation in student clubs enables learners to develop leadership qualities, teamwork, communication skills, and hands-on technical experience that extend beyond conventional classroom learning. Such experiences prepare students to adapt to real-world challenges while enhancing their personal and professional competencies.

When students are unable to identify or connect with suitable peer groups or extracurricular opportunities, they risk missing out on these essential growth experiences. This issue is particularly significant for early-year students, who often face difficulties due to fragmented information sources and limited awareness. Club Compass bridges this gap by ensuring equitable access to information and opportunities for all students, irrespective of their academic year or background. By simplifying discovery and engagement, the platform promotes inclusivity and encourages active participation across the campus.

Our Solution

Club Compass offers a comprehensive, role-based platform designed to serve the needs of all stakeholders within the campus club ecosystem:

i) For Students:
A searchable and well-organized directory of all active clubs, a smart assessment quiz that provides personalized club recommendations, and a centralized dashboard to manage joined clubs, favourites, and announcements efficiently.

ii) For Clubs:
Dedicated profile management tools that allow clubs to maintain consistent digital branding, showcase logos and galleries, and communicate updates effectively to both existing members and prospective students.

iii) For Administrators:
An administrative dashboard that enables monitoring of platform activity, approval of new club registrations, management of user roles, and analysis of engagement trends across the campus.

The Bigger Picture

Beyond simplifying club discovery, Club Compass aims to contribute to long-term improvements in campus engagement:

- Personalize Campus Engagement:
  Move beyond generic and one-size-fits-all announcements by intelligently matching students with clubs that align with their interests, skills, and career goals.

- Level the Playing Field for Clubs:
  Provide equal visibility to both established and emerging clubs, ensuring that student participation is driven by relevance and interest rather than popularity or existing reach.

- Enable Data-Driven Decision Making:
  Empower administrators with insights into student interests and participation patterns, supporting informed decision-making, improved planning, and effective resource allocation.

- Build a Digital Legacy:
  Establish a long-term, evolving repository of club histories, achievements, and activities that future student batches can explore, learn from, and build upon.

By transforming the fragmented and manual process of club discovery into a structured, intelligent, and personalized digital journey, Club Compass seeks to foster a more inclusive, engaged, and connected campus ecosystem where every student can confidently find a community they truly belong to.

2: SOFTWARE REQUIREMENTS

2.1 Hardware Requirements:

The system is lightweight and supports both local development and production deployment.

2.1.1 For Developers (Local Machine):
Processor: Intel Core i5 (8th gen+) or AMD Ryzen 5 equivalent.
RAM: 8 GB minimum (16 GB recommended for running Docker containers smoothly).
Storage: approx. 10 GB (required for container images and database volumes).
Network: Active internet connection for fetching dependencies.

2.1.2 For Production Server:
Instance: e2-micro (GCP Free Tier) or t3.micro (AWS).
Minimum: 1 vCPU, 1 GB RAM, 10 GB SSD.
Recommended (3000+ users): 2 vCPUs, 4–8 GB RAM, 20+ GB SSD, optional load balancer, and separate storage (e.g., S3/Cloudinary).

2.2 Software Requirements:
The project follows a modern Microservices-ready Monolithic architecture.

Operating System: Development: Windows 10/11 (via WSL2), macOS, or Linux (Ubuntu 22.04 LTS).
Server: Linux (Alpine based Docker images).

Backend (API & Data):
Language: Python 3.11+
Framework: FastAPI (High-performance async framework).
Database: PostgreSQL 15 (Relational Data), Redis 7 (Caching & Session management).
ORM: SQLAlchemy (Database interaction).
Authentication: OAuth2 with JWT (JSON Web Tokens).

Frontend (Client):
Framework: Next.js 14+ (React 18) with App Router.
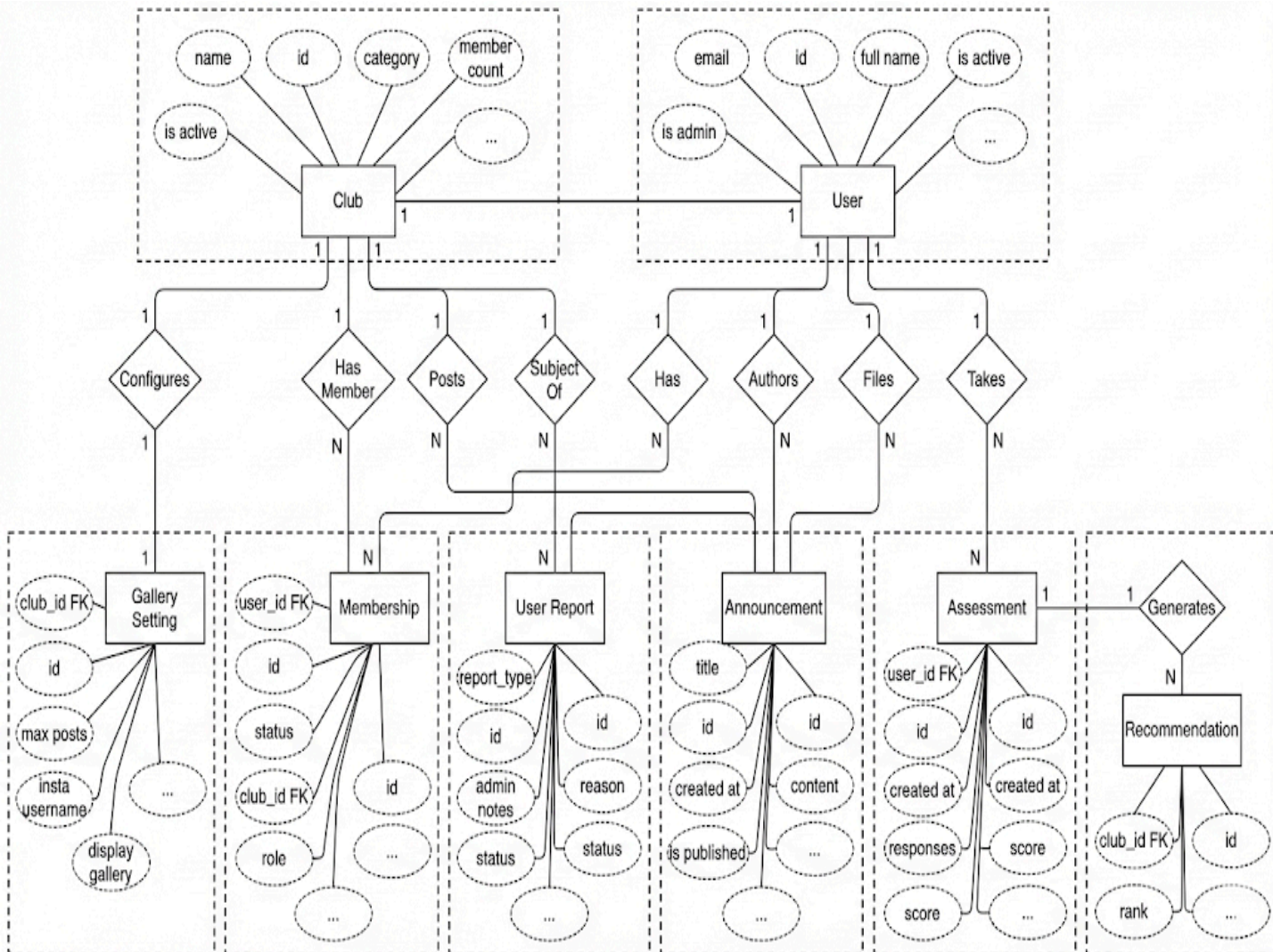Language: TypeScript (Strict type safety).
Styling: Tailwind CSS (Utility-first CSS) with Shadcn/UI components.
State Management: Zustand.
Runtime: Node.js 20+ (Alpine).

