Tunisian Republic
Ministry of Higher Education
and Scientific Research
University of Carthage
Higher Institute of Technologies
of Information and Communication

Federe Project Report

IRS2A

By:
Chams Edin Fares
Wajih Ben Zekri
Yassir Hidri
Aziz Nagati

InterClub

The Jury: **Mme.Sara Lasmar**

Academic Year: 2024-2025

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Presentation of the host organization

1.1.1 General Presentation

InterClub is a dynamic, forward-thinking organization dedicated to uniting fans and empowering communities through innovation and decentralization. With a strong commitment to secure and transparent solutions, Inter Club continuously embraces breakthrough technologies that enhance collaboration among its members. Our culture of continuous improvement, trust, and integrity drives every initiative, ensuring that every project not only meets modern technological standards but also strengthens the bond among our global community.

1.1.2 Novative Logo

Figure 1.1: Organization Logo

1.2 General Presentation of the project

1.2.1 Project Context

In an era where digital interactions define our communities, our project aims to revolutionize event and collaboration management. Traditional platforms often suffer from centralization, opacity, and limited automation. In contrast, our platform leverages blockchain [8] technology to ensure security, transparency, and automated governance. By integrating smart contracts [10] with conventional data management, the project creates a digital space where members can organize events, participate in discussions, and collaborate on projects with trust and accountability.

1.2.2 Project Objectives

The primary objectives of this project are to:

- Facilitate the creation and management of events by organizers.
- Provide a secure, blockchain-based [8] mechanism for ticket issuance and attendance verification.
- Enable decentralized user authentication via popular wallets.
- Foster a collaborative environment through integrated discussion spaces and automated promotion systems.
- Enhance trust and transparency in community governance through smart contract-driven [10] voting and promotion processes.

1.3 Feasibility Study

1.3.1 Existing solution

Traditional Platforms

- **Event Management Tools:**
  Platforms such as Eventbrite, Meetup, and Facebook Events dominate the market by offering robust event creation, registration, and promotion functions. These systems are well-established, featuring mature and user-friendly interfaces along with advanced data analytics and marketing tools. However, since they operate on centralized architectures, they suffer from a lack of transparency in participant management and decision-making. Centralized data control also means that there is a risk of data tampering or system failure, and many processes, such as administrative promotions, are handled manually, which can lead to opacity and inefficiency.
- **Collaboration & Discussion Systems:**
  Tools such as Slack, Microsoft Teams, and Discord are widely used for group discussions and project collaboration. They excel at real-time communication and community engagement but remain fully centralized. As a result, they do not provide immutable records of interactions, and their governance mechanisms are limited, making it difficult to ensure transparency and accountability in group decision-making.

Blockchain-Based Alternatives

- **Ticketing & Attendance Verification:**
  A few platforms, such as GUTS Tickets and Blockparty, have adopted blockchain [8] for ticketing and event validation. These solutions use smart contracts [10] to ensure that all transactions and attendance records are immutable, offering a higher degree of transparency. However, these platforms typically focus solely on ticketing and payment verification, and they rarely integrate additional features like group discussions or collaborative workspaces.
- **Decentralized Governance Platforms:**
  Some dApps in the blockchain [8] space integrate voting and community decision-making features. Although these apps provide transparent and automated governance through

smart contracts [10], they tend to address only one aspect of event management—often voting or rewards—rather than providing a comprehensive system for event creation, collaboration, and attendance management.

1.3.2 Our solution

Our platform combines the best features of both centralized and decentralized systems by integrating a dedicated event management system with blockchain-enabled [8] security, transparency, and automated governance.

- **End-to-End Transparency & Security:**
  We leverage blockchain [8] smart contracts [10] to record every critical activity—such as event registrations, attendance tracking via blockchain-linked [8] QR codes, and voting outcomes for promotions and leadership decisions. By immutably logging every significant action on-chain, our platform ensures complete auditability and greatly reduces the risk of fraudulent activity.

- **Automated Governance:**
  Our solution incorporates an automated promotion system that employs community-driven voting via smart contracts [10]. This decentralized approach removes manual intervention from decision-making processes, leading to fair, transparent, and unbiased leadership transitions within the community.

- **Integrated Collaboration:**
  Unlike many competitors that specialize in isolated functionalities, our platform offers a unified experience that seamlessly combines event creation, group discussions, and smart contract-based [10] access management. With secure, passwordless login facilitated by Web3Auth [13], users can easily access a comprehensive digital space designed for meaningful collaboration without compromising security.

- **Community Empowerment:**
  Designed specifically for niche communities that value transparency and trust, our system fosters genuine user engagement by enabling members to actively participate in administrative and organizational decisions. The decentralized decision-making process empowers users to have a real voice in club governance, ensuring that every change—from event management to leadership promotions—is both transparent and verifiable.

1.3.3 Comparison

The following table compares the strengths and weaknesses of traditional platforms, existing blockchain-based [8] solutions, and our integrated solution.

Aspect	**Traditional Platforms**	**Existing Blockchain-Based Solutions** [8]	**Our Solution**
Transparency & Trust	Mature, easy-to-use interfaces. Robust data analytics and marketing capabilities.	Immutable, transparent transaction records using smart contracts [10]. Automated processes for functions like ticket issuance.	Records every key action on the blockchain [8], ensuring full transparency and eliminating central points of failure.
Functionality	Offers mature interfaces and robust analytics.	Provides automated processes for ticket issuance.	Provides a unified suite that combines event management, discussion forums, and decentralized governance in one integrated platform.
Governance	Manual and often opaque decision-making processes, leading to bias and inconsistency.	Typically focused solely on niche functions (e.g., voting) without integrated community management tools.	Utilizes smart contract–based [10], automated voting to manage promotions and leadership transitions, ensuring fair and transparent governance.
User Experience	Well-established, intuitive interfaces.	May lack support for integrated collaboration or comprehensive event management features.	Offers seamless, secure login via decentralized wallet solutions (e.g., Web3Auth [13]) and dedicated discussion spaces for real-time collaboration.
Data Security	Centralized systems are susceptible to data breaches and tampering.	Provides immutable recordkeeping; however, security may be limited to isolated features.	Combines blockchain [8] immutability with decentralized authentication, providing robust, holistic security and a tamper-proof audit trail.

Table 1.1: Comparison of Platforms

Chapter 2

Key Features

2.1 Functional Requirements

The system must support the following functionalities to ensure a seamless and secure event management experience:

1. **Event Creation and Management:**
    - **Comprehensive Event Setup:**
      Organizers can create and update events with detailed information, including title, description, date, and location. with the Support for multimedia attachments (e.g., images, videos, documents) to enhance event details.
    - **User-Friendly Interface:**
      A calendar view visually displays upcoming events and facilitates scheduling.
    - **Blockchain Integration for Integrity:**
      Every event creation and update is logged on the blockchain [8], ensuring an immutable audit trail for transparency and data integrity.
2. **Smart Contract-Based Ticket Issuance:**
    - **Automated Ticket Generation:**
      Smart contracts [10] automatically generate unique digital tickets for each registered participant, guaranteeing authenticity.
    - **Immutable Record Keeping:**
      Ticket issuance details are stored on the blockchain [8], providing a tamper-proof record of all issued tickets.
3. **Ticket Verification via Blockchain [8]:**
    - **QR Code-Based Verification:**
      Attendees receive a QR code linked directly to their blockchain [8] ticket record. On event day, scanning the QR code validates attendance in real time via smart contract [10] interaction.
    - **Real-Time Updates:**
      Attendance is immediately recorded on-chain, ensuring transparent and accurate participant tracking.
4. **Decentralized User Authentication:**

- **Seamless, Passwordless Login:**
  Users authenticate using decentralized wallet solutions (e.g., MetaMask [12], Web3Auth [13]), eliminating the need for traditional passwords.
- **Enhanced Security Integration:**
  The system leverages Web3Auth [13] to provide a frictionless, secure sign-in experience, ensuring user identities are verified through blockchain [8] mechanisms.

5. **Group Discussions and Collaboration:**
   - **Integrated Communication Channels:**
     Each event features a dedicated discussion space for real-time chats, feedback, and collaborative brainstorming.
   - **Moderation and Management:**
     Organizers and administrators have access to moderation tools to manage discussions, enforce community guidelines, and ensure productive collaboration.
6. **Decentralized Governance and Voting:**
   - **Automated Voting System:**
     Members can cast votes for promotions and administrative decisions directly on the blockchain [8]. triggering smart contracts [10] to automatically execute leadership changes, ensuring a fair and transparent process.
   - **Transparent Decision-Making:**
     All voting records are stored immutably, providing an auditable trail of all governance actions.
7. **Club Management:**
   - **Membership Administration:**
     Supports single-club functionality with features for users to join, leave, and manage their membership roles. Administrators can manage member access, monitor participation, and update roles as needed.
   - **Audit Trail for Administrative Changes:**
     Every change in club membership or administrative roles is recorded on the blockchain [8], ensuring complete transparency.
8. **Security & Transparency:**
   - **End-to-End Encryption:**
     Data in transit is protected with robust encryption protocols, and secure JWT-based [11] sessions ensure that user interactions remain private.
   - **Immutable Blockchain Records:**
     Critical transactions and user actions are stored on-chain [8], providing a reliable, tamper-proof history for audit and trust.

2.2 Non-Functional Requirements

- **High Security and Fraud Prevention:**
  The system must utilize blockchain's [8] immutability and encryption to secure data and prevent unauthorized alterations.
- **Scalability:**
  The platform should efficiently handle large-scale events and high volumes of concurrent users without performance degradation.
- **Cost-Efficient Smart Contract Execution:**
  The solution should optimize smart contract [10] operations to minimize gas fees and other blockchain-related [8] costs.
- **User Experience:**
  Despite the underlying complexity, the interface must remain intuitive and user-friendly across both web and mobile platforms.

2.3 System Constraints

- **Blockchain Integration Overhead:**
  While blockchain [8] provides unmatched transparency and security, it introduces latency and cost considerations that must be balanced.
- **Centralized Data Components:**
  Some non-critical data (e.g., event discussions) will be managed in a traditional database, necessitating robust integration between centralized and decentralized systems.
- **Interoperability Limitations:**
  The solution is designed for single-club support, and multi-club functionality is intentionally excluded to maintain system simplicity.
- **Exclusion of NFT Rewards and Online Payments:**
  The focus is solely on event and collaboration management, omitting NFT-based incentives or integrated payment gateways to streamline the platform.

Chapter 3

Project Methodology

3.1 Development Methodology

3.1.1 Agile

Agile is a project management and software development approach that aims to be more effective [1]. The main challenge is to deliver small pieces of work regularly instead of one big launch or update, and make teams able to change quickly and provide customer value faster.

Agile Methodology

Agile methodology [1] is a way to manage projects by breaking them into smaller parts, focusing on working together, and making constant and regular improvements.

Life cycle: Team plan, work on the project, and then review how things are going in a repeating cycle [1].

