1. Business Statement

Universities host a diverse range of student-run societies and clubs that play a vital role in enriching campus life through events, activities, and community engagement. These societies manage memberships, plan events, conduct inductions, and handle finances—functions that are essential to their smooth operation. However, most of these processes are currently handled manually using spreadsheets, messaging apps, or disparate tools. This fragmented approach leads to inefficiencies, poor data handling, and a lack of transparency, ultimately reducing the effectiveness of student societies and overburdening their organizers.

The core problem lies in the absence of a centralized digital platform to manage all society-related operations. Membership data is inconsistent and often outdated, event planning lacks coordination, and financial records are scattered, increasing the risk of errors. Induction procedures are manual and vary across societies, making them time-consuming and non-standard. Furthermore, the lack of system-generated reporting tools means that societies and university administrators have little access to valuable insights or performance metrics, limiting opportunities for growth and accountability.

The proposed University Societies Management System (USMS) aims to solve these challenges by offering a centralized, secure, and user-friendly web application. Developed as part of an academic project, USMS will enable streamlined membership and event tracking, standardized and efficient induction processes, and accurate financial management. The system will also offer role-based access for administrators and users, as well as automated reporting and analytics. By digitizing and centralizing these core functions, USMS will significantly enhance operational efficiency, reduce manual effort, and improve the overall experience for both society leaders and student members.

2. Project Team

Resource Type	Quantity	Why They Are Needed
Database Developers	1	To design relational schemas, write SQL queries, manage data storage, ensure normalization, enforce constraints, and maintain data security.
Web Developers	1	To build and integrate both front-end (React.js/Angular) and back-end (Django/Flask/Node.js) parts of the web application.

UI/UX Designer	1	To design user-friendly and responsive interfaces for student dashboards, admin panels, and event management tools.
Project Manager	1	To manage timelines, sprints, resource allocation, risk management, and ensure deliverables are submitted on time.
Testing/QA Engineer	1	To rigorously test system functionality, catch bugs, and perform usability and security testing before deployment.
Deployment/DevOps	1	To manage server-side deployment, CI/CD pipelines, cloud hosting setup, and maintenance.
Expert/Consultant	1	To validate technical decisions, review architecture, and guide implementation from an institutional IT standards perspective.

Why These Resources Are Needed

- **Database Developers:** USMS is a data-heavy system involving societies, users, events, and finances. Efficient database design is critical to performance, data integrity, and future scalability.

- **Web Developers:** To implement the business logic and interface of the system, ensuring smooth communication between front-end and back-end with secure API endpoints.

- **UI/UX Designer:** For designing a seamless and accessible interface that supports multiple roles (Admin, Society Leader, Member), with a mobile-first approach.

- **Project Manager:** Required for orchestrating the Agile Scrum framework, managing tasks, holding sprint meetings, handling scope changes, and ensuring stakeholder satisfaction.

- **Testing/QA Engineer (Optional):** To ensure the software meets quality standards before deployment, reducing bug-related delays and ensuring user trust.

- **Deployment/DevOps (Optional):** Required during deployment phases for environment setup, domain hosting, version control, backups, and uptime management.

- **Expert/Consultant:** Plays a key advisory and approval role to align the solution with university policies and IT best practices.

Project Team Structure

Project Manager (Muniba)
│
├── UI/UX Designer (Rakhshanda)
├── Front-End Developer (Warda)
├── Back-End Developer (Urwa)
├── Database Developers (Ramin)
├── Expert Consultant (Muniba-coordinated with IT Dept)
└── QA/DevOps (Part-time support or cross-assigned)

3. Feasibility Analysis

Project Scope: Development of a large-scale University Society Management System as a full-featured, professional-grade website for company use.

Budget: PKR 700,000

Timeline: 6 Months

1. Technical Feasibility

Assessment
The proposed system requires the integration of frontend (React.js), backend (Python/Flask or Django), database (MySQL), and optional cloud deployment (e.g., AWS, Render, or DigitalOcean). These technologies are mature, well-supported, and suitable for building scalable web applications.

Key Considerations

- Availability of experienced developers proficient in React, Python, and MySQL.
- Availability of frameworks like Flask or Django that support MVC architecture.
- Compatibility with third-party services for features like authentication (OAuth), notifications, and scheduling.
- Existing infrastructure (local or cloud) to deploy the solution efficiently.