Tools & DevOps: Docker, Git & GitHub, Postman

3: ER DIAGRAM OF THE PROJECT

4. SCHEMA OF THE PROJECT

clubs

int	id	PK
string	name	
string	slug	
string	category	
string	subcategory	
string	tagline	
string	description	
string	overview	
string	logo_url	
string	cover_image_url	
string	instagram	
string	linkedin	
string	twitter	
string	website	
string	faculty_name	
string	faculty_email	
string	faculty_phone	
int	member_count	
int	view_count	
datetime	created_at	
datetime	updated_at	
boolean	is_active	
boolean	is_featured	
string	approval_status	
string	rejection_reason	
string	search_vector	

users

int	id	PK
string	email	
string	password_hash	
string	full_name	
datetime	created_at	
datetime	updated_at	
boolean	email_verified	
boolean	is_active	
boolean	is_admin	
string	reset_password_token	
datetime	reset_password_token_expiry	
string	email_verification_token	
datetime	email_verification_token_expiry	
json	preferences	

has config (1:1)

gallery_settings

int	id	PK
int	club_id	FK
string	instagram_username	
boolean	display_gallery	
int	max_posts	
json	cached_posts	
datetime	cache_updated_at	
datetime	created_at	
datetime	updated_at	

posts (1:N)   authors (1:N)

announcements

int	id	PK
int	club_id	FK
int	created_by	FK
string	title	
string	content	
boolean	is_published	
datetime	created_at	
datetime	updated_at	

has (1:N)   has (1:N)

memberships

int	id	PK
int	user_id	FK
int	club_id	FK
string	role	
string	status	
datetime	joined_at	
datetime	updated_at	

is in (1:N)   makes (1:N)

favorites

int	id	PK
int	user_id	FK
int	club_id	FK
datetime	created_at	

takes (1:N)

assessments

int	id	PK
int	user_id	FK
json	responses	
datetime	created_at	

is subject of (1:N)   files (as reporter) (1:N)   is subject of (1:N)   reviews (1:N)

user_reports

int	id	PK
int	reporter_id	FK
int	reported_user_id	FK
int	reported_club_id	FK
string	report_type	
string	reason	
string	description	
string	status	
int	reviewed_by	FK
string	admin_notes	
datetime	reviewed_at	
datetime	created_at	
datetime	updated_at	

is recommended in (1:N)

generates (1:N)

recommendations

int	id	PK
int	assessment_id	FK
int	club_id	FK
int	score	
int	rank	
json	reasoning	
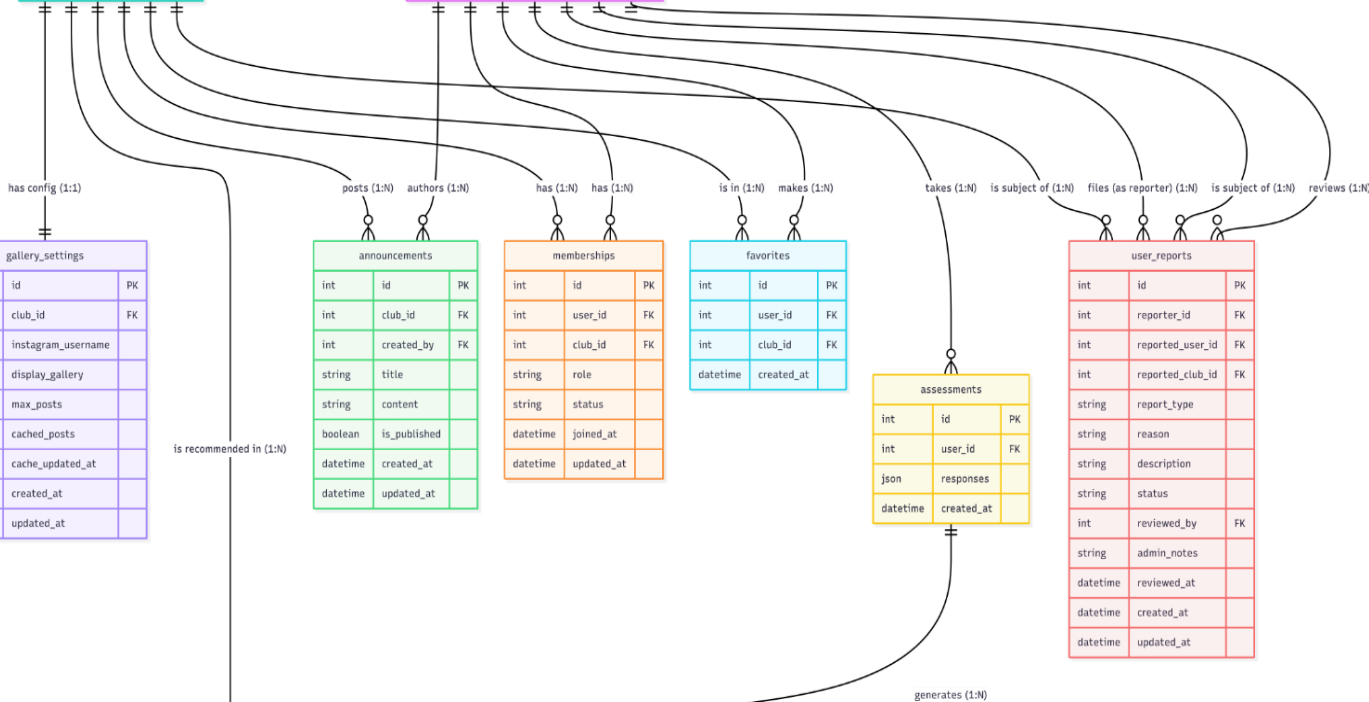
5: USER INTERFACE DESIGN