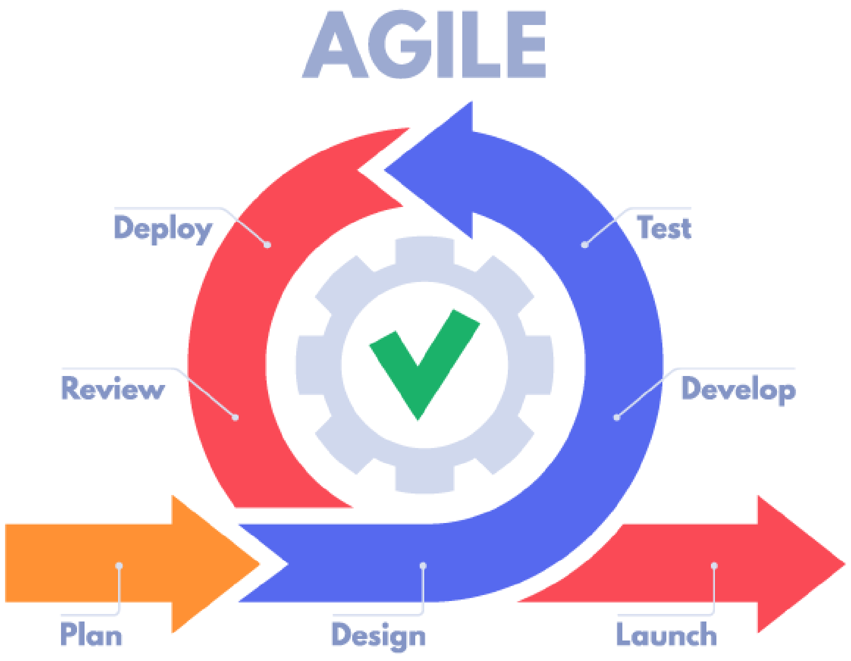


Figure 3.1: Agile Development Stages [1]

The advantages of Agile development methodology [1]:

- Flexibility and Adaptability
- Improved Collaboration
- Faster Delivery
- Enhanced Quality and Customer satisfaction
- Iterative Development
- Transparency
- Quality Assurance
- Continuous Improvement

The disadvantages of the agile model are as follows [1]:

- Less Documentation
- Challenges in Large Organizations
- Need for Senior Programmers
- Limited Scope Control
- Predictability

3.1.2 Scrum

Scrum is a management framework that teams use to self-organize tasks and work towards a common goal, which helps people address complex adaptive problems while maximizing productivity and creativity in delivering products [2].

Scrum lifecycle [2]: The Scrum lifecycle consists of a number of consecutive steps and iterative stages that should be performed during the realization of any Scrum project.

Scrum team Roles [2]:

1. **Product owner:**
   The product owner is responsible for what the team builds, and why they build it. The product owner is responsible for keeping the backlog of work up to date and in priority order.
2. **Scrum master:**
   The Scrum master ensures that the Scrum process is followed by the team. Scrum masters are continually on the lookout for how the team can improve, while also resolving impediments and other blocking issues that arise during the sprint. Scrum masters are part coach, part team member, and part cheerleader.
3. **Development team:**
   The members of the development team actually build the product. The team owns the engineering of the product, and the quality that goes with it.

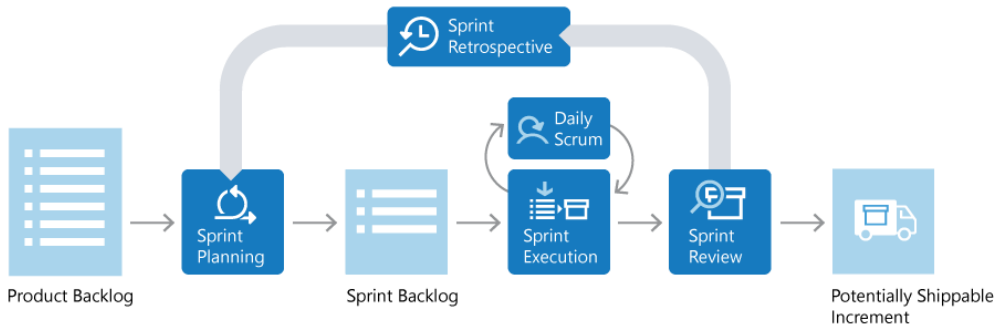


Figure 3.2: Agile Scrum LifeCycle [2]

Scrum Elements [2]:

1. **Product backlog:**
   The product backlog is a prioritized list of work the team can deliver. The product owner is responsible for adding, changing, and re-prioritizing the backlog as needed. The items at the top of the backlog should always be ready for the team to execute on.

2. **Plan the sprint:**
   In sprint planning, the team chooses backlog items to work on in the upcoming sprint. The team chooses backlog items based on priority and what they believe they can complete in the sprint. The sprint backlog is the list of items the team plans to deliver in the sprint. Often, each item on the sprint backlog is broken down into tasks. Once all members agree the sprint backlog is achievable, the sprint starts.

3. **Execute the sprint:**
   Once the sprint starts, the team executes on the sprint backlog. Scrum [2] does not specify how the team should execute. The team decides how to manage its own work.

   Scrum [2] defines a practice called a daily Scrum, often called the daily standup. The daily Scrum is a daily meeting limited to fifteen minutes. Team members often stand during the meeting to ensure it stays brief. Each team member briefly reports their progress since yesterday, the plans for today, and anything impeding their progress.

   To aid the daily Scrum, teams often review two artifacts:

   - Task board:
     The task board lists each backlog item the team is working on, broken down into the tasks required to complete it. Tasks are placed in To do, In progress, and Done columns based on their status. The board provides a visual way to track the progress of each backlog item.
   - Sprint burn-down chart:
     The sprint burn-down is a graph that plots the daily total of remaining work, typically shown in hours. The burn-down chart provides a visual way of showing whether the team is on track to complete all the work by the end of the sprint.

4. **Sprint review and sprint retrospective:**
   At the end of the sprint, the team performs two practices:

   - Sprint review:
     The team demonstrates what they've accomplished to stakeholders. They demo the software and show its value.

- Sprint retrospective:
  The team takes time to reflect on what went well and which areas need improvement. The outcome of the retrospective are actions for the next sprint.

5. **Increment:**
   The product of a sprint is called the increment or potentially shippable increment. Regardless of the term, a sprint's output should be of shippable quality, even if it's part of something bigger and can't ship by itself. It should meet all the quality criteria set by the team and product owner.

6. **Repeat, learn, improve:**
   The entire cycle is repeated for the next sprint. Sprint planning selects the next items on the product backlog and the cycle repeats. While the team executes the sprint, the product owner ensures the items at the top of the backlog are ready to execute in the following sprint.

Advantages of Scrum framework [2]:

- Fast moving and money efficient.
- Dividing the large product into small sub-products. It's like a divide and conquer strategy.
- Customer satisfaction is very important.
- Adaptive in nature because it has short sprints.
- Constant feedback therefore the quality of the product increases in less amount of time.

Disadvantages of Scrum framework [2]:

- Does not allow changes into their sprint.
- Depends on external methods to fully describe your own model like:
  - Extreme Programming (XP)
  - Kanban
  - Dynamic Systems Development Method (DSDM)
- Difficulty in planning, structuring, and organizing a project that lacks a clear definition.
- The daily Scrum meetings and frequent reviews require substantial resources.

3.2 User Stories

1. `US-01: (Admin) Create Event:`
   As an Admin, I want to create events with comprehensive details (title, description, date, location, and multimedia attachments) so that I can effectively manage upcoming events and maintain an immutable record via blockchain [8].

2. `US-02: (Member) Register for Event:`
   As a Member, I want to register for events so that my participation is securely recorded and later validated through blockchain-based [8] mechanisms.

3. `US-03: (Member) Confirm Attendance via QR:`
   As a Member, I want to confirm my attendance by scanning a QR code linked to my digital ticket stored on the blockchain [8], ensuring real-time and tamper-proof attendance verification.

4. `US-04: (Member) Join Group Discussion:`
   As a Member, I want to join group discussions specific to each event so that I can collaborate, share feedback, and engage in real-time communication with other participants.

5. `US-05: (Admin) View Participants:`
   As an Admin, I want to view a complete list of registered participants so that I can monitor attendance and manage follow-up actions efficiently.

6. `US-06: (Founder) Create Club:`
   As a Founder, I want to create a club with a unique identity, enabling members to join, collaborate, and participate in club-specific events and discussions.

7. `US-07: (Member) Join Club:`
   As a Member, I want to join an existing club to access exclusive events and collaborative discussions, contributing to the community's growth.

8. `US-08: (Member) Vote for Promotion:`
   As a Member, I want to cast votes on administrative decisions (e.g., promotions, leadership changes) through a blockchain-based [8] voting system, ensuring a transparent and fair process.

9. `US-09: (Admin) Manage Club Membership:`
   As an Admin, I want to manage club memberships—approving new members and removing inactive ones—to maintain an organized and secure community.

10. `US-10: (Founder) Replace president:`
    As a Founder, I want the system to facilitate an automatic, transparent leadership transition if the majority supports replacing the founder, ensuring seamless and democratic change.

11. `US-11: (Member) Secure Login:`
    As a Member, I want to log in securely using decentralized wallet solutions, so that I can access the platform without traditional password-based vulnerabilities (e.g. using Web3Auth [13]).

12. `US-12: (Member/Admin) Discussion Notifications:`
    As a Member or Admin, I want to receive real-time notifications about new discussion messages and updates so that I remain engaged and informed on collaborative activities.

3.3 Product Backlog

ID	**Actor**	**User Story**	**Priority**
US-01	Admin	Create Event	High
US-02	Member	Register for Event	High
US-03	Member	Confirm Attendance via QR	High
US-04	Member	Join Group Discussion	High
US-05	Admin	View Participants	Medium
US-06	Founder	Create Club	High
US-07	Member	Join Club	Medium
US-08	Member	Vote for Promotion	High
US-09	Admin	Manage Club Membership	Medium
US-10	Founder	Replace Founder	Medium
US-11	Member	Secure Login (Web3Auth [13])	High
US-12	Member/Admin	Discussion Notifications	Low

Table 3.1: Product Backlog

3.4 Sprint Planning

- **Sprint 1:** Core Foundation (2 weeks)
  - Objectives: Set up basic infrastructure, authentication, and simple event creation.
  - Stories:
    - US-11 (Member Secure Login with Web3Auth [13])
    - US-01 (Admin Create Event)
    - US-06 (Founder Create Club)
    - Basic API setup (backend), database schema for events and clubs.
  - Deliverables:
    1. Working login flow (Web3Auth [13]).
    2. Ability to create events and clubs in the database.
    3. Basic UI in both web & Flutter apps to create an event/club.
- **Sprint 2:** Registration & Attendance (2 weeks)
  - Objectives: Implement event registration and on-chain [8] attendance validation.
  - Stories:
    - US-02 (Member Register for Event)
    - US-03 (QR Attendance)
    - US-05 (Admin View Participants)
  - Deliverables:
    1. Blockchain [8] contract for event registration.
    2. QR code scanning for attendance (web + mobile).
    3. Admin screen to see participant lists.
- **Sprint 3:** Collaboration & Voting (2 weeks)
  - Objectives: Add group discussion and voting for promotions.
  - Stories:
    - US-04 (Join Group Discussion)
    - US-08 (Vote for Promotion)
    - US-09 (Manage Club Membership)
    - Possibly US-10 (Replace Founder) if time allows
  - Deliverables:
    1. Discussion board UI + backend routes.
    2. Smart contract [10] functions for voting.
    3. Basic membership management interface.
- **Sprint 4:** Polish & Optional Features (2 weeks)
  - Objectives: Bug fixes, notifications, final touches.
  - Stories:
    - US-10 (Replace Founder) if not completed

    - ∗ US-12 (Discussion Notifications)
    - ∗ Performance improvements, security audits.
    - ∗ Prepare for final deployment.
  - – Deliverables:
    1. Notification system (push or email).
    2. Full end-to-end testing coverage.
    3. Production-ready build and deployment.