Rating: 9/10

Justification
The technologies are well-proven and available. The development team has access to modern development tools and deployment options. The only minor risk lies in integration delays or cloud setup issues, which are manageable with experienced developers.

2. Economic Feasibility

Assessment

The budget of PKR 700,000 is allocated to cover all aspects: development, testing, deployment, domain/hosting, and initial post-deployment support. Cost-effective choices like open-source libraries, reusable components, and in-house development are emphasized.

Key Considerations

- Developer salaries and outsourcing if required.
- Domain, hosting, and cloud resource costs.
- Cost of testing, design, and future maintenance.
- Long-term ROI through automation and digitization of society operations.

Rating: 8.5/10

Justification

The budget is realistic for a mid-scale enterprise web project in Pakistan. Open-source tools reduce licensing costs, and in-house developers further cut outsourcing expenses. ROI is expected in the form of reduced manual work, centralized operations, and improved user satisfaction. Slight cost overruns may occur if scope increases unexpectedly.

3. Operational Feasibility

Assessment

The system aligns well with the operations of university societies. Admins will manage users, events, and activities; students will interact via dashboards and notifications. The platform improves coordination, visibility, and accountability.

Key Considerations

- Ease of adoption by non-technical users (students, admin staff).
- Inclusion of user roles (admin, student, event manager).
- Support for mobile-responsive design for wider accessibility.
- Training and documentation availability.

Rating: 9.5/10

Justification

User roles and interface are designed for intuitive access, so no major retraining is required. Since the system automates existing workflows rather than introducing unfamiliar ones, operational adoption will be smooth. Inclusion of onboarding support will further ease the transition.

4. Legal and Schedule Feasibility

Assessment

The platform does not conflict with any legal or copyright issues, as it uses open-source tech stacks and stores university data internally. Time-wise, the 6-month deadline is reasonable with a proper project plan and milestones.

Key Considerations

- Use of licensed or open-source components only.
- Data privacy and compliance (if handling student data).
- Development in sprints with deliverables each month.
- Availability of team for testing, bug fixes, and deployment.

Rating: 8/10

Justification

No known legal issues, and the timeline is achievable with proper planning. Potential delays can occur if scope increases or major bugs are found late. These risks can be reduced by following Agile methodology and regular client feedback.

Critical Success Factors (CSFs)

- **Modular Design:**

  Ensures easy upgrades and maintenance.

  Critical for scalability and feature additions.

  Measure: Develop in components and services.

- **User-Friendly UI/UX:**

  Encourages adoption by students and society admins.

  Measure: Conduct usability testing before launch.

- **Real-Time Communication and Event Tracking:**

  Critical for managing society events, registrations, and notices.

  Measure: Use WebSockets or polling for real-time updates.

- **Role-Based Access Control (RBAC):**

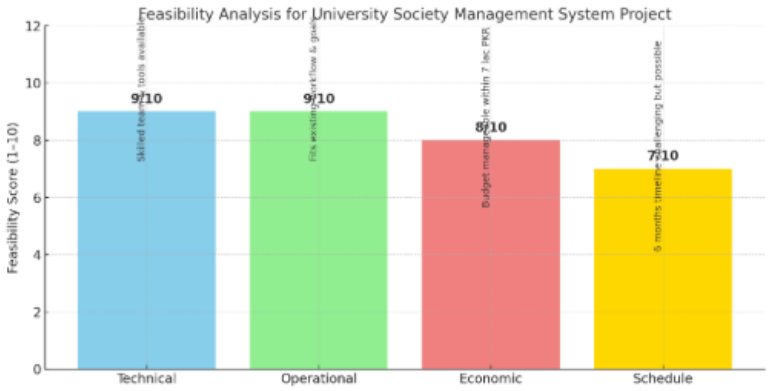
  Avoids the misuse of admin features by students.

  Measure: Implement middleware to handle route protection.

- **Data Security and Backup:**

  Ensures reliability and trust in the system.

Measure: Regular database backups and secure API handling.



4. Project Goal / Purpose / Aim / Objectives

Aim of the Project
The aim of the Universal Societies Management System (USMS) is to create an easy-to-use web platform that helps universities manage their student societies in a better and faster way.