5.1 Landing Page
The entry point of the application features a hero section with the "Find Your Community" call-to-action. It displays featured clubs (fetched via `get_featured_clubs` API) and a search bar for quick navigation.

5.2 Authentication (Login/Register)
Secure pages utilizing JWT authentication.
  a. Register: Form validating BMSCE email domain formats.
  b. Login: Standard credential entry with error handling.
  c. Forgot Password: Workflow triggering email-based reset tokens.

5.3 Club Discovery Page
A grid-layout catalogue of all active clubs.
  d. Filters: Sidebar filters for Categories (Tech/Non-Tech) and Subcategories.
  e. Search: Real-time text search using PostgreSQL Full-Text Search.
  f. Cards: Each club is displayed as a card showing its logo, category tag, and member count.

5.4 Smart Assessment Interface
An interactive, step-by-step quiz interface.
  g. Progress: Displays a progress bar as users answer questions about their interests (e.g., "Do you prefer coding or public speaking?").
  h. Results: A dynamic results page showing "Top Recommended Clubs" with a percentage match score based on the algorithm.

5.5 Club Detail Page
A dedicated profile page for each club.
  i. Header: Cover image and club logo.
  j. Tabs: Sections for "Overview", "Announcements", and "Gallery" (Instagram integration).
  k. Action Button: A dynamic "Join Club" or "Leave Club" button depending on membership status.

5.6 User Dashboard / Profile
A personalized space for the student.
  l. My Clubs: List of clubs the user has joined.
  m. Favorites: Bookmarked clubs for quick access.
  n. Settings: Options to update profile details or change password.

5.7 Admin Dashboard (/admin endpoint)
Restricted area for platform administrators. (User should have `is_admin=`TRUE`)
  o. Stats: Cards showing Total Users, Active Clubs, and New Registrations.
  p. Moderation: A list of new club requests pending approval (`approval_status=PENDING`).
  q. User Management: Table to view and manage user accounts/roles
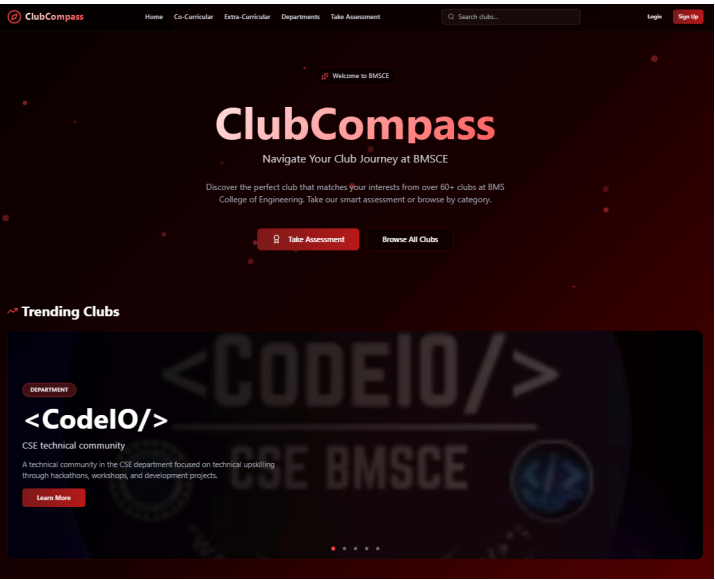
Fig 1. Landing Page