Chapter 4

Conceptual Design

4.1 System Architecture

4.1.1 General System Diagram

Our platform is designed as a hybrid, distributed architecture that seamlessly integrates traditional and decentralized technologies to deliver a secure, transparent, and user-friendly solution for event and collaboration management. The design focuses on three core layers:

1. **Frontend Layer:** User Interfaces for Web and Mobile:
   - The platform provides a unified user experience across different devices. The web application (e.g., built using React [3]) and the mobile dApp (developed in Flutter) ensure that users have a consistent interface to access key functionalities such as event creation, group discussions, ticket registration, and voting.
   - Users log in via decentralized authentication methods using tools like Web3Auth [13], ensuring a secure, passwordless experience, interfaces are designed with intuitive navigation and real-time collaboration features, enabling members to participate in discussions, vote on decisions, and view events seamlessly.
2. **Backend Layer:** API and Business Logic:
   - The backend is built on a robust API server using Node.js [5] and Express [6], which centralizes business logic and integration between frontend requests and decentralized operations.
   - Core Responsibilities:
     (a) Data Management: The system interacts with a centralized off-chain database (e.g., MongoDB [14]) to manage non-critical data such as event details, discussion threads, and user profiles.
     (b) Smart Contract Integration: The business logic layer handles interactions with blockchain [8] smart contracts [10] for critical operations like event registration, ticket issuance, attendance validation, and decentralized voting. This layer includes a role validator that checks user permissions (e.g., ensuring only admins can create events) and verifies poll participation before allowing actions like poll creation.
     (c) Security and Encryption: Sensitive requests and responses are secured using JWT-based [11] sessions and encrypted payloads, further enhancing data security and user privacy.

3. **Decentralized Layer:**

   - Blockchain [8] and Smart Contracts [10]:
     The blockchain [8] component ensures that key transactions are immutable, transparent, and auditable. Smart contracts [10] are deployed to:
     - Record event registrations and modifications.
     - Generate unique digital tickets and associated QR codes.
     - Validate attendance in real time through QR code scanning.
     - Manage decentralized governance functions such as voting and automated administrative promotions.

     Immutable Ledger:
     Every significant action is logged on a blockchain [8] ledger, providing an unchangeable audit trail that reinforces trust and accountability in the system.
   - Integration & Communication
     - Data Flow: The design enforces a clear separation of concerns: critical, time-sensitive, and verifiable transactions are handled by the blockchain [8] layer, while rich data storage and fast retrieval are managed by the centralized database.
     - Interoperability: Despite the division between on-chain and off-chain data, our system is designed for smooth interoperability. The backend serves as the integrator that securely processes user requests and coordinates between the decentralized smart contracts [10] and the centralized services.
     - Scalability and Flexibility: By combining decentralized security with traditional high-performance databases, our design ensures that the system can scale to handle large events and high volumes of concurrent users. Future integrations, such as AI-driven analytics or multiclub support, can be incorporated with minimal disruption, making the design truly future-proof.

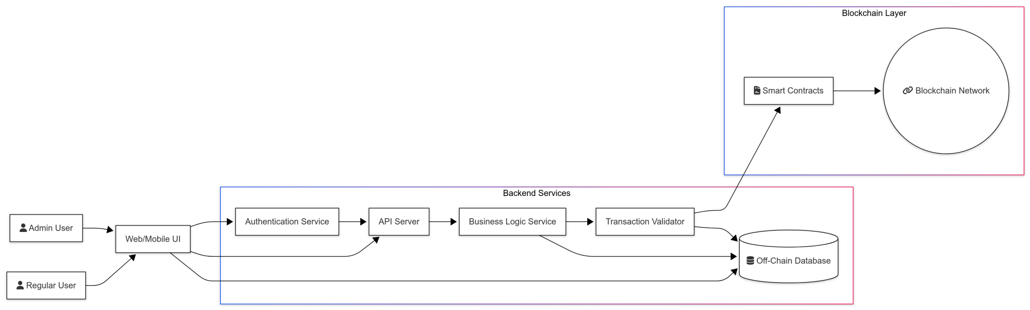


Figure 4.1: System Architecture Diagram depicting the overall structure of the platform, including client applications (Web and Mobile), backend services, blockchain [8] network integration, database layers, and external services (e.g., decentralized authentication with Web3Auth [13]). It highlights the secure, modular, and scalable flow of data and interactions among system components.

4.1.2 Use Case Diagram

The Use Case Diagram provides a high-level view of the key interactions between the platform's actors and its functionalities. It captures the essential workflows of our decentralized event and collaboration system:

- Actors:
  - **Admin:** Represents a user with elevated administrative privileges, responsible for managing the platform and its content/users. It's implied that an Admin can also perform all actions available to a standard User.
  - **Member:** Represents a standard registered user interacting with the platform's core features.
- User Use Cases:
  - **Authenticate (Web3):** Logs into the platform using a Web3 wallet (specifically mentioned as using `Web3Auth [13]`). This is a foundational action included by many other use cases.
  - **View Own Profile:** Allows users to see and potentially manage their profile information. Requires authentication.
  - **View Events:** Enables users to browse and find events hosted on the platform.
  - **Join Event:** Allows users to register for or participate in an event. This process requires authentication and involves QR code verification (`«include» Verify QR Code`).
  - **Post Message:** Enables users to contribute messages, likely in event discussions or forums. Requires authentication.
  - **Vote in Poll:** Allows authenticated users to participate in polls. This action directly interacts with a blockchain [8] component (`«include» Execute Voting Smart Contract [10]`).
- Admin Use Cases:
  - **Create Event:** Allows Admins to set up new events on the platform.
  - **Create Poll:** Enables Admins to initiate new polls for users to vote on.
  - **Manage User Roles:** Allows Admins to assign or modify the roles and permissions of other users.
  - **View System Reports:** Grants Admins access to administrative reports regarding the platform's operation.
  - **Moderate Content:** Allows Admins to review and manage user-generated content (like messages).
- Relationships & Constraints:
  - `«include»` **Relationships:** Indicate dependencies where one use case incorporates the functionality of another:
    - `Authenticate (Web3)` is included by `Join Event`, `Post Message`, `Vote in Poll`, and `View Own Profile`, meaning a user must be logged in to perform these actions.

    * `Vote in Poll` includes `Execute Voting Smart Contract [10]`, showing that voting directly involves a smart contract [10] interaction.
    * `Join Event` includes `Verify QR Code`, suggesting QR codes are used as part of the event joining or check-in process.
  - **Notes:** Annotations providing additional context:
    * The system *"Uses Web3Auth [13] for wallet-based login,"* specifying the authentication technology.
    * The system *"Ensures tamper-proof voting via blockchain [8],"* highlighting the security and integrity benefit of using blockchain [8] for the voting mechanism facilitated by the `Execute Voting Smart Contract [10]` use case.

- Summary:
  The diagram depicts a blockchain-based [8] platform for event management and user interaction. It clearly distinguishes between standard user actions (such as viewing/joining events, posting messages, and voting) and specific administrative tasks (like creating events/polls, managing users/content, and viewing reports). The use case highlights the integration of Web3 technologies for authentication (`Web3Auth [13]`) and blockchain [8] smart contracts [10] for core functionalities like secure, tamper-proof voting.

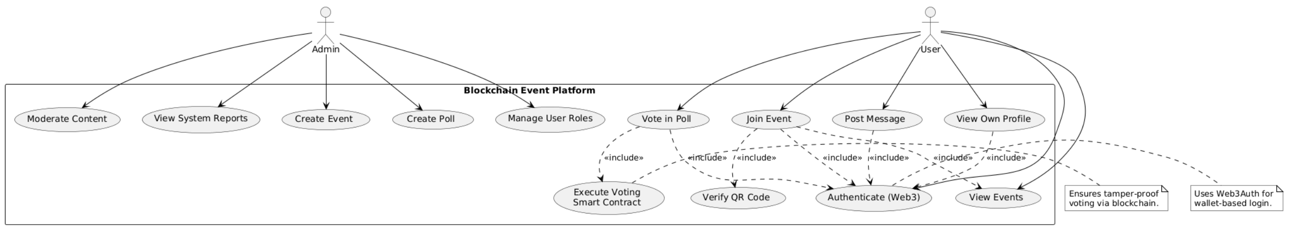


Figure 4.2: Use Case Diagram representing the interactions between different user roles (Admin, Member, Founder) and the platform functionalities, such as event management, secure login (Web3Auth [13]), voting, club management, and blockchain-based [8] verifications.

4.1.3 Class Diagram

The Class Diagram outlines the system's core data structures and their relationships, capturing both the traditional and blockchain-driven [8] components of the platform:

- Core Classes:
  - **User:** Represents any individual using the platform. Common attributes include an identification number, full name, email, and associated wallet for blockchain [8] interactions.
  - **Wallet:** Stores key blockchain [8] authentication details (such as address and private key) and supports operations like transaction signing.
  - **Event:** Encapsulates details of an event (e.g., title, description, date) and includes methods for event creation, updating, and logging changes on the blockchain [8].
  - **Ticket:** Represents the digital ticket issued for an event, linking to a QR code for attendance verification.
  - **Poll:** Contains poll-related information such as a question, list of options, and poll type (election or decision-making). Manages the creation and retrieval of results through smart contracts [10].
  - **Option:** Defines each answer choice within a poll, tracking the number of votes received.
  - **Vote:** Captures information about user votes, linking users to their selected poll options.
  - **Club:** Manages club membership and related administrative functions, including creation and tracking of membership changes on the blockchain [8].
- Enumerations and Inheritance:
  - **PollType and ClubType:** Enumerated types that define the possible categories of polls (e.g., Election or Decision-Making) and club types (e.g., workshops, entertainment, sportive).
  - **Inheritance:** Specific user roles such as Admin and Member inherit from the User class, sharing common attributes while introducing additional role-specific behaviors.

