Functional Requirements Document (FRD)

Project: Global Kokani Committees' Council (GKCC)
Components:

- GKCC Frontend
- GKCC Backend
- GKCC Admin Panel (Portal)

Version: 1.0.0
Date: 2025-02-08
Authors: Juned Khan, Sanad Naqvi

Authorization Memorandum:

I have reviewed the Functional Requirements Document for the GKCC project.

- ☑ The document is accepted.
- ☑ The document is accepted pending the changes noted.
- ☑ The document is not accepted.

Authorized by: Prashant Patil

_____
Name: Mr. Prashant Patil **Date:**

_____
Name: Mr.Awab Fakih **Date:**

CONFIDENTIALITY NOTICE

This document contains confidential information intended solely for the use of authorized individuals working on or with the GKCC project. Any unauthorized review, use, disclosure, distribution, or copying of this document or any part thereof is strictly prohibited. If you are not an authorized recipient, please notify the sender immediately and delete all copies of this document

1.0 VERSION HISTORY

Version	Approved By	Revision Date	Description of Change	Author

Table of Contents

---

1. Introduction

1.1 Purpose

This Functional Requirements Document (FRD) defines the functional and technical expectations for the Global Kokani Committees' Council (GKCC) project. The system is divided into three main components: the **Frontend**, the **Backend**, and the **Admin Panel (Portal)**. The purpose of this document is to serve as a blueprint for the design, development, and deployment of the complete system, ensuring that each component meets the needs of the end users, stakeholders, and developers.

1.2 Scope

The document covers:

- **User Interfaces (Frontend):** A Next.js–based client application that enables users to navigate, view content, and interact with GKCC services.
- **Backend Services:** A Node.js and Express–based API server with a MongoDB database for managing data, business logic, and integration with third-party services.
- **Admin Panel (Portal):** A management interface (also built with Next.js) designed for administrators to manage content, users, newsletters, media, and other administrative functions.

---

2. Business Requirements Overview

The GKCC project is designed to support the management and interaction of multiple community organizations under the Global Kokani Committees' Council. The key business objectives include:

- **Centralized Management:** Provide a unified platform to manage committees, memberships, events, and media.
- **Enhanced User Experience:** Ensure that the user-facing frontend is responsive, user-friendly, and efficient.
- **Robust Data Handling:** Implement a scalable and secure backend to handle user authentication, data storage, and business logic.
- **Efficient Administration:** Equip administrators with a dedicated portal for content and user management, including features such as newsletter creation, media galleries, and sponsor management.

---

3. Functional Requirements and User Impacts

3.1 GKCC Frontend

Overview

The Frontend is built using Next.js (leveraging version 14 structure), with a focus on modularity, dynamic routing, and integration with backend APIs via Axios.

2.0 Key Functional Areas

- **Usability Requirements:**
  - **Responsive UI:** Ensure the interface adapts to various devices (desktop, tablet, mobile).
  - **Intuitive Navigation:** Use folder-based routing for clear and logical page organization.
  - **Ease of Use:** Incorporate dynamic elements (e.g., sliders, modals) and user-friendly forms (using libraries like `react-hook-form` and `yup` for validation).
- **Performance Requirements:**
  - **Fast Loading:** Optimize asset delivery and ensure that pages load quickly.
  - **Efficient Data Fetching:** Use Axios with environment-based API endpoints to ensure reliability and efficiency.
- **Security Requirements:**
  - **Environment-based API URLs:** Use secure environment variables (configured via a `.env` file) to manage API endpoints.
  - **Data Protection:** Secure communication between the frontend and backend, ensuring that sensitive data (like user credentials) is transmitted safely.
- **Interface Requirements:**
  - **API Integration:** Use Axios to connect with the backend API (e.g., dynamically calling endpoints such as `/association/getOneAssociation/{id}`).
  - **UI Components:** Organized in directories (e.g., `components/home`, `components/aboutus`, etc.) to ensure reusability and maintainability.
- **Developer Workflow and Deployment:**
  - **Development Environment:** Instructions for installing dependencies, running the development server (`npm run dev`), and building for production.

- **Version Control:** Maintains a detailed commit history (see Version Control History section).

3.2 GKCC Backend

Overview

The Backend is implemented with Node.js, Express, and MongoDB. It follows a modular architecture to facilitate scalability, maintainability, and robust performance.

3.0 Key Functional Areas

- **API Services and Data Management:**
  - **RESTful APIs:** Define endpoints for operations related to admin functions, advertisements, memberships, newsletters, and media.
  - **Database Integration:** Use MongoDB and Mongoose for data storage, ensuring that schemas are defined clearly (e.g., `admin.model.js`, `membership.model.js`).
- **Performance Requirements:**
  - **Scalability:** Design with a modular architecture to scale with increasing data and user requests.
  - **Efficient Routing:** Optimize Express routes and middleware for high-performance API delivery.
- **Security Requirements:**
  - **Authentication:** Use JWT-based authentication to protect API endpoints.
  - **Password Management:** Implement secure password handling using bcrypt.
  - **Rate Limiting:** Incorporate middleware (e.g., `limiter.js`) to mitigate API abuse.
  - **Environment Variables:** Secure sensitive information such as database URIs, API keys, and token secrets via an `.env` file.
- **Error Handling and Logging:**
  - **Error Classes:** Use custom error handlers (e.g., `ApiError.js`) to manage errors gracefully.
  - **Response Wrappers:** Standardize responses using classes like `ApiResponse.js`.
  - **Async Handling:** Ensure asynchronous operations are managed with middleware like `asyncHandler`.
- **Developer Workflow and Deployment:**
  - **Scripts:** Provide clear instructions in `package.json` for starting the development (`npm run dev`) and production (`npm start`) environments.
  - **Version Control:** Detailed commit history is maintained to track changes and updates.