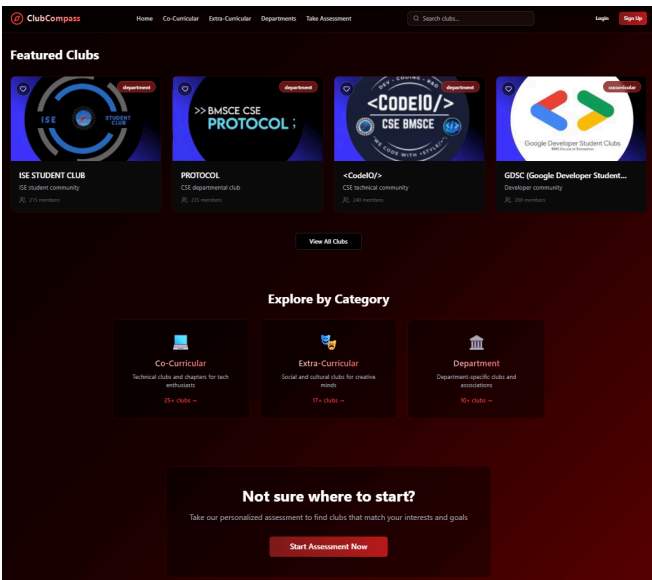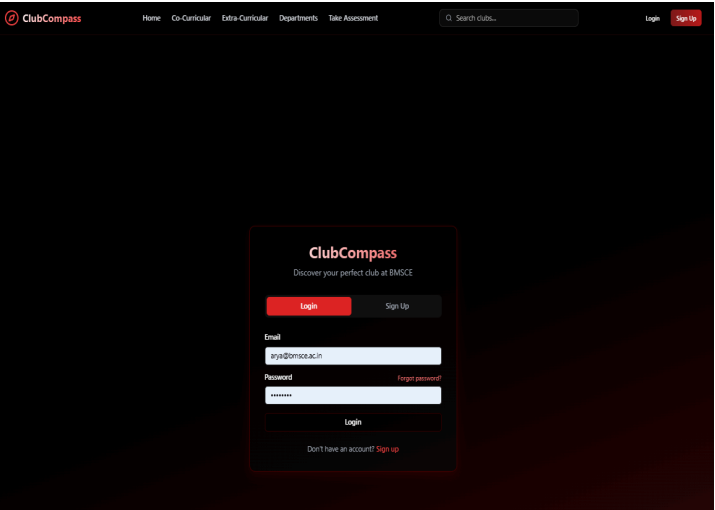Fig 2. Landing Page with Featured Clubs by admin.

Fig 3. Login Page

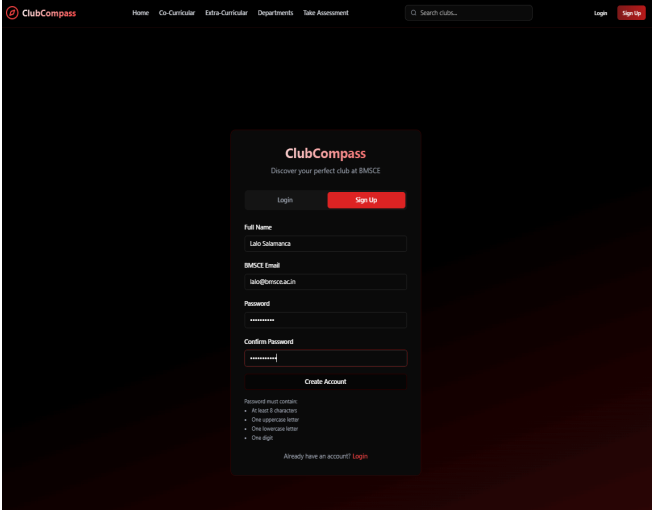Fig 4. Signup Page (form validates BMSCE email)

Fig 5. Category Page - Co-Curricular with diff. filters

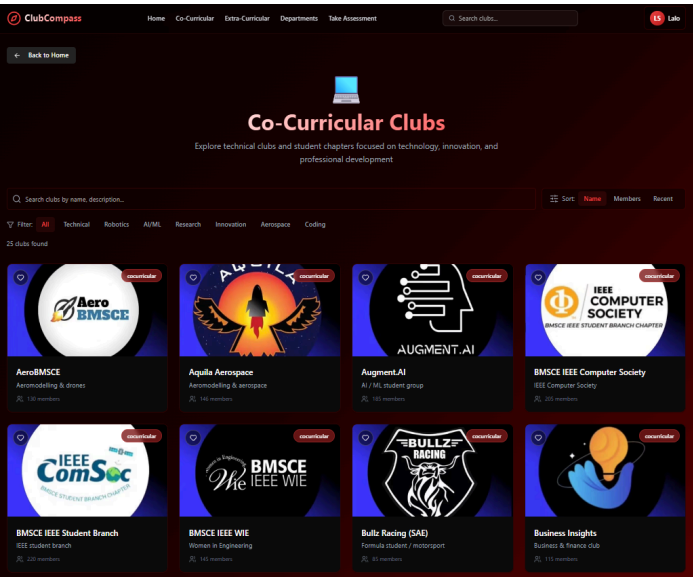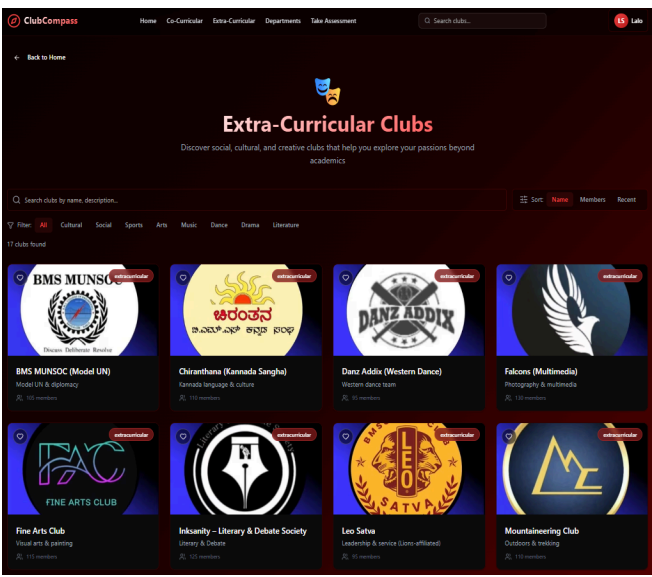Fig 6. Category Page - Extra Curricular with diff. filters
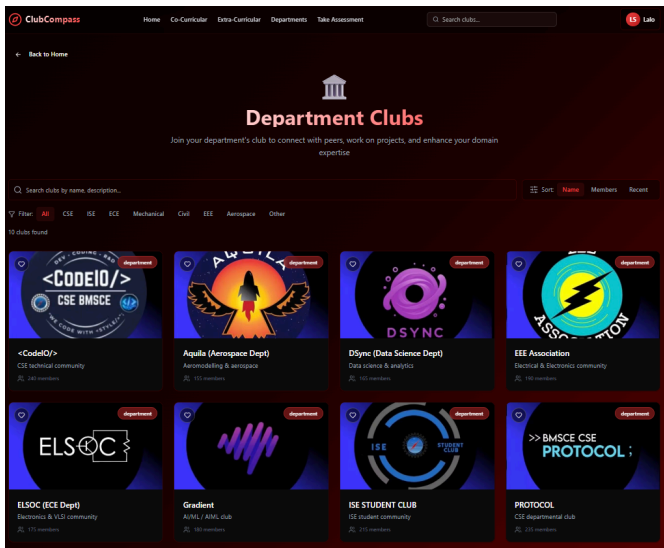
Fig 7. Category Page - Departments with diff. Filters