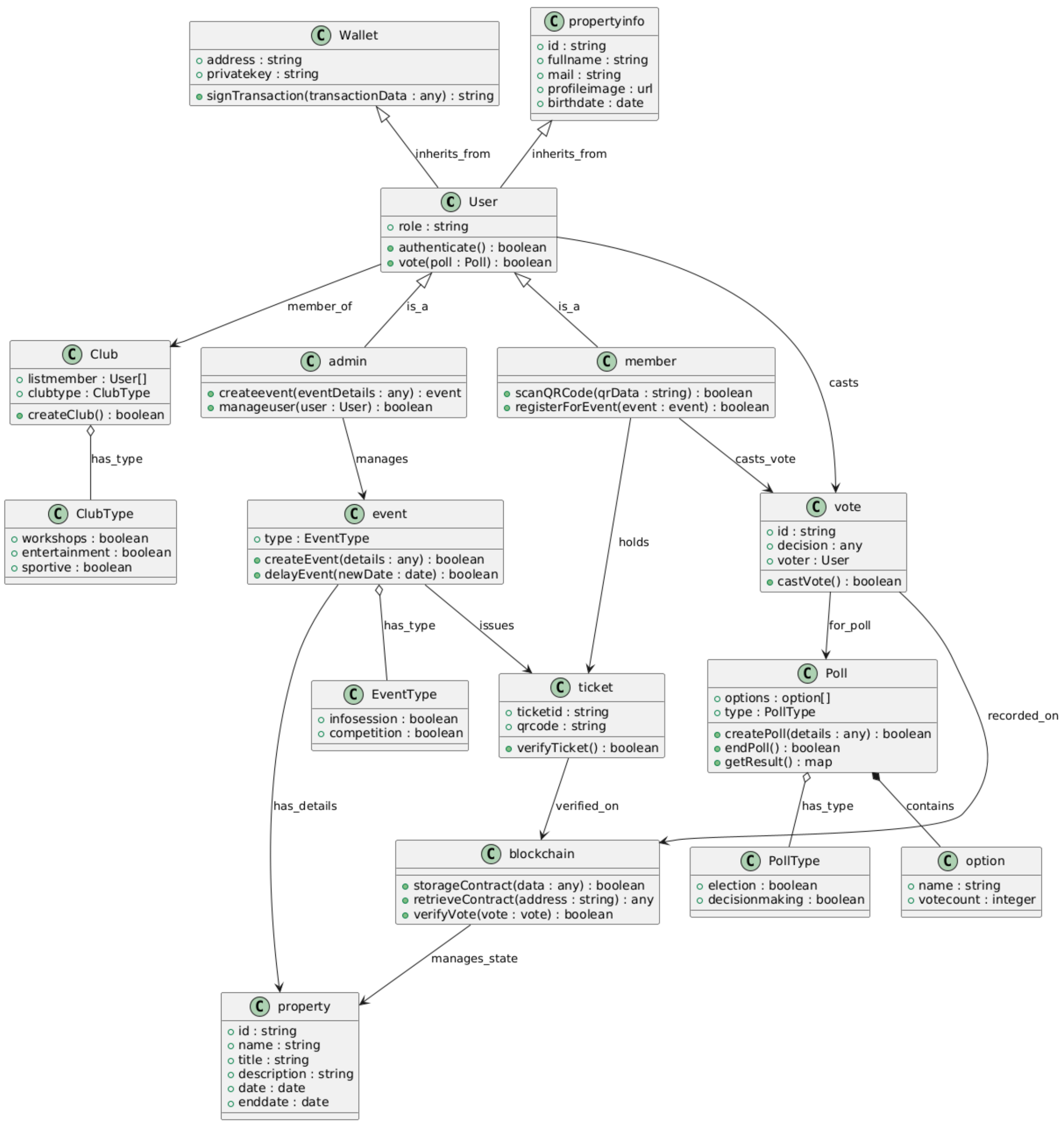


Figure 4.3: Class Diagram illustrating the core entities of the platform, their attributes, methods, and relationships. It outlines how users, events, tickets, polls, and blockchain [8] interactions are structured to ensure modular, scalable, and secure development.

4.1.4 Sequence Diagram

This sequence diagram illustrates the core logic of user interaction with our decentralized event management platform. It captures the step-by-step process that a user follows—from initial authentication to finalizing a vote—while ensuring that key actions are validated and recorded through blockchain [8] smart contracts [10].

- *Detailed Flow Description:*

   1. *User Authentication and Wallet Verification:*
   The flow begins when a user initiates a login request through our web interface using Web3Auth [13]. Upon submission, the system first checks whether the user already exists in the database. If the user is found, the system proceeds to verify the existence and validity of the user's wallet. In the event that the wallet is properly configured, the blockchain [8] component validates the wallet via a smart contract [10], and the user is successfully authenticated and granted access with an issued JWT [11]. If the wallet does not exist, the system prompts the user to create or link a new wallet before proceeding.
   2. *Role-Based Event Creation:*
   Once authenticated, the user may request to create a new event. The system then queries the database to determine whether the user has administrative privileges. If the user is an admin, the system calls the relevant smart contract [10] function to create the event on the blockchain [8]. The blockchain [8] records this event creation immutably, ensuring that the event data, along with all associated metadata, is securely logged. In case the user does not possess admin rights, the system terminates the request and returns a permission denied message.
   3. *Conditional Poll Creation Managed by Admin:*
   In the updated flow, only Admins are authorized to initiate poll creation for events. When an Admin requests to create a poll, the system first verifies whether the event has active participants by checking corresponding registration or attendance records. If the event has confirmed participants, the platform proceeds to create the poll by interacting with the smart contract [10], which immutably logs the poll details (including its type—such as election or decision-making) on the blockchain [8]. If no participants are found, the system denies the poll creation request, ensuring that polls are only created for events with active and meaningful engagement.
   4. *Secure Voting Mechanism:*
   Following poll creation, members can cast their votes on various decisions (e.g., administrative promotions). When a user submits a vote, the system first verifies that the user has not already voted for that particular poll. Once confirmed, the vote is recorded through a smart contract [10] call, and the blockchain [8] logs the vote—ensuring it is immutable and preventing duplicate submissions. The confirmed vote is then communicated back to the user, completing the secure voting process.

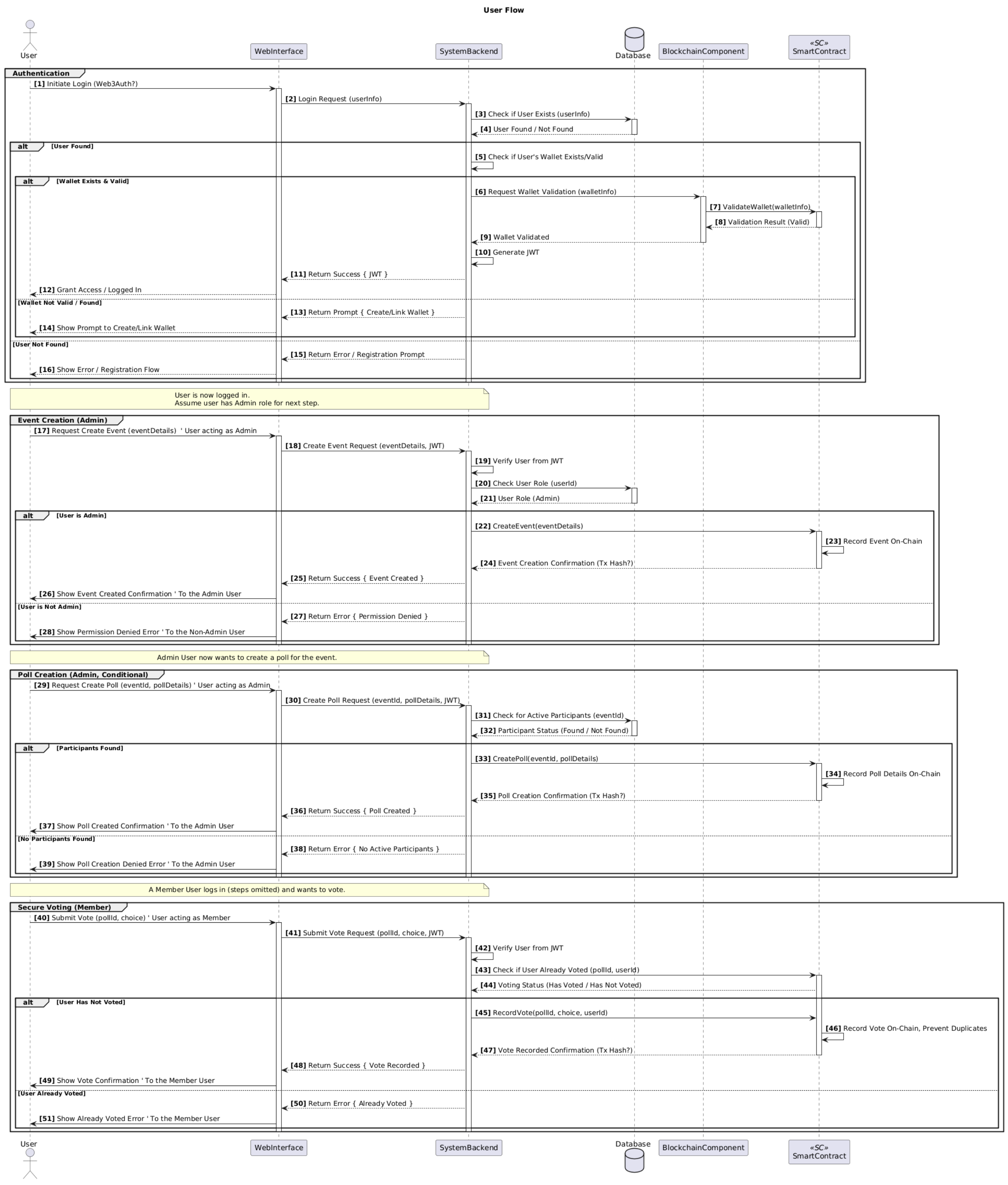

User Flow
User
WebInterface
SystemBackend
Database
BlockchainComponent
«SC» SmartContract
Authentication
[1] Initiate Login (Web3Auth?)
[2] Login Request (userInfo)
[3] Check if User Exists (userInfo)
[4] User Found / Not Found
alt [User Found]
[5] Check if User's Wallet Exists/Valid
alt [Wallet Exists & Valid]
[6] Request Wallet Validation (walletInfo)
[7] ValidateWallet(walletInfo)
[8] Validation Result (Valid)
[9] Wallet Validated
[10] Generate JWT
[11] Return Success { JWT }
[12] Grant Access / Logged In
[Wallet Not Valid / Found]
[13] Return Prompt { Create/Link Wallet }
[14] Show Prompt to Create/Link Wallet
[User Not Found]
[15] Return Error / Registration Prompt
[16] Show Error / Registration Flow
User is now logged in.
Assume user has Admin role for next step.
Event Creation (Admin)
[17] Request Create Event (eventDetails) ' User acting as Admin
[18] Create Event Request (eventDetails, JWT)
[19] Verify User from JWT
[20] Check User Role (userId)
[21] User Role (Admin)
alt [User is Admin]
[22] CreateEvent(eventDetails)
[23] Record Event On-Chain
[24] Event Creation Confirmation (Tx Hash?)
[25] Return Success { Event Created }
[26] Show Event Created Confirmation ' To the Admin User
[User is Not Admin]
[27] Return Error { Permission Denied }
[28] Show Permission Denied Error ' To the Non-Admin User
Admin User now wants to create a poll for the event.
Poll Creation (Admin, Conditional)
[29] Request Create Poll (eventId, pollDetails) ' User acting as Admin
[30] Create Poll Request (eventId, pollDetails, JWT)
[31] Check for Active Participants (eventId)
[32] Participant Status (Found / Not Found)
alt [Participants Found]
[33] CreatePoll(eventId, pollDetails)
[34] Record Poll Details On-Chain
[35] Poll Creation Confirmation (Tx Hash?)
[36] Return Success { Poll Created }
[37] Show Poll Created Confirmation ' To the Admin User
[No Participants Found]
[38] Return Error { No Active Participants }
[39] Show Poll Creation Denied Error ' To the Admin User
A Member User logs in (steps omitted) and wants to vote.
Secure Voting (Member)
[40] Submit Vote (pollId, choice) ' User acting as Member
[41] Submit Vote Request (pollId, choice, JWT)
[42] Verify User from JWT
[43] Check if User Already Voted (pollId, userId)
[44] Voting Status (Has Voted / Has Not Voted)
alt [User Has Not Voted]
[45] RecordVote(pollId, choice, userId)
[46] Record Vote On-Chain, Prevent Duplicates
[47] Vote Recorded Confirmation (Tx Hash?)
[48] Return Success { Vote Recorded }
[49] Show Vote Confirmation ' To the Member User
[User Already Voted]
[50] Return Error { Already Voted }
[51] Show Already Voted Error ' To the Member User
User
WebInterface
SystemBackend
Database
BlockchainComponent
«SC» SmartContract

Figure 4.4: Sequence Diagram visualizing the interaction between Users, the platform backend, blockchain [8] smart contracts [10], and event participation data during the poll creation process. It ensures that only events with confirmed participants are eligible for poll-based decisions.