Goals that the Project Will Meet

1. **Functional Goals**
    - Keep track of student members in each society
    - Help plan and manage society events
    - Manage society budgets and spending
    - Make inductions and applications automatic
    - Send notifications and generate reports
2. **Technological Goals**
    - Use modern tools like React or Angular for the front end
    - Use Django, Flask, or Node.js for the back end
    - Host the system online using AWS or university servers
3. **Quality Goals**
    - Make sure the system is easy to use, fast, and secure
    - Design it to work well on all screen sizes (responsive)
    - Ensure it is accessible to all users
4. **Organizational Goals**
    - Improve the way societies work by moving to a digital system

- Use agile methods to build the system quickly and effectively

Benefits to the Organization

- Makes it easier for societies to manage their members and events
- Saves time by reducing manual work
- Helps societies keep better records of their finances
- Makes communication and inductions more organized
- Encourages digital transformation within the university

5. Project Deliverables

Deliverable	Objective Linked	Success Criteria
Requirement Specification Document	Clearly define project scope and user expectations	Approved by stakeholders and development team
Software Design Document (SDD)	Outline architectural and technical framework	Detailed and understandable by developers
Frontend Web Interface (HTML/CSS/JS)	Enable users to interact with the system	Responsive design, browser compatibility, meets UI/UX standards
Backend APIs (Python/Django/Flask)	Process business logic, data handling	Pass all unit and integration tests, secure endpoints
Database Schema (MySQL)	Ensure structured storage of society and user data	Supports all CRUD operations efficiently and securely
Authentication & Authorization Module	Secure access to society functions	Only verified users access relevant features
Admin Dashboard	Manage societies, events, and members centrally	Functional and user-friendly interface for administrators
User Module	Allow members to join/view societies and register for events	Meets functional requirements of end users

Testing Reports	Ensure quality and reliability of software	Passes acceptance tests, minimal bugs reported
Deployment Setup (Hosting + Domain)	Make the system accessible online	Deployed successfully on selected infrastructure
Final Project Report and Documentation	Provide complete project lifecycle and future support guide	Comprehensive, accurate, and client-ready

Success Criteria Summary

- All core modules (event management, society dashboards, admin approvals) function as specified.
- User experience is smooth with <2% critical bugs post-launch.
- All deliverables are completed, reviewed, and approved by relevant stakeholders.
- Deployment is live, and user onboarding is successfully conducted.
- System supports real-time interactions with no downtime during initial month of usage.

6. Project Scope

1. Boundaries of the Project

The Universal Societies Management System (USMS) is a web-based application designed specifically for managing student societies at the university. The system's primary boundary is that it will operate within the university environment and serve registered societies, their members, and administrators. The system will handle membership registration, event planning, financial tracking, induction processes, and report generation — all securely managed through a centralized relational database.

The project does not extend beyond university-managed societies and does not cover multi-institution or commercial use.

2. Things Included & Excluded

Included:

- Project Management Activities: Initiation, planning, execution, monitoring, and closure.
- Requirements Gathering: Interviews, surveys, and analysis with stakeholders (society admins, members).
- Database Design & Implementation:
    - Conceptual, logical, and physical models.
    - Normalization (1NF to 3NF), entity relationships, keys, indexing, etc.
- Web Application Development:

    - Membership, event, finance, induction, and reporting modules.
    - Dashboard, society directory, and role-based access system.
- Testing: Unit, integration, system, security, usability, and UAT.
- Deployment:
    - Pilot and full deployment.
    - Configuration of servers and databases.
    - Post-deployment support and user training.
- Documentation:
    - User guides, technical manuals, training material.

Excluded:

- Mobile application or mobile-first responsive design.
- Integration with external financial tools or online payment gateways.
- Social media integrations for event sharing.
- Cross-university or multi-campus functionality.
- AI or automated decision-making tools.
- Hardware procurement or installation.

3. Various Parts & Phases of the Project

This project is divided into seven structured phases, each composed of detailed activities as defined in the WBS:

Phase	Description
1	Project Management: Planning, risk analysis, scheduling, communication, and stakeholder management.
2	Requirements Analysis: Stakeholder interviews, surveys, workshops, requirement specifications.
3	Database Design: ER modeling, normalization, schema creation, and implementation.
4	Application Design: UI/UX wireframes, system architecture, interface design.
5	Development: Frontend, backend, modules for society features, reporting, and integrations.
6	Testing: Unit, integration, system, and user acceptance testing across modules.
7	Deployment: Setup of production environment, pilot testing, final deployment, training.