Fig 8. Assessment Page with questionnaire

Fig 9. Individual detailed club page with announcements and gallery
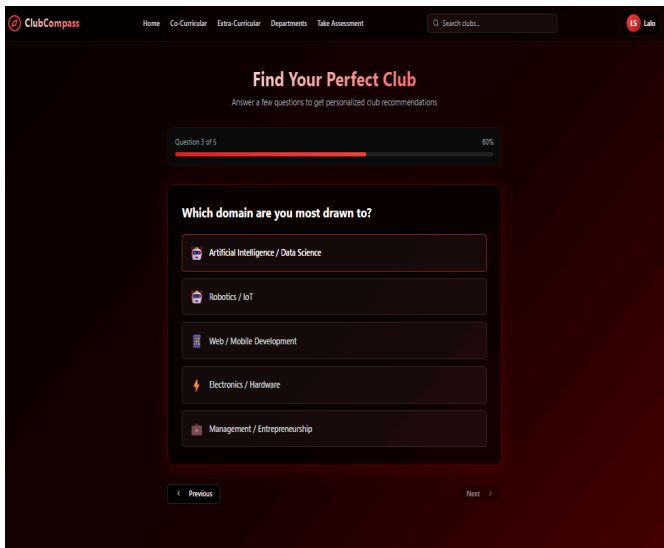
Fig 10. Club 'quick view' modal from category page

Fig 11. user detailed club page with membership/history
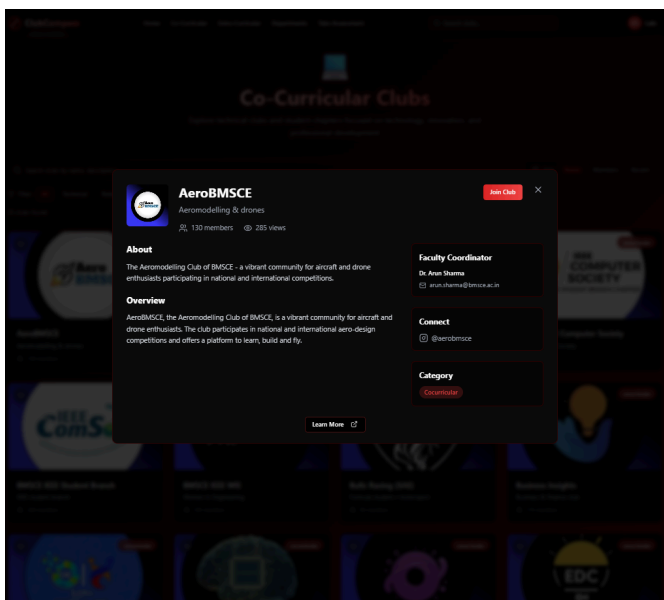

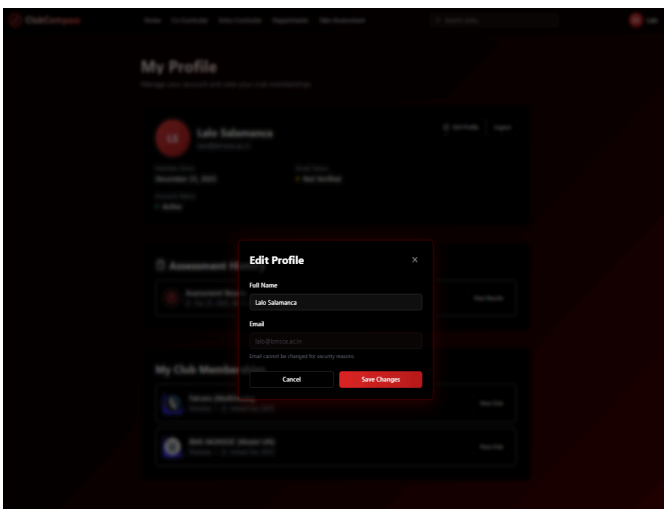Fig 12. User profile edit modal

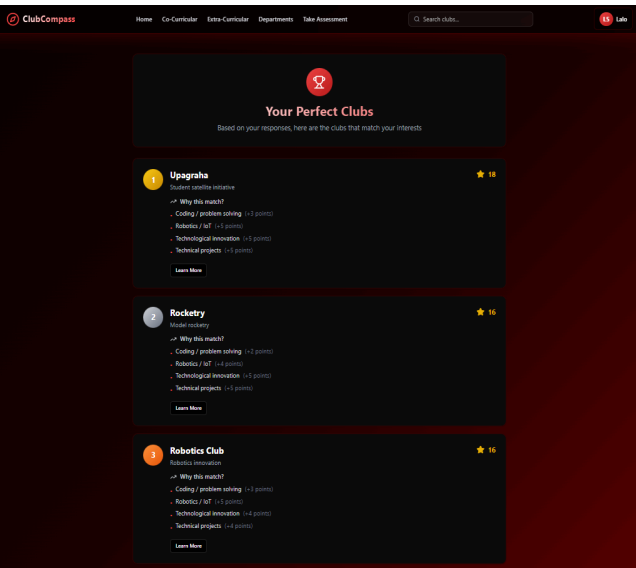Fig 13. Clubs recommended based on relevance score