4.2 Smart Contract Design

4.2.1 What is a Smart Contract

Smart contracts [10] are self-executing programs that enforce the terms of an agreement automatically. They run on blockchain [8] platforms (like Ethereum [9]) and are designed to facilitate, verify, or enforce the negotiation or performance of a contract without the need for intermediaries. In this section, we introduce the concept of smart contracts [10], explaining their ability to securely record transactions and interactions in an immutable, transparent, and decentralized ledger. We also discuss how they function as a trusted and tamper-proof mechanism to execute business logic.

4.2.2 Why Use Smart Contracts

This section outlines the benefits of utilizing smart contracts [10] within our platform. Key reasons include:

- **Enhanced Security and Trust:** By recording all critical actions—such as event registrations, ticket issuance, and voting outcomes—on-chain, smart contracts [10] ensure that transactions are immutable and verifiable.
- **Automated Governance and Efficiency:** Smart contracts [10] eliminate the need for manual processes by automating decision-making (e.g., promotions and leadership transitions) through pre-defined rules, reducing human error and bias.
- **Transparency:** With every transaction stored on the blockchain [8], there is a verifiable audit trail that builds trust among users and stakeholders.
- **Decentralization:** Removing central authorities in key operations not only minimizes vulnerabilities associated with central points of failure but also supports community-driven governance.

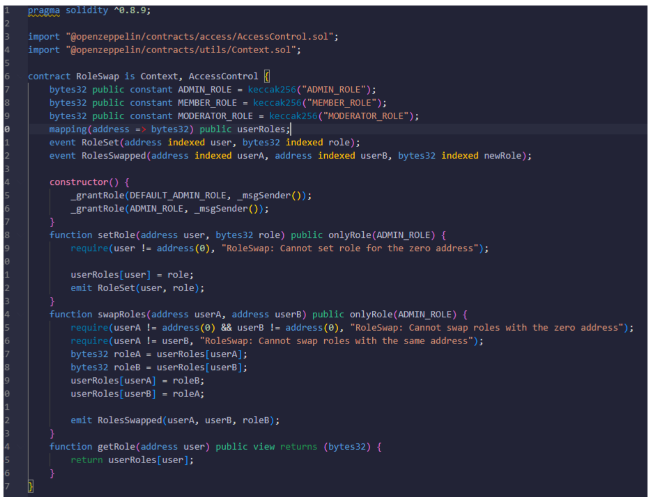
In summary, smart contracts [10] drive our solution's fundamental value proposition—providing a secure, efficient, and transparent operational backbone for the platform.

4.2.3 Smart Contract Code

This subsection presents the actual code for our smart contracts [10]. It includes the implementation details essential for key functions such as:

- **Event Registration:** Recording new event details on the blockchain [8].
- **Ticket Issuance:** Generating unique digital tickets linked to QR codes for attendance validation.
- **Voting Mechanism:** Enabling decentralized votes for promotions and leadership transitions.

We provide sample code snippets to illustrate how these contracts are structured, highlighting important functions, events, and error-handling mechanisms. The code is written following best practices in Solidity [10] (or another chosen blockchain [8] language), ensuring that all interactions are both secure and gas-efficient.



```
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/access/AccessControl.sol";
import "@openzeppelin/contracts/utils/Context.sol";

contract RoleSwap is Context, AccessControl {
    bytes32 public constant ADMIN_ROLE = keccak256("ADMIN_ROLE");
    bytes32 public constant MEMBER_ROLE = keccak256("MEMBER_ROLE");
    bytes32 public constant MODERATOR_ROLE = keccak256("MODERATOR_ROLE");
    mapping(address => bytes32) public userRoles;
    event RoleSet(address indexed user, bytes32 indexed role);
    event RolesSwapped(address indexed userA, address indexed userB, bytes32 indexed newRole);

    constructor() {
        _grantRole(DEFAULT_ADMIN_ROLE, _msgSender());
        _grantRole(ADMIN_ROLE, _msgSender());
    }
    function setRole(address user, bytes32 role) public onlyRole(ADMIN_ROLE) {
        require(user != address(0), "RoleSwap: Cannot set role for the zero address");

        userRoles[user] = role;
        emit RoleSet(user, role);
    }
    function swapRoles(address userA, address userB) public onlyRole(ADMIN_ROLE) {
        require(userA != address(0) && userB != address(0), "RoleSwap: Cannot swap roles with the zero address");
        require(userA != userB, "RoleSwap: Cannot swap roles with the same address");
        bytes32 roleA = userRoles[userA];
        bytes32 roleB = userRoles[userB];
        userRoles[userA] = roleB;
        userRoles[userB] = roleA;

        emit RolesSwapped(userA, userB, roleB);
    }
    function getRole(address user) public view returns (bytes32) {
        return userRoles[user];
    }
}
```

Figure 4.5: Sample Code of Smart Contract (Solidity [10])

Chapter 5

Technology Stack

Component	Technology Used
Frontend	React.js [3], Bootstrap [4], Flutter
Backend	Node.js [5], Express.js [6], WebSocket [7] (Socket.IO)
Blockchain	Ethereum [9]
Smart Contracts	Solidity [10]
Authentication	MetaMask [12], Web3Auth [13]
Database	MongoDB [14] (off-chain data)

Table 5.1: Technology Stack

5.1 Frontend

1. **ReactJs [3]:**
   is a free and open-source front-end JavaScript library [3], that aims to make building user interfaces based on components more `seamless`. It is maintained by Meta and a community of individual developers and companies.
   The main features of React [3] are:

   - Virtual DOM [3]:
     Virtual Document Object Model, React [3] creates an in-memory data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently.

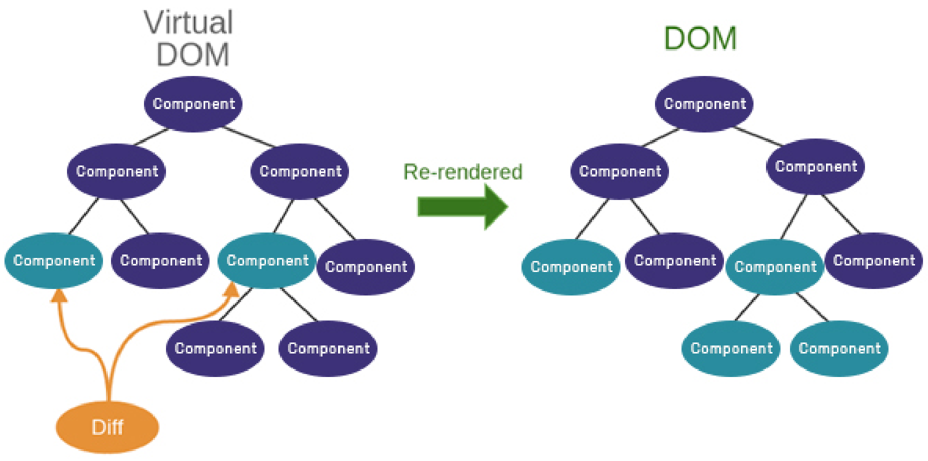


Figure 5.1: Virtual DOM in React [3]

   - JSX [3]:
     JavaScript XML, is an extension to the JavaScript language syntax. Similar in appearance to HTML, JSX [3] provides a way to structure component rendering using syntax familiar to many developers.
   - Components [3]:
     React [3] code is made of entities called components. These components are modular and can be reused. React [3] applications typically consist of many layers of components. The components are rendered to a root element in the DOM using the React [3] DOM library.
   - Server side rendering [3]:
     Server-side rendering (SSR) [3] refers to the process of rendering a client-side JavaScript application on the server, rather than in the browser. This can improve the performance of the application, especially for users on slower connections or devices.

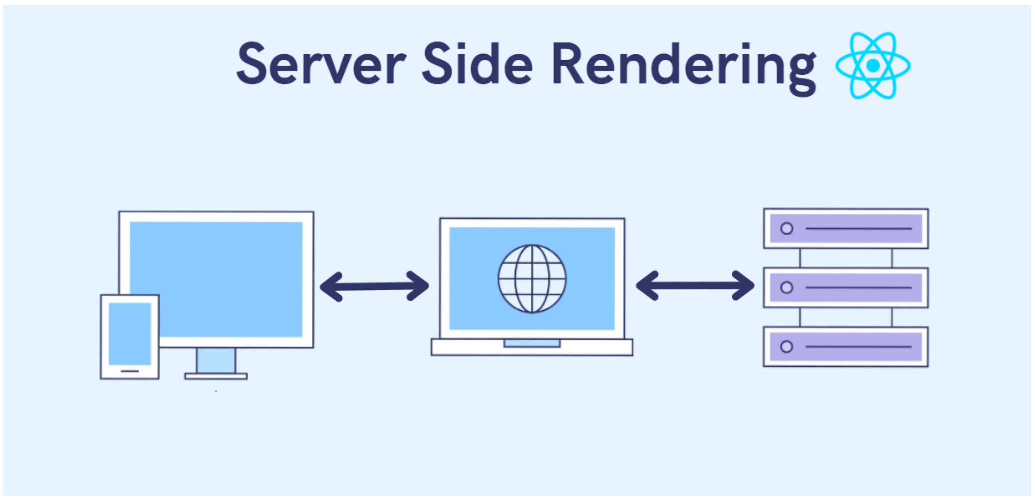


Figure 5.2: Server Side Rendering (SSR) with React [3]

2. **Bootstrap [4]:**
   Bootstrap [4] is an HTML, CSS, and JS library that focuses on simplifying the development of informative web pages (as opposed to web applications). The primary purpose of adding it to a web project is to apply Bootstrap's [4] choices of color, size, font, and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap [4] provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables, and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap [4] to further customize the appearance of their contents.

5.2 Backend

1. **NodeJs [5]:**
   Node.js [5] is a cross-platform, open-source JavaScript runtime environment that can run on Windows, Linux, Unix, macOS, and more. Node.js [5] runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser, lets developers use JavaScript to write command line tools and for server-side scripting. The ability to run JavaScript code on the server is often used to generate dynamic web page content before the page is sent to the user's web browser.

2. **ExpressJs [6]:**
   is a back end web application framework for building **RESTful APIs** with Node.js [5], released as free and open-source software under the MIT License. It is designed for building web applications and APIs [6].

3. **WebSocket [7]:**
   WebSocket [7] is a standardized communication protocol that enables simultaneous two-way communication over a single TCP connection. It has full-duplex or bi-directional capabilities that distinguishes it from HTTP.

Figure 5.3: Node.js [5] and ExpressJs [6] logo

5.3 Blockchain

1. **BlockChain [8]:**
   blockchain [8] is a distributed ledger with growing lists of records (blocks) that are securely linked together via cryptographic hashes. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data.

2. **Ethereum [9]:**
   Ethereum [9] is a decentralized blockchain [8] with smart contract [10] functionality. Ether is the native cryptocurrency of the platform. Among cryptocurrencies, ether is second only to bitcoin in market capitalization. It is open-source software [9].

