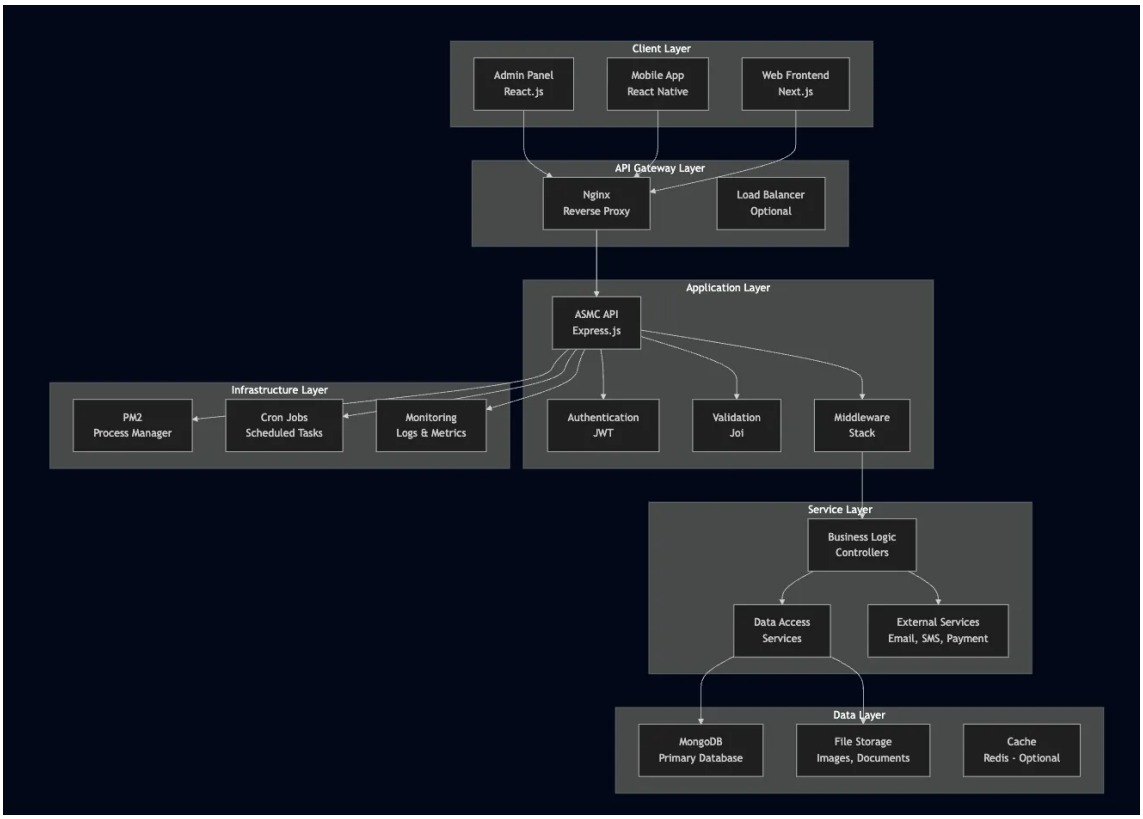
ASMC System Architecture

Overview

The ASMC (Association Management System) follows a layered architecture pattern that separates concerns and provides scalability, maintainability, and security.

Architecture Diagram



Component Details

Client Layer

- **Admin Panel (React.js)**: Web-based administration interface for staff users
- **Mobile App (React Native)**: Cross-platform mobile application for members
- **Web Frontend (Next.js)**: Public-facing website and member portal

API Gateway Layer

- **Nginx**: Reverse proxy server for load balancing and SSL termination
- **Load Balancer**: Optional component for high availability setups

Application Layer

- **ASMC API**: Express.js-based REST API server
- **Authentication**: JWT-based authentication system
- **Validation**: Request validation using Joi schemas

- **Middleware Stack**: Security, logging, and error handling middleware

Service Layer

- **Controllers**: Business logic implementation
- **Services**: Data access layer and external service integrations
- **External Services**: Email, SMS, and payment gateway integrations

Data Layer

- **MongoDB**: Primary database for application data
- **File Storage**: Storage for images, documents, and other files
- **Cache**: Redis-based caching layer for performance optimization

Infrastructure Layer

- **PM2**: Process manager for Node.js applications
- **Cron Jobs**: Scheduled tasks for maintenance and automation
- **Monitoring**: Logging and metrics collection system

Data Flow

1. **Client Request**: Users interact through web, mobile, or admin interfaces
2. **API Gateway**: Nginx routes requests to appropriate backend services
3. **Authentication**: JWT tokens validate user sessions
4. **Validation**: Request data is validated against defined schemas
5. **Business Logic**: Controllers process requests and implement business rules
6. **Data Access**: Services interact with MongoDB and external systems
7. **Response**: Processed data is returned to clients

Security Considerations

- **JWT Authentication**: Secure token-based authentication
- **Input Validation**: Comprehensive request validation using Joi
- **HTTPS**: SSL/TLS encryption for all communications
- **CORS**: Cross-origin resource sharing configuration
- **Rate Limiting**: API rate limiting to prevent abuse

Scalability Features

- **Horizontal Scaling**: Multiple API server instances
- **Load Balancing**: Nginx-based load distribution
- **Caching**: Redis caching for improved performance
- **Database Optimization**: MongoDB indexing and query optimization
- **File Storage**: Scalable file storage solutions

Monitoring and Maintenance

- **PM2 Process Management**: Automatic restart and clustering
- **Logging**: Comprehensive application and error logging
- **Metrics**: Performance and usage monitoring
- **Cron Jobs**: Automated maintenance and cleanup tasks
- **Health Checks**: Application health monitoring endpoints