Fig 14. admin user panel (shows stats & admin user actions)
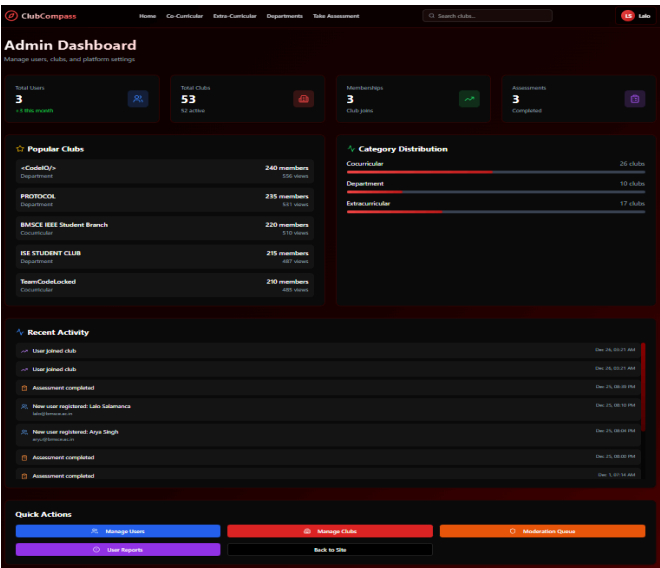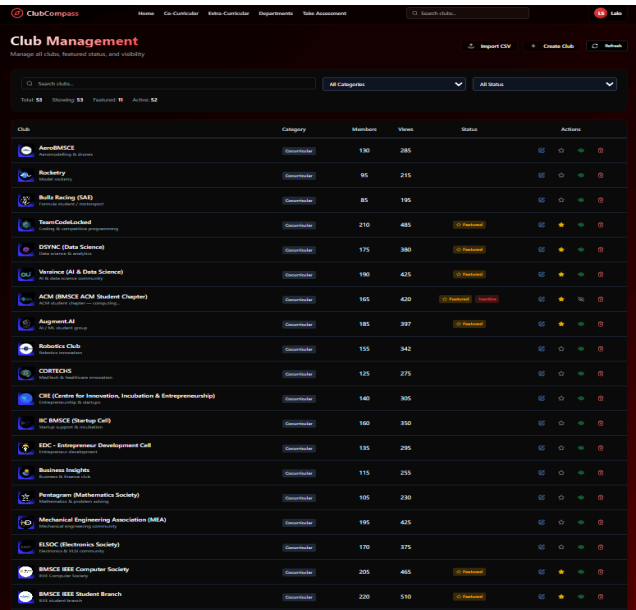

Fig 15. admin - club management

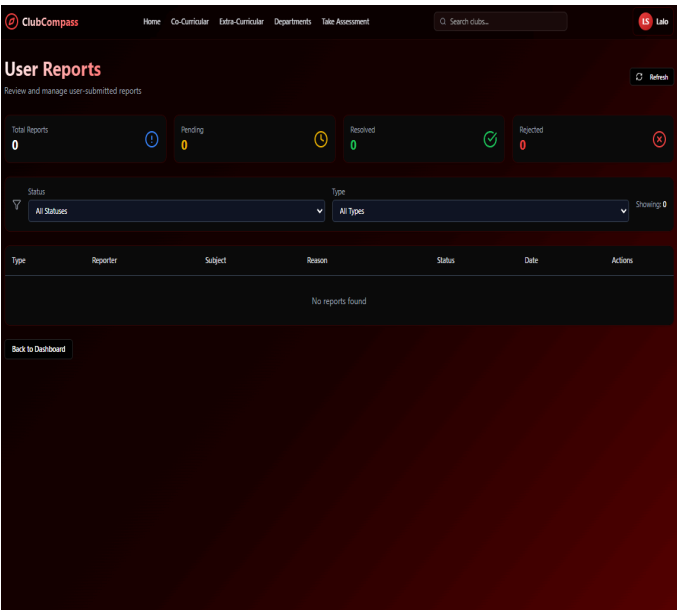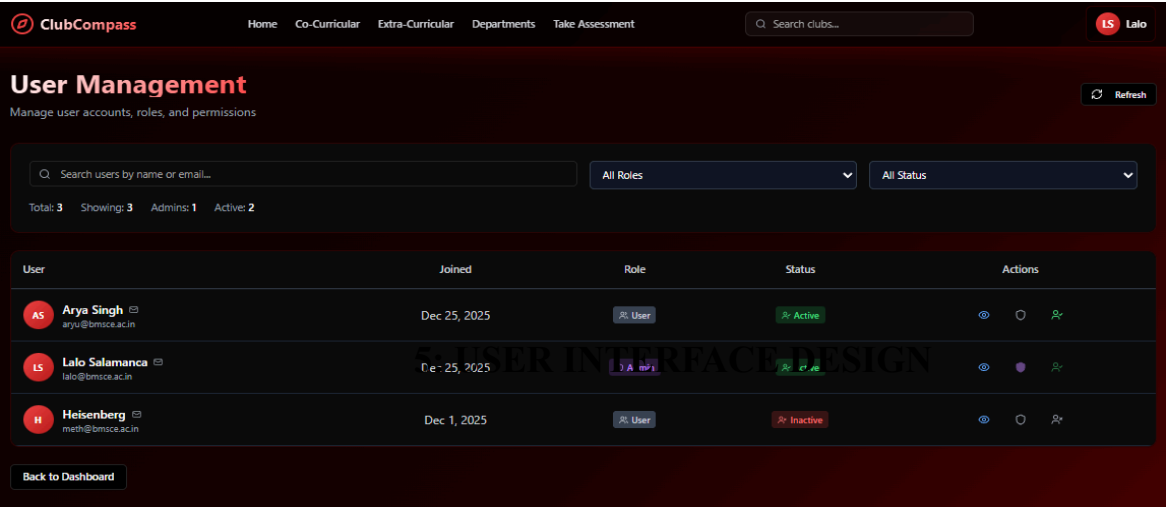Fig 16. admin - club management

Fig 17. admin - user management

6: BACKEND DESIGN & APIs

6.1 Feature Overview

6.1.1 Student Access & Security

What it does: Allows students to securely sign up and log in using their official college email addresses.