5.4 Smart Contracts

1. **Solidity [10]:**
   Solidity [10] is a programming language for implementing smart contracts [10] on various blockchain [8] platforms, most notably, Ethereum [9]. Solidity [10] is licensed under GNU General Public License v3.0.

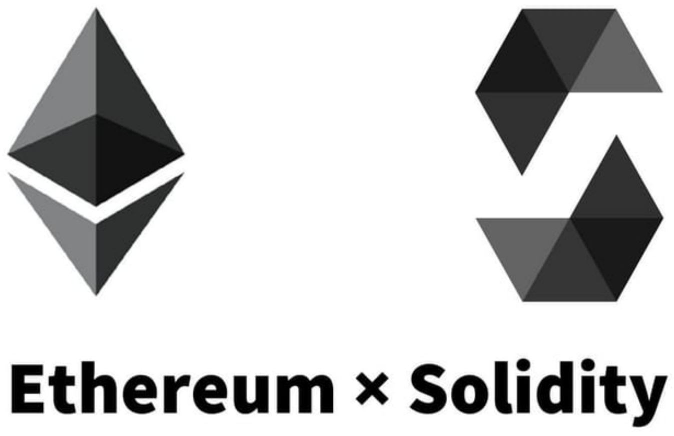


Figure 5.4: Ethereum [9] and Solidity [10] logo

5.5 Authentication

1. **JWT [11]:**
   JSON Web Token (JWT) [11] is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key.

2. **MetaMask [12]:**
   MetaMask [12] is a software cryptocurrency wallet used to interact with the Ethereum [9] blockchain [8], allows users to store and manage account keys, broadcast transactions, send and receive Ethereum-based [9] cryptocurrencies and tokens, and securely connect to decentralized applications through a compatible web browser or the mobile app's built-in browser.

3. **Web3Auth [13]:**
   Web3Auth [13] is a pluggable, embedded wallet technology that facilitates Web3 wallet integration and user onboarding. It supports OAuth-based logins on many platforms, allowing users to access Web3 applications using familiar authentication techniques in less than a minute.

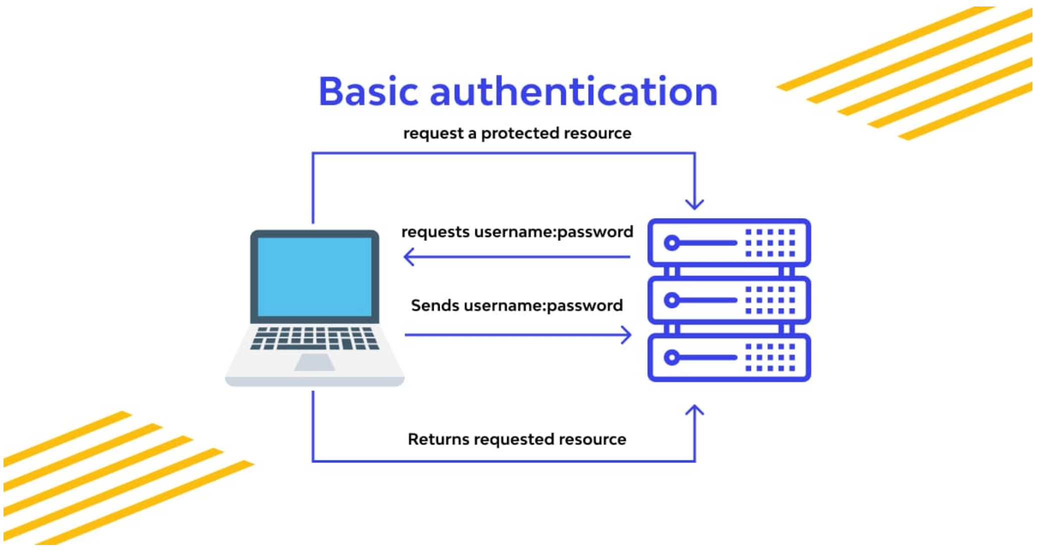

Basic authentication
request a protected resource
requests username:password
Sends username:password
Returns requested resource

Figure 5.5: Authentication technologies including MetaMask [12] and Web3Auth [13]

5.6 Database

1. **MongoDB [14]:**
   MongoDB [14] is a source-available, cross-platform, document-oriented database program. Classified as a NoSQL database product, MongoDB [14] uses JSON-like documents with optional schemas.

mongoDB

Figure 5.6: MongoDB [14] logo

5.7 User Interface (UI/UX) Design

5.7.1 Design Philosophy

The design philosophy behind this interface is grounded in the principles of simplicity, accessibility, and user-centeredness. The primary goal is to create an intuitive and collaborative platform that facilitates communication and interaction among members of a university.

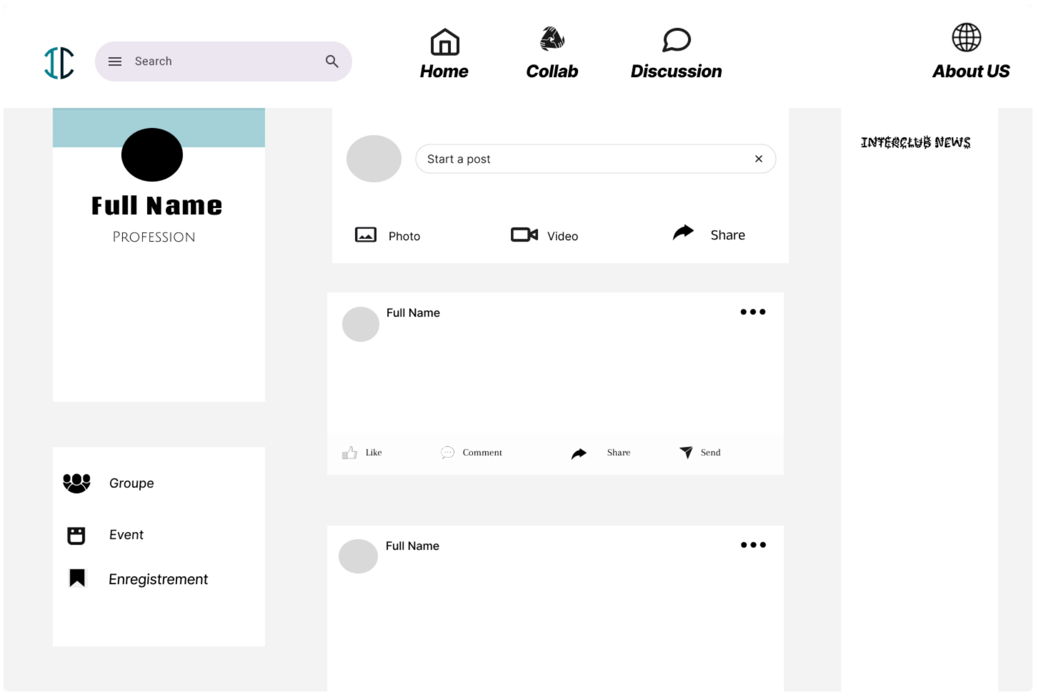


Figure 5.7: Home page

- **Simplicity and Clarity:** The interface adopts a clean and minimal layout, using neutral colors, clear icons, and a well-structured visual hierarchy. This approach reduces cognitive load and ensures a smooth user experience, even for first-time users.
- **Intuitive Navigation:** The top navigation bar features clearly labeled and easily recognizable icons (Home, Collab, Discussion, About Us), helping users move through the platform effortlessly.
- **User-Centered Approach:** Core elements like the news feed, quick post module ("Start a post"), and interaction buttons (like, comment, share, send) are inspired by popular social platforms, which enhances usability and encourages engagement. The personal profile section (profile photo, profession) reinforces user identity and belonging.
- **Community Engagement:** Sections such as "Group", "Event", "Saved", and "Interclub News" are designed to promote participation, highlight important events, and foster a strong sense of community within the platform.

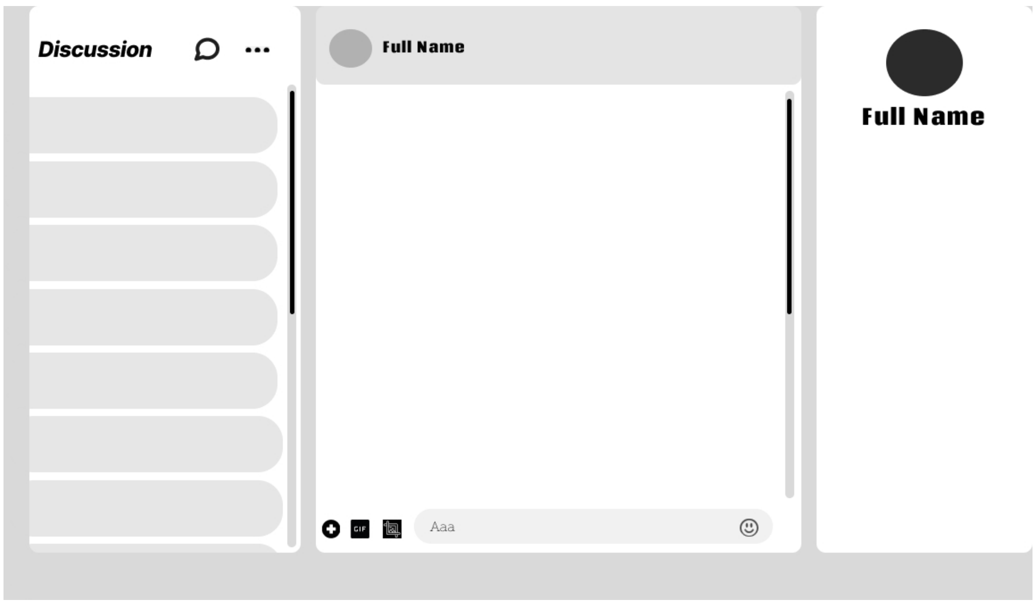


Figure 5.8: Discussion page

- **Communication-Centric Layout:** The design is structured to prioritize messaging :
- The left panel displays the list of ongoing conversations for quick access.
- The central area is dedicated to the active chat, emphasizing message readability and interaction.
- The right panel shows user information, maintaining context and identity throughout the conversation.
- **Minimalist Aesthetic:**A clean, monochrome color scheme and simple icons keep the interface distraction-free. This promotes focus on the conversation and reduces visual noise, especially in long or active threads.
- **Familiar User Experience:** By adopting familiar interaction patterns found in popular messaging platforms (like Messenger, Discord, or Slack), the design ensures intuitive use and fast user onboarding.
- **User Presence and Identity:**The dedicated profile section reinforces a sense of user identity and presence, which enhances the interpersonal aspect of conversations and supports collaborative use cases.