4. Success Criteria

The project will be considered successful when the following measurable criteria are achieved:

- The system supports full membership management, event planning, financial tracking, online inductions, and report generation as per the requirements.
- The database is normalized, secured, and performs without data integrity issues.
- The web application UI is functional and user-friendly across all major modules.
- At least 3 university societies use the system during the pilot deployment and provide positive feedback.
- All testing phases are passed, and bugs identified during UAT are resolved.
- System is deployed successfully in the university environment.
- Complete training materials are delivered and approved by the supervisor or expert.
- The expert signs off on the deliverables at the end of the project.

7. Agile User Stories

Epic 1: Society Lifecycle Management

- As an Admin, I want to manage society records.
- As an Admin, I want to create new societies so that society leaders can manage their activities under the university's system.
- As an Admin, I want to update or delete society records so that outdated or duplicate entries can be managed properly.
- As a Society Leader, I want to edit my society's profile so that I can update objectives, events, or member details.

Epic 2: Member Management

- As a Society Leader, I want to manage member records.
- As a Member, I want to register for societies online so that I can participate in events and activities.
- As a Society Leader, I want to approve or reject member requests so that I can control who joins the society.
- As a Member, I want to view my membership status in different societies so that I know where I am accepted or pending.

Epic 3: Induction System

- As a Society Leader, I want a standardised induction system.

- As a Member, I want to fill out an online induction form so that I can apply to join a society.
- As an Admin, I want to customise induction templates so that each society can define specific induction criteria.
- As a Society Leader, I want to view and manage induction applications so that I can select suitable members.

Epic 4: Event Management

- As a Society Leader, I want to manage event scheduling.
- As a Society Leader, I want to schedule new events so that members can be notified in advance.
- As a Member, I want to RSVP to events so that the society knows how many people will attend.
- As an Admin, I want to view all scheduled events across societies so that I can avoid date clashes.

Epic 5: Financial Tracking

- As an Admin, I want to oversee financial reports.
- As a Society Leader, I want to add and track expenses so that I can manage the society's finances.
- As an Admin, I want to generate financial reports by society and date so that I can audit spending.
- As a Society Leader, I want to upload receipts or transaction records so that there is proof of spending.

Epic 6: Reporting and Analytics

- As an Admin, I want to generate various reports.
- As an Admin, I want to view membership statistics so that I can understand society growth and engagement.
- As a Society Leader, I want to download event attendance reports so that I can evaluate event success.
- As an Admin, I want to filter reports by date or society so that I can make data-driven decisions.

Epic 7: Security and Access Control

- As a User, I want my data to be protected.
- As a User, I want role-based access control so that I can only see and do what I'm permitted to.

- As an Admin, I want to create and assign roles so that users have appropriate access.
- As a User, I want encrypted login and backup features so that my data stays secure.

8. Assumptions, Constraints, and Dependencies

Assumptions

- **User Availability & Cooperation:** It is assumed that all stakeholder groups (students, society admins, and university IT staff) will be available for requirement gathering, testing, and feedback during the 6-month development cycle.
- **Stable Internet Access:** All intended users (including off-campus students) will have stable internet access to use the web-based platform effectively.
- **Skill Availability:** The development team is assumed to have sufficient technical skills in React.js, Python (Flask/Django), MySQL, HTML/CSS, and API integration.
- **No Major Changes in Scope:** It is assumed that once the scope is finalized and approved, there will be no major scope expansions that would impact the timeline or budget significantly.
- **System Usage Starts Post-Deployment:** We assume that the university societies will begin using the system immediately after the successful deployment and UAT approval.

Constraints

- **Time Constraints:** The project must be completed within 6 months, with key milestones each month. Delays in testing or client feedback could compress development windows for later phases.
- **Financial Constraints:** A fixed budget of PKR 700,000 must cover developer salaries, hosting/domain fees, testing and documentation, UI/UX design, deployment and post-deployment support. No additional funds are assumed unless explicitly approved.
- **Resource Constraints:** A small, fixed-size development team (e.g., 3–5 members) is available. Resources such as professional testers, UI designers, or DevOps engineers may be limited or shared.
- **Technical Constraints:** Technology stack is fixed (React.js for frontend, Python backend, MySQL database). The system must be compatible with all major modern browsers and responsive across devices. No use of proprietary paid SDKs or libraries unless cost is approved.
- **Business Constraints:** The system must align with university policy regarding student data privacy and society governance rules. All event approval workflows must follow the current university administrative hierarchy.