Use Case: Ensures only verified students can access the platform, keeping the community safe and exclusive to the college. Includes password recovery and profile management.

6.1.2. Club Discovery & Exploration

What it does: Provides a central directory where students can browse all available clubs, search by name, or filter by categories (like technical, cultural, or sports).

Use Case: Helps students quickly find clubs that match their interests without needing to know the exact club name beforehand.

6.1.3. Smart Club Recommendations

What it does: A personalized "quiz" that asks students about their interests and time availability, then suggests the best-fitting clubs.

Use Case: Solves the "choice overload" problem for new students who don't know where to start, guiding them to clubs they are most likely to enjoy.

6.1.4. Interactive Club Pages

What it does: Each club has a dedicated page displaying its logo, description, latest announcements, and a photo gallery (integrated with Instagram).

Use Case: Gives clubs a digital presence to showcase their activities and attract new members. Students can stay updated on club events and news.

6.1.5. Membership Management

What it does: Enables students to join or leave clubs with a simple click.

Use Case: Streamlines the recruitment process. Students can easily manage their memberships, and clubs can track their member base.

6.1.6 Favorites & Bookmarking

What it does: Lets students save specific clubs to a personal "Favorites" list.

Use Case: Allows students to keep track of clubs they are interested in but might not be ready to join yet, or to quickly access their top clubs.

6.1.7. Community Safety & Reporting

What it does: A confidential channel for users to report inappropriate content, behavior, or clubs to platform administrators.

Use Case: Maintains a safe and respectful environment by allowing rapid flagging of issues for administrative review.

6.1.8. Administrator Dashboard

What it does: A powerful control center for college staff to oversee the entire platform. Includes statistics on usage, tools to manage users and clubs, and content moderation features.

Use Case: Gives the college full control over the platform, allowing them to promote specific clubs, handle reports, and ensure the system is running smoothly.

6.2 API Overview: backend features and their corresponding API endpoints for the ClubCompass platform.

6.2.1 Authentication & User Management
Source: backend/app/api/v1/auth.py, backend/app/api/v1/users.py
Handles user registration, login, token management, and profile updates. It uses JWT for secure authentication and includes BMSCE email validation.

Endpoints

Method	Endpoint	Description	Access
POST	/api/v1/auth/register	Register a new user with BMSCE email.	Public
POST	/api/v1/auth/login	Login with email and password.	Public

Method	Endpoint	Description	Access
POST	/api/v1/auth/refresh	Refresh access token using refresh token.	Public
GET	/api/v1/auth/me	Get current user profile (Auth context).	Authenticated
POST	/api/v1/auth/password-reset/request	Request password reset email.	Public
POST	/api/v1/auth/password-reset/confirm	Reset password using token.	Public
POST	/api/v1/auth/email/send-verification	Send verification email.	Authenticated
POST	/api/v1/auth/email/verify	Verify email using token.	Public
GET	/api/v1/users/me	Get current user profile (User context).	Authenticated
PATCH	/api/v1/users/me	Update user profile.	Authenticated
GET	/api/v1/users/me/memberships	Get current user's club memberships.	Authenticated

Example Usage
Login:

POST /api/v1/auth/login
{
  "email": "student@bmsce.ac.in",
  "password": "StrongPassword123!"
}

6.2.2 Club Management

Source: backend/app/api/v1/clubs.py

Core functionality for listing, searching, and viewing club details. Includes admin features for creating and managing clubs.

Endpoints

Method	Endpoint	Description	Access
GET	/api/v1/clubs/	Get all clubs with optional filtering & search.	Public
GET	/api/v1/clubs/featured	Get featured clubs.	Public
GET	/api/v1/clubs/popular	Get popular clubs by member count.	Public
GET	/api/v1/clubs/{slug}	Get club details by slug.	Public
POST	/api/v1/clubs/	Create a new club.	Admin
PATCH	/api/v1/clubs/{club_id}	Update a club.	Admin
DELETE	/api/v1/clubs/{club_id}	Delete a club.	Admin

Example Usage
Get Club by Slug:

GET /api/v1/clubs/acm-student-chapter

6.2.3 Club Interactive Features (Membership, Announcements, Gallery)

Source: backend/app/api/v1/clubs.py

Features that allow users to interact with clubs, including joining/leaving, viewing announcements, and checking the gallery.

Endpoints

Method	Endpoint	Description	Access
POST	/api/v1/clubs/{club_id}/join	Join a club.	Authenticated
DELETE	/api/v1/clubs/{club_id}/leave	Leave a club.	Authenticated
GET	/api/v1/clubs/{club_id}/announcements	Get announcements for a club.	Public
POST	/api/v1/clubs/{club_id}/announcements	Create a new announcement.	Admin
PATCH	/api/v1/clubs/announcements/{id}	Update an announcement.	Admin
DELETE	/api/v1/clubs/announcements/{id}	Delete an announcement.	Admin
GET	/api/v1/clubs/{club_id}/gallery	Get gallery settings/posts.	Public

Method	Endpoint		
POST	/api/v1/clubs/{club_id}/gallery	Update gallery settings.	Admin
POST	/api/v1/clubs/{club_id}/gallery/refresh	Refresh Instagram gallery cache.	Admin

6.2.4. Assessment & Recommendations

Source: backend/app/api/v1/assessment.py

AI-driven or logic-based assessment system to recommend clubs to users based on their interests and personality.