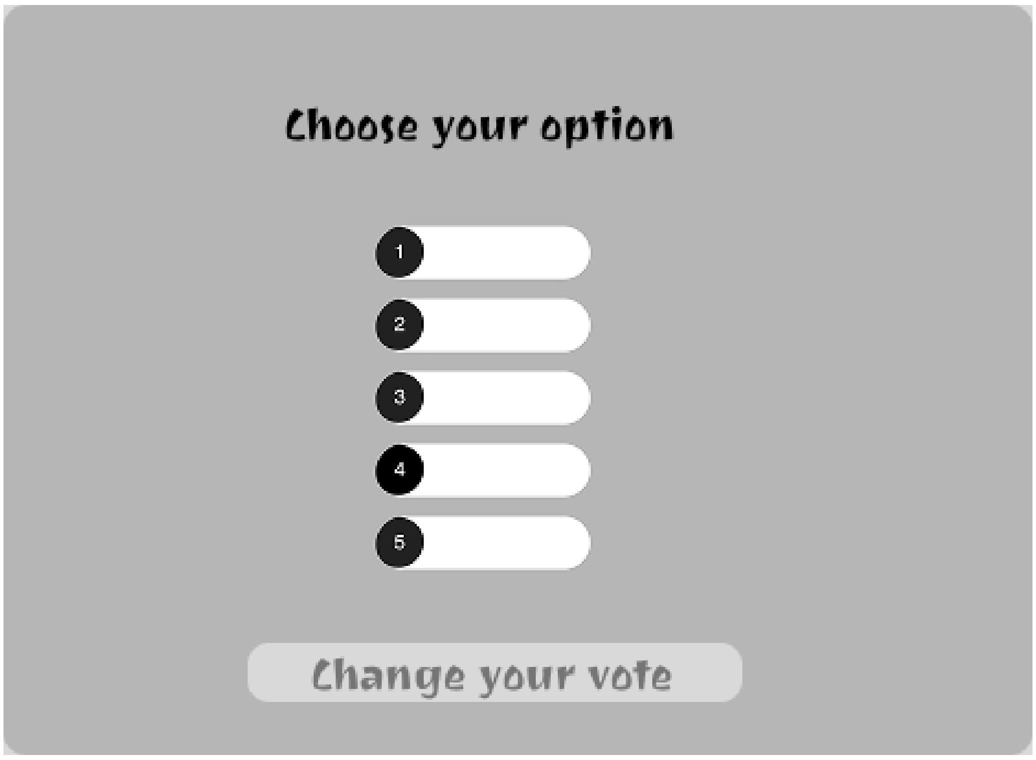


Figure 5.9: Vote card in the collaboration page

- **Simplicity in Interaction**The interface is minimalistic, with a clear vertical arrangement of choices (1 to 5), allowing users to quickly make a selection without distractions. This linear layout ensures that the voting process remains intuitive and efficient.
- **Clarity of Choices**Each option is presented with equal visual weight, using a consistent design language (rounded shapes, numbered labels). The current selection is highlighted, reinforcing user feedback and avoiding confusion.
- **Focus on User Control**The presence of a "Change your vote" button enhances user autonomy, allowing participants to revise their decision. This reinforces trust and flexibility in collaborative participation.
- **Neutral Aesthetic**The use of a grayscale color palette minimizes visual noise and maintains focus on the functionality, which is particularly effective in decision-making interfaces where clarity is essential.

5.7.2 Mobile App UI

1. **Home Page:** displaying posts from users with standard interaction options like liking and commenting. It incorporates a search function at the top and allows users to follow others directly from posts

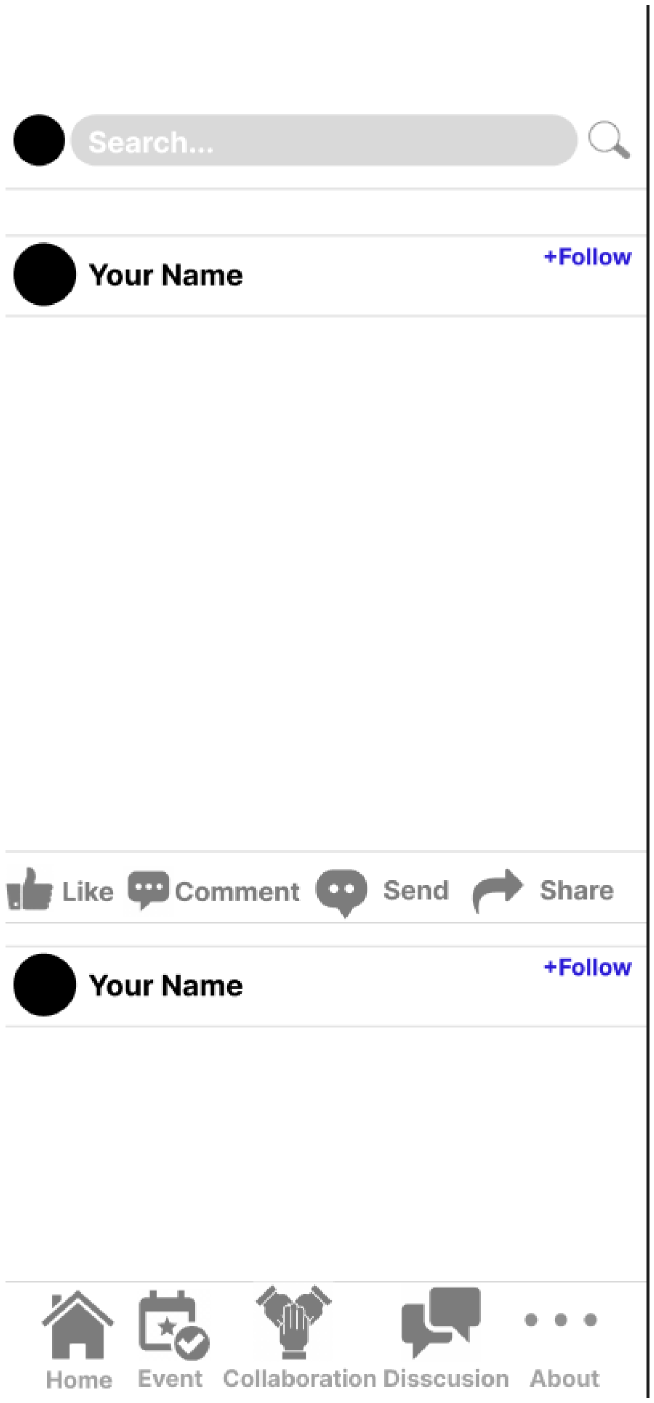


Figure 5.10: Home Page

2. **Profile Page:** This page showcases an individual user's profile information, including their name and social stats like following/follower counts. It allows other users to follow them and provides access to their posts and activity/registrations.

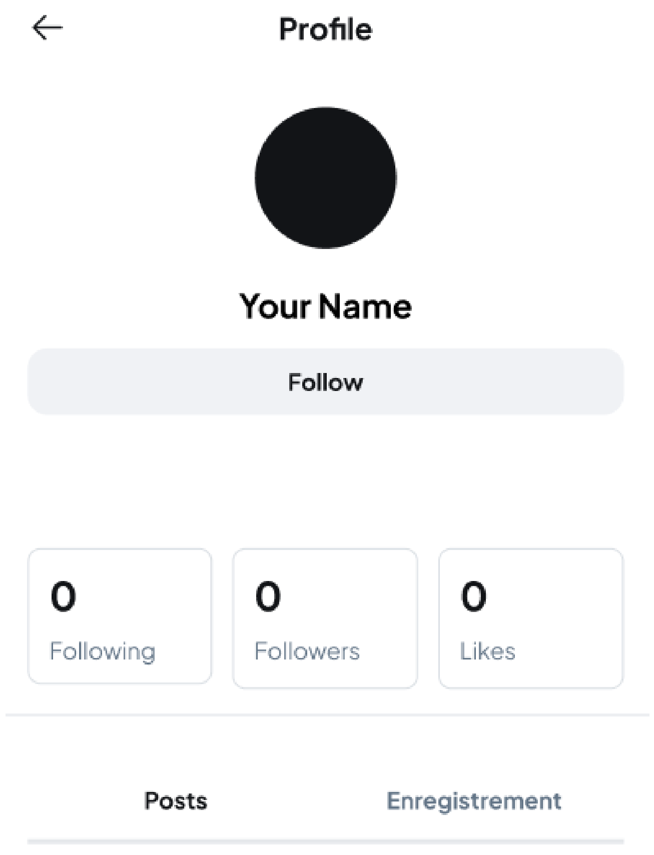
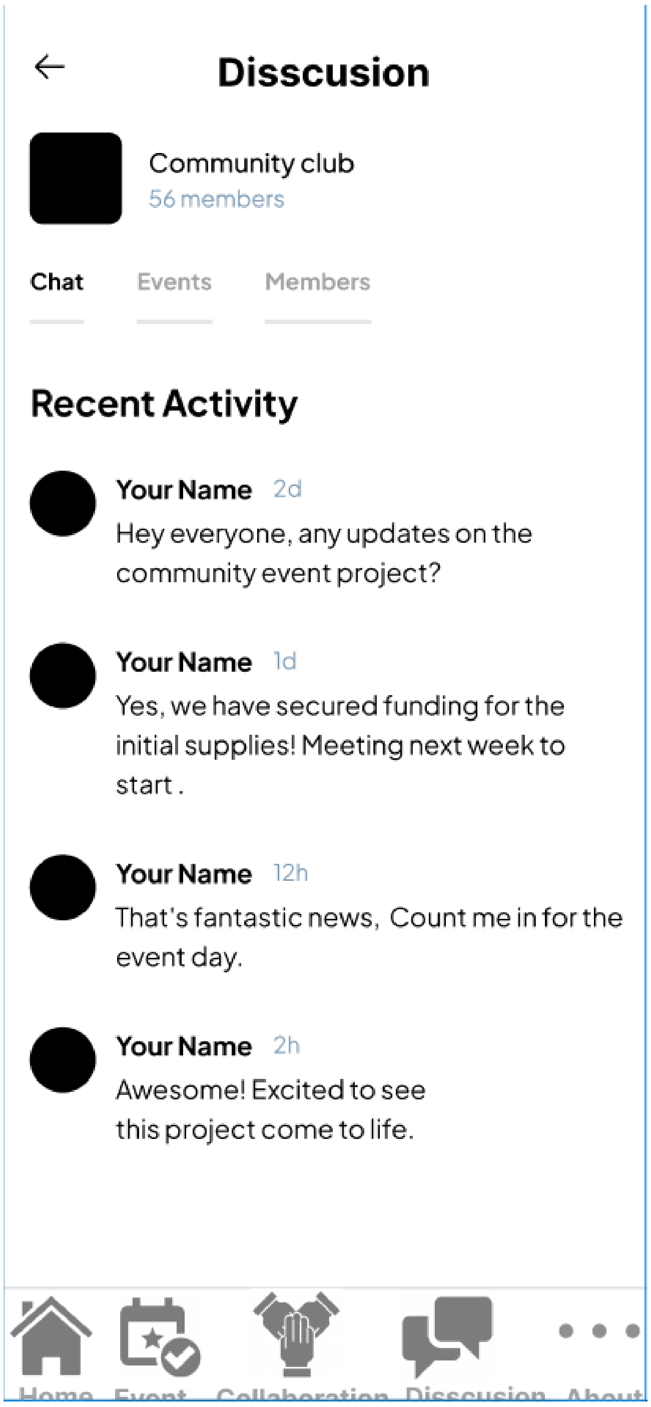


Figure 5.11: Profile Page

3. **Discussion Page:** the central hub for a specific community club's interactions. It primarily displays the ongoing chat conversation and provides tabs to access club events and member lists.



Disscusion

Community club
56 members

Chat Events Members

Recent Activity

Your Name 2d
Hey everyone, any updates on the community event project?