Dependencies

- **University IT Infrastructure:** The final deployment may depend on integration with the university's existing authentication systems (e.g., student portals or LDAP).
- **Timely Feedback from Stakeholders:** Feedback loops from society leaders and university admin are essential during prototyping, testing, and UAT stages.
- **Third-Party Services:** The system may rely on email or SMS APIs (e.g., SendGrid, Twilio) for notifications, cloud deployment providers (e.g., AWS, Render, or Heroku), and calendar or event scheduling APIs (Google Calendar, etc.).
- **Academic Calendar:** The platform must launch at the beginning of a semester or academic session for effective adoption—hence project timelines are partially dependent on university academic scheduling.
- **Regulatory/Policy Changes:** If university policy or student data protection laws change during development, it may require last-minute alterations to modules handling data, privacy, or consent.

9. Stakeholders

Stakeholder	Role in the project	Key responsibilities
University Administration	Project Sponsor and Approval Authority. Provides funding, policy alignment, and final approval of deliverables.	- Approve the project scope and final deliverables - Allocate budget - Ensure system aligns with policies
Society Presidents	Power Users and Functional Validators. Represent the operational needs of their societies and approve prototypes	- Provide requirements - Review prototypes - Validate society management features
Society Members	End Users. Interact with the system for memberships, event registrations, inductions, and communication	- Use the system and provide feedback on usability and functionality
Project Manager	Oversees the entire project, manages timelines, resources, and ensures stakeholder expectations are me	- Schedule planning and review meetings - Manage risks, communication, and resource allocation
Development Team	Responsible for system design, development, testing, deployment,	- Build web app and database

	and technical documentatio	- Implement induction and reporting modules - Integrate RBAC and security
Database Administrator	Designs, implements, and maintains the database for performance, integrity, and security	-Create ER models - Enforce data consistency - Configure backups and access control
Quality Assurance Team	Tests the system for functionality, security, and usability before release	- Perform system testing - Report bugs - Validate performance under load
IT Support Team	Provides technical support post-deployment, handles training, maintenance, and upgrade	- Train users - Provide user guides and support channels - Maintain system post-launch

Deliverable

Deliverable	Responsible Stakeholder(s)	Signing Authority
Requirements Specification Document	Project Manager, Society Presidents	University Administration
Functional Web Application	Development Team	University Administration
Database Design and Schema	Database Administrator	Project Manager
Online Induction Module	Development Team	Society Presidents
Role-based Access Control System	Development Team	Project Manager
Financial Tracking & Reporting Tools	Development + QA Team	University Administration
User Manuals and Training Guides	IT Support Team	Society Presidents & Project Lead

Final Project Report & Documentation	Project Manager	University Administration

10. Expert

Details as per project context.