3.3 GKCC Admin Panel (Portal)

Overview

The Admin Panel provides a content management system (CMS) interface for administrators. It is built using Next.js and mirrors much of the frontend architecture while focusing on administrative features.

4.0 Key Functional Areas

- **Usability Requirements:**
  - **Responsive Design:** Ensure the admin interface is accessible on various devices.
  - **Content Management:** Provide features for managing media (photo/video albums), sponsors, newsletters, and user data.

    - **Intuitive Navigation:** Use folder-based and dynamic routing to organize the admin panel for ease of use.
- **Performance Requirements:**
    - **Efficient Data Handling:** Load administrative data quickly and allow for real-time updates.
    - **Seamless Transitions:** Implement smooth transitions and animations (using libraries like `framer-motion` and `gsap`).
- **Security Requirements:**
    - **Authentication and Authorization:** Secure admin routes with proper authentication mechanisms, ensuring that only authorized users can access sensitive operations.
    - **Environment Configuration:** Leverage environment variables for secure API endpoint configuration.
    - **Data Protection:** Ensure that administrative actions are logged and monitored.
- **Interface Requirements:**
    - **API Integration:** Use Axios for communicating with the backend, with API endpoints secured by environment variables.
    - **CMS Features:** Include functionalities for creating, editing, and deleting albums, sponsors, newsletters, and other content types.
- **Developer Workflow and Deployment:**
    - **Installation and Setup:** Detailed instructions provided for setting up the project locally, including dependency installation, environment configuration, and deployment commands.
    - **Version Control:** Commit history reflects changes specific to admin functionalities, ensuring traceability of modifications.

---

4. Compliance Requirements

- **Data Protection & Privacy:**
    - All components must comply with relevant data protection laws (e.g., GDPR, local data privacy regulations).
    - Sensitive data (e.g., user credentials, API keys) must be secured using industry-standard practices.
- **Security Audits:**
    - Regular security audits and code reviews should be conducted to ensure the system is secure against potential threats.
- **Operational Compliance:**
    - The system should comply with the organization's operational and IT policies, including backup and recovery procedures.
- **Regulatory Standards:**
    - Ensure that all dependencies and third-party integrations meet the latest security and performance standards.

---

5. Version Control History

5.0 GKCC Frontend

- **Key Commit Highlights:**

- Updates to critical files such as `StepOne.jsx`, `globals.css`, and various component files.
- Frequent updates for responsiveness and performance improvements.
- Commits from 2024-11-29 to 2025-02-08 detail structural changes, feature additions, and fixes.

6.0 GKCC Backend

- **Key Commit Highlights:**
  - Multiple updates to `app.js`, controllers, and models.
  - Introduction of middleware for error handling and authentication.
  - Commits reflect enhancements in API endpoints, database connectivity, and rate limiting from late 2024 through early 2025.

7.0 GKCC Admin Panel (Portal)

- **Key Commit Highlights:**
  - Updates focusing on creating and editing media albums, newsletters, and sponsor management.
  - Environment configuration changes and improvements in the deployment process.
  - Commit history indicates a progression from initial commit in December 2024 to significant feature additions in early 2025.

---

6. Appendices

Appendix A: References

- **Frontend Documentation:** Detailed README for GKCC Frontend including setup, folder structure, and API integration.
- **Backend Documentation:** Technical walkthrough covering project structure, API endpoints, and middleware implementations.
- **Admin Panel Documentation:** Setup and usage instructions for the GKCC Admin Panel with focus on CMS features.
- **Version Histories:** Full commit logs from GitHub for each component are available in the respective repositories.

8.0 Appendix B: Key Terms

Term	Definition
Next.js	A React framework used for building server-side rendered applications.
Node.js	A JavaScript runtime built on Chrome's V8 engine used for backend services.
Express	A web application framework for Node.js.
MongoDB	A NoSQL database used for scalable data storage.
JWT (JSON Web Token)	A compact, URL-safe means of representing claims to be transferred between two parties.
Mongoose	An ODM (Object Data Modeling) library for MongoDB and Node.js.
Axios	A promise-based HTTP client for making API requests.
TailwindCSS	A utility-first CSS framework for rapid UI development.
framer-motion/gsap	Libraries used for animations in the UI.

---

7. Disclaimer

The information contained in this document is for reference purposes only. While every effort has been made to ensure the accuracy and completeness of the functional requirements, the GKCC project team makes no warranties, either express or implied, regarding the completeness, accuracy, or reliability of the documentation. Any reliance on this document is strictly at the user's own risk.