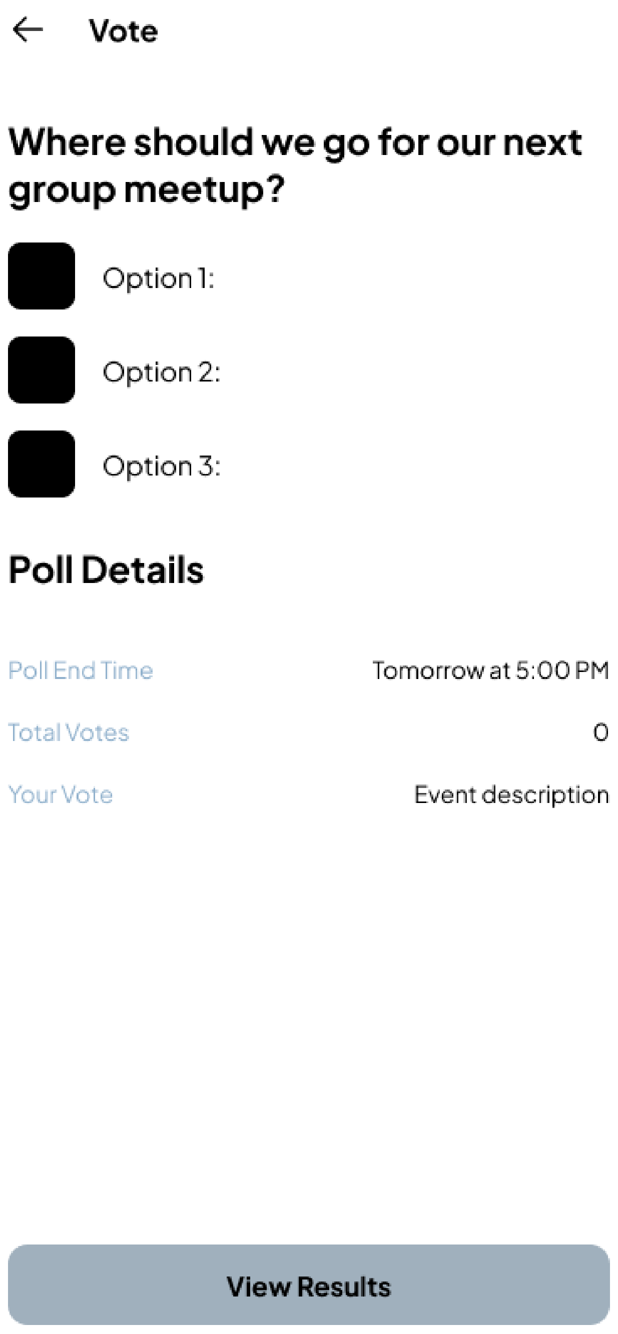
Your Name 1d
Yes, we have secured funding for the initial supplies! Meeting next week to start .

Your Name 12h
That's fantastic news, Count me in for the event day.

Your Name 2h
Awesome! Excited to see this project come to life.

Figure 5.12: Discussion Page

4. **Vote Card:** presents a poll, allowing users to cast a vote by selecting from predefined options. It also displays relevant poll details like the end time and provides access to view the voting results.



Where should we go for our next group meetup?

Option 1:

Option 2:

Option 3:

Poll Details

Poll End Time	Tomorrow at 5:00 PM
Total Votes	0
Your Vote	Event description

Figure 5.13: Vote Card

5.8 Development Applications Used

Throughout the development lifecycle of the InterClub project, a suite of applications and tools were employed to streamline design, coding, testing, and collaboration. These include:

- **Integrated Development Environments (IDEs):**
  - **Visual Studio Code (VS Code) [15]:** Utilized for frontend development (React.js [3], Flutter) and backend development (Node.js [5]), benefiting from its extensive library of extensions and integrated terminal.
  - **Remix IDE [16]:** Employed for Solidity [10] smart contract [10] development, debugging, and initial testing in a simulated environment.
- **Version Control Systems:**
  - **Git [17]:** For distributed version control and source code management.
  - **GitHub [18]:** For hosting repositories, managing collaborative workflows (pull requests, code reviews), and CI/CD pipelines.
- **Design and Prototyping Tools:**
  - **Figma [19]:** For UI/UX design, creating wireframes, mockups, and interactive prototypes for both web and mobile interfaces.
  - **PlantText [20]:** For creating system architecture diagrams, use case diagrams, class diagrams, and sequence diagrams using PlantUML [20] syntax.
- **API Development and Testing:**
  - **Postman [21]:** For designing, building, testing, and documenting APIs, ensuring backend services function as expected.
- **Blockchain Development and Testing Tools:**
  - **Ganache:** For setting up a personal Ethereum [9] blockchain [8] for local development and testing of smart contracts [10].
  - **Truffle Suite:** A development environment, testing framework, and asset pipeline for blockchains [8] using the Ethereum Virtual Machine (EVM) [9].
- **Project Management and Collaboration:**
  - **Jira [22]:** For agile [1] project management, sprint planning, task tracking, and monitoring project progress.
  - **Microsoft Teams [23]:** For real-time team communication, sharing updates, and facilitating quick discussions.
- **Database Management Tools:**
  - **MongoDB Compass:** For interacting with the MongoDB [14] database, managing collections, and running queries.
- **Documentation Tools:**
  - **LaTeX [24] (Overleaf [25]):** For typesetting this comprehensive project report.

Chapter 6

Conclusion and Future Work

6.1 Conclusion and Future Work

In conclusion, our platform uniquely merges robust event and collaboration management with the unmatched transparency and security of blockchain [8] technology. By recording every critical action—ranging from event creation and registration to attendance verification and governance voting—on-chain, we ensure immutable records that meet current demands for trust and accountability. This design not only addresses current market challenges but also lays the foundation for future regulatory demands and evolving market expectations.

6.1.1 Future-Proofing and Future Predictions

- **Enhanced Transparency and Trust:**
  As blockchain [8] continues to mature and expand its applications beyond cryptocurrency, our approach of logging all critical actions on-chain is expected to become a standard in ensuring trust and accountability. This method will meet current needs for immutable records and serve as a robust foundation as regulatory requirements and market expectations evolve.

- **Scalable and Adaptable Governance:**
  The integrated automated voting and promotion mechanisms lay the groundwork for fully decentralized governance models. With increased community participation, our system could evolve into a comprehensive platform for democratic decision-making, setting trends that may influence governance across various sectors.

- **Unified Digital Experience:**
  The convergence of event management, secure authentication, and real-time collaboration in a single interface positions our solution for emerging digital ecosystems. As interoperability between decentralized networks and centralized applications advances, our platform is designed to integrate seamlessly with new technologies, ensuring a smooth transition toward a more unified digital future.

- **Continuous Innovation with User Empowerment:**
  Our commitment to leveraging decentralized identity solutions (e.g., Web3Auth [13]) and blockchain-based [8] smart contracts [10] not only enhances security today but also ensures that our platform can quickly adapt to emerging trends in user interaction and security protocols. Future enhancements could include AI-driven predictive analytics and immersive augmented reality experiences for event management and collaboration.

- **Market Growth and Demand:**
  With increasing interest in decentralized applications for community and organizational governance, we predict a surge in demand for platforms that provide a transparent, secure, and integrated approach. Our solution is positioned to capture this evolving market by addressing the needs of niche communities that value decentralized trust and automated governance.

6.1.2 Additional Future Features

While our current solution focuses on comprehensive event management, blockchain-based [8] attendance verification, secure discussions, and decentralized governance, several additional features could further enhance the platform's capabilities:

- **Multi-Club Support:**
  Extend the system to support multiple clubs or communities within a single platform, thereby increasing scalability and broader community engagement.
- **NFT Rewards:**
  Incorporate optional NFT-based rewards to incentivize member participation and offer collectors verifiable digital assets.
- **Online Payment Integration:**
  Add secure online payment processing to facilitate seamless ticket sales and potential monetization opportunities.
- **Advanced Analytics and Reporting:**
  Integrate real-time analytics and reporting tools to provide insights into event participation, discussion trends, and governance outcomes, empowering organizers with data-driven decision-making.
- **Social Media Integration:**
  Enable seamless connectivity with popular social media platforms to expand event promotion and extend community reach.

Overall, our platform is engineered to be future-proof—secure, scalable, and adaptable. By anticipating emerging trends and continuously innovating, we are well-positioned to lead in digital event and collaboration management, transforming how communities engage, govern, and thrive in the digital age.

Chapter 7

References

[1] Agile: https://www.geeksforgeeks.org/what-is-agile-methodology/ (Accessed: 30 mars 2025).
https://learn.microsoft.com/en-us/devops/plan/what-is-scrum (Accessed: 30 mars 2025).

[2] Scrum: https://www.geeksforgeeks.org/scrum-software-development/ (Accessed: 30 mars 2025).
https://learn.microsoft.com/en-us/devops/plan/what-is-scrum (Accessed: 30 mars 2025).

[3] ReactJs: https://en.wikipedia.org/wiki/React_(software) (Accessed: 30 mars 2025).

[4] Bootstrap: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) (Accessed: 30 mars 2025).

[5] NodeJs: https://en.wikipedia.org/wiki/Node.js (Accessed: 30 mars 2025).

[6] ExpressJs: https://en.wikipedia.org/wiki/Express.js (Accessed: 30 mars 2025).

[7] WebSocket: https://nodejs.org/en/learn/getting-started/websocket (Accessed: 30 mars 2025).

[8] BlockChain: https://en.wikipedia.org/wiki/Blockchain (Accessed: 30 mars 2025).

[9] Ethereum: https://en.wikipedia.org/wiki/Ethereum (Accessed: 30 mars 2025).

[10] Solidity: https://en.wikipedia.org/wiki/Solidity (Accessed: 30 mars 2025).

[11] JWT: https://en.wikipedia.org/wiki/JSON_Web_Token (Accessed: 30 mars 2025).

[12] MetaMask: https://en.wikipedia.org/wiki/MetaMask (Accessed: 30 mars 2025).

[13] Web3Auth: https://www.gate.io/learn/articles/what-is-web3auth/6459 (Accessed: 30 mars 2025).

[14] MongoDB: https://en.wikipedia.org/wiki/MongoDB (Accessed: 30 mars 2025).

[15] Visual Studio Code: https://en.wikipedia.org/wiki/Visual_Studio_Code (Accessed: 30 mars 2025).

[16] Remix IDE: https://remix.ethereum.org/ (Accessed: 30 mars 2025).

[17] Git: https://en.wikipedia.org/wiki/Git (Accessed: 30 mars 2025).

[18] GitHub: `https://en.wikipedia.org/wiki/GitHub` (Accessed: 30 mars 2025).

[19] Figma (software): `https://en.wikipedia.org/wiki/Figma_(software)` (Accessed: 30 mars 2025).

[20] PlantText (PlantUML): `https://en.wikipedia.org/wiki/PlantUML` (Accessed: 30 mars 2025).

[21] Postman (software): `https://en.wikipedia.org/wiki/Postman_(software)` (Accessed: 30 mars 2025).

[22] Jira (software): `https://en.wikipedia.org/wiki/Jira_(software)` (Accessed: 30 mars 2025).

[23] Microsoft Teams: `https://en.wikipedia.org/wiki/Microsoft_Teams` (Accessed: 30 mars 2025).

[24] LaTeX: `https://en.wikipedia.org/wiki/LaTeX` (Accessed: 30 mars 2025).

[25] Overleaf: `https://en.wikipedia.org/wiki/Overleaf` (Accessed: 30 mars 2025).