11. Work Breakdown Structure (WBS)

1.0 Project Management

1.1 Project Initiation

1.1.1 Conduct initial project planning meeting

1.1.2 Define project scope and objectives

1.1.3 Identify project stakeholders

1.1.4 Develop project charter

1.1.5 Obtain project approvals

1.1.6 Establish project communication channels

1.1.7 Define project success criteria

1.2 Project Planning

1.2.1 Develop detailed project schedule

1.2.2 Create Work Breakdown Structure (WBS)

1.2.3 Define project activities and tasks

1.2.4 Estimate activity durations

1.2.5 Allocate resources

1.2.6 Develop risk management plan

1.2.6.1 Identify potential risks

1.2.6.2 Assess risk impact and probability

1.2.6.3 Develop risk response strategies

1.2.6.4 Assign risk owners

1.2.7 Develop communication plan

1.2.7.1 Define communication stakeholders

1.2.7.2 Determine communication frequency

1.2.7.3 Establish communication methods

1.2.8 Develop quality management plan

1.2.8.1 Define quality standards

1.2.8.2 Develop quality control processes

1.2.8.3 Establish quality assurance procedures

1.2.9 Develop procurement plan

1.2.9.1 Identify procurement needs

1.2.9.2 Select vendors

1.2.9.3 Negotiate contracts

1.2.10 Finalize project management plan

1.2.11 Develop a change management plan

1.3 Project Execution

1.3.1 Kick-off meeting

1.3.2 Manage project execution

1.3.3 Coordinate team activities

1.3.4 Monitor project progress

1.3.5 Track project deliverables

1.3.6 Manage project issues

1.3.7 Manage project changes

1.3.8 Conduct progress meetings

1.3.9 Report project status

1.3.10 Manage stakeholder expectations

1.4 Project Monitoring and Control

1.4.1 Monitor schedule performance

1.4.2 Monitor cost performance

1.4.3 Monitor quality performance

1.4.4 Perform integrated change control

1.4.5 Update project management plan

1.4.6 Conduct risk monitoring and control

1.4.7 Conduct quality audits

- 1.5 Project Closure
    - 1.5.1 Obtain project acceptance
    - 1.5.2 Conduct project closure meeting
    - 1.5.3 Document lessons learned
    - 1.5.4 Archive project documents
    - 1.5.5 Release project resources
    - 1.5.6 Conduct post-implementation review

2.0 Requirements Analysis

- 2.1 Elicit Requirements

        - 2.1.1 Conduct stakeholder interviews
            - 2.1.1.1 Prepare interview questions
            - 2.1.1.2 Conduct interviews with society admins
            - 2.1.1.3 Conduct interviews with society members
        - 2.1.2 Conduct surveys
            - 2.1.2.1 Design survey questionnaire
            - 2.1.2.2 Distribute surveys
            - 2.1.2.3 Analyze survey results
        - 2.1.3 Organize requirements workshops
            - 2.1.3.1 Prepare workshop materials
            - 2.1.3.2 Facilitate workshops
            - 2.1.3.3 Document workshop outcomes
        - 2.1.4 Review existing systems
            - 2.1.4.1 Identify existing systems
            - 2.1.4.2 Analyze existing system functionalities
            - 2.1.4.3 Document existing system limitations
        - 2.1.5 Gather user feedback
            - 2.1.5.1 Collect feedback from potential users
            - 2.1.5.2 Analyze user feedback
- 2.2 Analyze Requirements
        - 2.2.1 Analyze functional requirements (membership, events, finance, inductions, reporting)
            - 2.2.1.1 Analyze membership management requirements
            - 2.2.1.2 Analyze event management requirements

        - 2.2.1.3 Analyze financial tracking requirements
        - 2.2.1.4 Analyze online induction requirements
        - 2.2.1.5 Analyze reporting tool requirements
    - 2.2.2 Analyze non-functional requirements (security, performance, usability)
        - 2.2.2.1 Analyze security requirements
        - 2.2.2.2 Analyze performance requirements
        - 2.2.2.3 Analyze usability requirements
    - 2.2.3 Define user roles and permissions
    - 2.2.4 Document business processes
        - 2.2.4.1 Document current state business processes
        - 2.2.4.2 Document future state business processes
    - 2.2.5 Define data requirements
- 2.3 Specify Requirements
    - 2.3.1 Create use case diagrams
    - 2.3.2 Develop user stories
    - 2.3.3 Write detailed requirements specifications
        - 2.3.3.1 Write functional requirements specification
        - 2.3.3.2 Write non-functional requirements specification
    - 2.3.4 Develop system requirements specification document
- 2.4 Validate Requirements
    - 2.4.1 Review requirements with stakeholders
        - 2.4.1.1 Prepare requirements review documents
        - 2.4.1.2 Conduct requirements review meetings

- 2.4.2 Obtain sign-off on requirements

3.0 Database Design

3.1 Conceptual Database Design

3.1.1 Identify entities (memberships, events, finances, inductions, societies, users, roles)

3.1.2 Define attributes for each entity

3.1.2.1 Define membership attributes

3.1.2.2 Define event attributes

3.1.2.3 Define finance attributes

3.1.2.4 Define induction attributes

3.1.2.5 Define society attributes

3.1.2.6 Define user attributes

3.1.2.7 Define role attributes

3.1.3 Determine relationships between entities

3.1.4 Develop Entity-Relationship Diagram (ERD)

3.2 Logical Database Design

3.2.1 Normalize database tables

3.2.1.1 Apply first normal form

3.2.1.2 Apply second normal form

3.2.1.3 Apply third normal form

3.2.2 Define primary and foreign keys

3.2.3 Refine ERD

3.3 Physical Database Design

3.3.1 Select Database Management System (DBMS)

3.3.2 Define data types and sizes for each attribute

3.3.3 Design database schema

3.3.4 Plan for database storage

3.3.5 Design database indexes

3.4 Database Implementation

3.4.1 Create database tables

3.4.2 Implement relationships

3.4.3 Implement constraints

3.4.3.1 Implement primary key constraints

3.4.3.2 Implement foreign key constraints

3.4.3.3 Implement data integrity constraints

3.4.4 Develop database scripts

3.4.4.1 Develop table creation scripts

3.4.4.2 Develop data insertion scripts

4.0 Application Design

4.1 User Interface (UI) Design

4.1.1 Develop wireframes for key screens (membership, events, finance, inductions, reports, society directory, dashboard, user management, role management)