Endpoints

Method	Endpoint	Description	Access
POST	/api/v1/assessments/	Submit assessment and get recommendations.	Public/Auth
GET	/api/v1/assessments/{assessment_id}	Get assessment results by ID.	Public
GET	/api/v1/assessments/user/{user_id}	Get all assessments for a user.	Authenticated

Example Usage

Submit Assessment:

POST /api/v1/assessments/

```
{
  "responses": {
    "interests": ["tech", "coding"],
    "time_commitment": "medium"
  }
}
```

6.2.5. Favorites

Source: backend/app/api/v1/favorites.py

Allows users to bookmark/favorite clubs for quick access.

Endpoints

Method	Endpoint	Description	Access
GET	/api/v1/favorites/	Get current user's favorited clubs.	Authenticated
POST	/api/v1/favorites/	Add a club to favorites.	Authenticated
DELETE	/api/v1/favorites/{club_id}	Remove a club from favorites.	Authenticated
GET	/api/v1/favorites/check/{club_id}	Check if a club is favorited.	Authenticated

6.2.6. Reporting System

Source: backend/app/api/v1/reports.py

Mechanism for users to report inappropriate behavior, content, or clubs. Includes admin moderation tools.

Endpoints

Method	Endpoint	Description	Access
POST	/api/v1/reports/	Create a new report.	Authenticated
GET	/api/v1/reports/	Get a list of all reports.	Admin
GET	/api/v1/reports/{report_id}	Get report details.	Admin
PATCH	/api/v1/reports/{report_id}	Update report status/notes.	Admin
DELETE	/api/v1/reports/{report_id}	Delete a report.	Admin
GET	/api/v1/reports/stats/summary	Get report statistics.	Admin

6.2.7. Admin Dashboard & Operations

Source: backend/app/api/v1/admin.py

Comprehensive admin suite for platform management, statistics, user/club oversight, and content moderation.

Endpoints

Method	Endpoint	Description	Access
GET	/api/v1/admin/dashboard/stats	Get dashboard statistics.	Admin
GET	/api/v1/admin/users	List all users.	Admin

Method	Endpoint	Description	Role
GET	/api/v1/admin/users/{user_id}	Get user details.	Admin
PATCH	/api/v1/admin/users/{user_id}/role	Update user role (promote/demote).	Admin
PATCH	/api/v1/admin/users/{user_id}/status	Activate/deactivate the user.	Admin
GET	/api/v1/admin/clubs	List all clubs (including inactive).	Admin
PATCH	/api/v1/admin/clubs/{club_id}/featured	Toggle club featured status.	Admin
PATCH	/api/v1/admin/clubs/{club_id}/active	Activate/deactivate club.	Admin
DELETE	/api/v1/admin/clubs/{club_id}	Delete club (Admin override).	Admin
GET	/api/v1/admin/activity	Get recent platform activity.	Admin
POST	/api/v1/admin/clubs/bulk-import	Bulk import clubs from CSV.	Admin

Moderation Endpoints

Method	Endpoint	Description	Access
GET	/api/v1/admin/moderation/pending-clubs	Get clubs pending approval.	Admin
PATCH	/api/v1/admin/moderation/clubs/{id}/approve	Approve a pending club.	Admin
PATCH	/api/v1/admin/moderation/clubs/{id}/reject	Reject a pending club.	Admin
PATCH	/api/v1/admin/moderation/clubs/{id}/request-revision	Request revisions.	Admin
GET	/api/v1/admin/moderation/stats	Get moderation statistics.	Admin

6: REFERENCES

Official Documentation (MUST READ):

Core Concepts for Club Compass:

1. Frontend (Next.js & React)

   A. Components & Props: [Passing Props to a Component – React](#) - Building reusable UI blocks and passing data between them.
   B. State & Lifecycle: [State: A Component's Memory – React](#) - Managing internal component data and side effects (using useState and useEffect).
   C. Handling Events: [Responding to Events – React](#) - Interacting with user inputs like clicks and form submissions.
   D. Conditional Rendering: [Conditional Rendering – React](#) - Displaying different UI elements based on logic (e.g., show login button if logged out).
   E. Lists & Keys: [Rendering Lists – React](#) - Efficiently rendering arrays of data (like club cards) using unique identifiers.
   F. Forms: [Sharing State Between Components – React](#) - Managing user input through controlled components and validation.
   G. Lifting State Up: [Sharing State Between Components – React](#) - Moving state to a common ancestor to share data between sibling components.
   H. Context API: [Passing Data Deeply with Context – React](#) - Managing global state (like User Auth status) accessible throughout the component tree.
   I. App Router & Layouts: [Next.js Docs: App Router](#) - Used for file-system based routing and shared UI (Layouts) across pages.
   J. Server vs. Client Components: [React Foundations | Next.js](#) - Distinguishing between components rendered on the server (for performance) and interactive client-side components.
   K. Data Fetching: [Getting Started: Fetching Data | Next.js](#) - Fetching data asynchronously on the server and passing it to components.
   L. Middleware: [File-system conventions: proxy.js | Next.js](#) - Intercepting requests for authentication checks before rendering pages.

2. Backend (FastAPI & Python)

   A. Path Operations: [First Steps - FastAPI](#) - Defining API endpoints (GET, POST, etc.) to handle specific client requests.
   B. Pydantic Models: [Pydantic](#) - Used for data validation, serialization, and defining strict schemas for request/response bodies.
   C. Dependency Injection: [Dependencies - FastAPI](#) - Managing database sessions (get_db) and authentication tokens (get_current_user) modularly.
   D. ORM (Object Relational Mapping): [SQLAlchemy Documentation](#) - Mapping Python classes (User, Club) to database tables for safe and easy data manipulation.
   E. Async/Await: [asyncio — Asynchronous I/O — Python 3.14.2 documentation](#) - Writing non-blocking code to handle multiple concurrent requests efficiently.

7.3 Infrastructure (Docker)

   A. Containerization: [Get started | Docker Docs](#) - Packaging the application with all its dependencies into standardized units (containers).
   B. Orchestration (Compose): [Docker Compose](#) - Defining and running multi-container applications (Frontend + Backend + Database) with a single command.