4.1.2 Design user interface mockups

4.1.3 Create UI design guidelines

4.1.3.1 Define color scheme

4.1.3.2 Define typography

4.1.3.3 Define layout

4.1.4 Conduct UI usability testing

4.1.4.1 Prepare usability test scenarios

4.1.4.2 Conduct usability test sessions

4.1.4.3 Analyze usability test results

4.2 System Architecture Design

4.2.1 Define system modules

4.2.2 Design data flow diagrams

4.2.3 Select technology stack

4.2.3.1 Select front-end technology

4.2.3.2 Select back-end technology

4.2.3.3 Select framework

4.2.4 Design application architecture

4.2.4.1 Design layered architecture

4.2.4.2 Design component-based architecture

4.3 Detailed Design

4.3.1 Design class diagrams

4.3.2 Design sequence diagrams

4.3.3 Design component diagrams

4.3.4 Develop detailed design specifications for each module

4.3.4.1 Develop detailed design for membership module

4.3.4.2 Develop detailed design for event module

4.3.4.3 Develop detailed design for finance module

4.3.4.4 Develop detailed design for induction module

4.3.4.5 Develop detailed design for reporting module

5.0 Development

5.1 Database Development

5.1.1 Implement database schema

5.1.2 Develop database functions and procedures

5.1.3 Optimize database performance

5.1.4 Perform database testing

5.1.4.1 Perform unit testing on database components

5.1.4.2 Perform integration testing on database modules

5.2 Web Application Development

5.2.1 Develop user authentication module

5.2.1.1 Implement user login functionality

5.2.1.2 Implement user registration functionality

5.2.1.3 Implement password recovery functionality

5.2.2 Develop membership management module

5.2.2.1 Develop member registration functionality

5.2.2.2 Develop member profile management functionality

5.2.2.3 Develop member role assignment functionality

5.2.2.4 Develop member search functionality

5.2.2.5 Develop membership renewal functionality

5.2.3 Develop event management module

5.2.3.1 Develop event creation functionality

5.2.3.2 Develop event scheduling functionality

5.2.3.3 Develop RSVP functionality

5.2.3.4 Develop attendance tracking functionality

5.2.3.5 Develop event notification functionality

5.2.4 Develop financial tracking module

5.2.4.1 Develop transaction recording functionality

5.2.4.2 Develop budget management functionality

5.2.4.3 Develop expense tracking functionality

5.2.4.4 Develop financial reporting functionality

5.2.5 Develop induction module

5.2.5.1 Develop online registration functionality

5.2.5.2 Develop induction process management functionality

5.2.5.3 Develop induction material management functionality

5.2.6 Develop reporting tools

5.2.6.1 Develop financial report generation functionality

5.2.6.2 Develop membership statistics generation functionality

5.2.6.3 Develop event summary generation functionality

5.2.6.4 Develop custom report generation functionality

5.2.7 Develop society creation and management module

5.2.7.1 Develop society creation functionality

5.2.7.2 Develop society editing functionality

5.2.7.3 Develop society directory functionality

5.2.7.4 Develop society member management functionality

5.2.8 Develop dashboard

5.2.8.1 Develop key metrics display

5.2.8.2 Develop data visualization components

5.2.9 Develop notification system

5.2.9.1 Implement email notifications

5.2.9.2 Implement in-app notifications

5.2.10 Implement search and filter functionality

5.2.11 Implement data security features (encryption, access controls)

5.2.11.1 Implement data encryption

5.2.11.2 Implement role-based access control

5.2.11.3 Implement input validation

5.3 Integration

5.3.1 Integrate database with web application

5.3.2 Integrate modules

5.3.2.1 Integrate membership and event modules

5.3.2.2 Integrate finance and reporting modules

6.0 Testing

6.1 Unit Testing

6.1.1 Test database modules

6.1.2 Test web application modules

6.1.2.1 Test user authentication module units

6.1.2.2 Test membership module units

6.1.2.3 Test event module units

6.1.2.4 Test finance module units

6.1.2.5 Test induction module units

6.1.2.6 Test reporting module units

6.1.2.7 Test society management module units

6.1.3 Test individual functions and features

6.2 Integration Testing

6.2.1 Test integration between database and web application

6.2.2 Test module interactions

6.2.2.1 Test membership and event module integration

6.2.2.2 Test finance and reporting module integration

6.3 System Testing

6.3.1 Perform functional testing

6.3.2 Perform security testing

6.3.2.1 Test for vulnerabilities

6.3.2.2 Test access controls

6.3.3 Perform usability testing

6.3.4 Perform performance testing

6.3.4.1 Test load handling

6.3.4.2 Test response times

6.3.5 Perform compatibility testing

- 6.4 User Acceptance Testing (UAT)
  - 6.4.1 Develop UAT test plan
  - 6.4.2 Conduct UAT with stakeholders
    - 6.4.2.1 Conduct UAT with society admins
    - 6.4.2.2 Conduct UAT with society members
  - 6.4.3 Gather user feedback
  - 6.4.4 Address user feedback and issues

7.0 Deployment

- 7.1 Prepare Deployment Environment
  - 7.1.1 Set up server environment
    - 7.1.1.1 Configure hardware
    - 7.1.1.2 Install operating system
  - 7.1.2 Install required software
    - 7.1.2.1 Install DBMS
    - 7.1.2.2 Install web server
    - 7.1.2.3 Install application server
  - 7.1.3 Configure database server
  - 7.1.4 Configure web server
  - 7.1.5 Configure application server
- 7.2 Data Migration
  - 7.2.1 Plan data migration
  - 7.2.2 Extract data from existing systems
  - 7.2.3 Transform data

7.2.4 Load data into the new system

7.2.5 Verify data migration

7.3 System Deployment

7.3.1 Deploy database

7.3.2 Deploy web application

7.3.3 Conduct pilot deployment

7.3.4 Obtain feedback from pilot users

7.3.5 Perform full deployment

7.4 Post-Deployment Support

7.4.1 Provide user training

7.4.1.1 Develop training materials

12. MS Project Plan

Done in MS project file.

13. Risk Management

5 Risky Tasks

Task	Why it's risky
Implementing the online induction module	It involves user data handling and integration with the membership database, increasing chances of bugs or data leaks.
Designing role-based access control (RBAC)	Errors in access logic could allow unauthorized users to view/edit sensitive financial or membership data.
Developing financial tracking functionality	Financial misreporting could damage trust and raise disputes over society budgets or expenses.

Deployment and server setup	Initial deployment may face server crashes or access issues, disrupting the launch phase.
Collecting user requirements from multiple societies	Misunderstandings or missed inputs can lead to incomplete or misaligned features.

Qualitative Risk Analysis:

Probability impact matrix

High			
Medium		Risk 5	Risk 1,Risk 2,Risk 3,Risk 4
Low			
	Low	Medium	High

Impact

Risk Assessment

Risks	Likelihood	Impact	Risk Measure
Online induction module may fail or expose personal data	7	8	56
Role-based access control may allow unauthorized data access	6	9	54
Financial module may miscalculate or misreport data	5	9	45
Deployment may overload server or disrupt access	6	8	48
Delay in collecting society requirements	7	7	49

Response Strategies

Risks	Risk Category	Risk Strategy	Reason
Online induction module may fail or expose personal data	Technology risk	Avoidance	Eliminate the threat by removing insecure components, redesigning the module with strong encryption and validation, reducing exposure.
Role-based access control may allow unauthorized data access	Structure risk	Avoidance	Eliminate the threat by designing RBAC accurately from the start, applying least-privilege, and removing flawed logic.
Financial module may miscalculate or misreport data	Financial	Mitigation	Can't fully avoid financial logic complexity, so reduce impact through validation rules, audit logs, and thorough testing.
Deployment may overload server or disrupt access	Technology	Transference	Transfer risk by using cloud providers, outsourcing infrastructure, and applying SLAs and warranties for hosting services.
Delay in collecting society requirements	People	Mitigation	Reduce probability by conducting interviews, surveys, and requirement validation with prototypes to catch errors early.

14. Time Phased Budget

Excel sheet provided.