**Homepage Concept for RCCG Cornerstone Church**

**1. Background and Layout:**

Full-Width Transparent Video Banner:

* Position: This banner should be at the very top of the homepage, spanning the full width of the screen. The height should be around 60-70% of the viewport height to make a strong visual impact.
* Content: The banner will play a looping video of church services, community events, or serene nature scenes. The video should be overlaid with a semi-transparent dark gradient to ensure the text overlay is readable.
* Implementation: Use the `<video>` tag with the `autoplay`, `loop`, and `muted` attributes. Add CSS for positioning and the gradient overlay. Use React hooks to control the video play on different screen sizes.

Church Logo Placement:

* Position: The logo should be centered horizontally and placed slightly above the middle of the video banner, with an animation that makes it gently fade in when the page loads.
* Implementation: Use a CSS class for the logo that includes properties for transparency, fade-in animation (e.g., with `@keyframes`), and responsive scaling.

Section Breakers:

After the banner, use smooth scrollable dividers with subtle animations (e.g., parallax or fade-in) to transition into each section of the homepage.

2. Sections:

“I'm New":

* Purpose: To welcome newcomers and guide them through what the church offers.
* Design: A card or tile-based layout that highlights key information like service times, what to expect, and an invitation to a welcome session.
* Implementation: Use Material-UI’s Card component. Add hover effects with `react-spring` to make the cards slightly zoom in when hovered.

"Contact Us":

* Purpose: To provide an easy way for visitors to reach out to the church.
* Design: A contact form with fields for name, email, and message. It should also include a map showing the church’s location.
* Implementation: Use a form component with controlled inputs in React. Add a Google Maps embed using the `react-google-maps` library. Enhance the form with validation using a library like `Formik` and `Yup`.

"Connect with Us":

* Purpose: To highlight social media channels and community groups.
* Design: Use a grid layout for social media icons with hover effects and links to the church’s pages. Additionally, add a section for small group information, with cards that provide details and sign-up links.
* Implementation: Use a CSS grid or Material-UI’s Grid component. For icons, use `react-icons` or `@mui/icons-material` for consistency. Add links with animations using `react-spring`.

"Manage":

* Purpose: To give church admins access to their member profiles, tithing management, and event registrations and publicity.
* Design: This section should be more utilitarian, with quick access buttons for various management tasks. Each button should open a modal or direct to another page for the specific task.
* Implementation: Use Material-UI’s Button and Modal components. Implement routing with `react-router-dom` to handle page transitions.

3. Widgets:

Countdown Timer Widget:

* Purpose: Display a countdown to the next service or event.
* Implementation: Use the `react-countdown` package. Customize the display to match the church’s branding and add animations to make it visually appealing.

Testimonial Slider Widget:

* Purpose: Showcase positive testimonials from church members or visitors.
* Implementation: Use a carousel component like `react-slick` or `swiper.js`. Add a fade transition and make each slide have a background image related to the testimonial.

Donation/Offering Widget:

* Purpose: Provide an easy and secure way for users to give offerings or donations.
* Implementation: Integrate a third-party payment service (like Stripe) using their React components. Add a button in the "Manage" section that opens the donation modal.

Live Chat Widget:

* Purpose: Offer real-time support for visitors who have questions or need prayer.
* Implementation: Integrate a live chat service like Tawk.to or use a custom-built chat with `Socket.IO` and React. Position it as a floating button on the bottom-right corner of the screen.

**Testimony Management Page Concepts**

1. Overview and Layout:

Page Structure:

* Header: The page should have a header section that includes a title like " Testimonies" and action buttons (e.g., "Add New Testimony").
* Main Content: Below the header, display a list of testimonies in a card or table format, with options to view, edit, or delete each testimony.
* Sidebar (Optional): Include a sidebar with filters or categories to sort testimonies (e.g., by date, author, or topic).

Background and Styling:

* Use a clean, minimalistic background with light colors to keep the focus on the content. Incorporate the church’s branding with subtle accent colors.
* The layout should be responsive, adjusting smoothly between desktop and mobile views.

2. Sections and Components:

Testimony List:

Design: Each testimony should be presented as a card or table row, displaying key information like the author, date, and a snippet of the testimony.

Actions: Include buttons/icons for editing, deleting, or publishing/unpublishing the testimony. Also, provide a button for viewing the full testimony in a modal or on a separate page.

Implementation: Use Material-UI’s Card or Table components for structure. For actions, utilize `react-icons` for intuitive icons like edit, delete, or publish.

Search and Filter Bar:

* Design: A bar at the top of the testimony list that allows users to search for specific testimonies by keyword and filter by date, author, or status (e.g., published, unpublished).
* Implementation: Use a combination of controlled input fields and dropdowns for filtering. Implement searching and filtering logic in the state management (e.g., using React’s `useState` and `useEffect` or `Redux` for more complex state management).

Add/Edit Testimony Form:

* Design: A modal or separate page where users can add or edit a testimony. The form should include fields for the author’s name, date, testimony text, and an optional image or video upload.

Implementation: Use a form library like `Formik` for form handling and validation. Integrate `react-dropzone` or Material-UI’s file input for media uploads. Implement text editing with a rich text editor like `react-quill`.

Pagination:

* Design: For pages with a large number of testimonies, include pagination at the bottom of the list to navigate through multiple pages.
* Implementation: Use Material-UI’s Pagination component. Manage the current page state and update the displayed testimonies accordingly.

Moderation Tools:

Design: Include features for approving or rejecting testimonies before they go live. This could be a status toggle (e.g., pending, approved, rejected) or a separate moderation queue.

Implementation: Add status indicators in the testimony cards or table rows. Implement moderation logic using state management, and consider using a Redux store or a context provider for managing testimony statuses.

Analytics and Insights (Optional):

Design: A section that provides insights on testimony engagement, like the number of testimonies submitted, approved, and viewed.

Implementation: Use a data visualization library like `recharts` or `chart.js` to display graphs or charts. Fetch and manage data with React and a backend service.

3. User Experience Enhancements:

Confirmation Dialogs:

* Before deleting or unpublishing a testimony, show a confirmation dialog to prevent accidental actions.
* Implement this with Material-UI’s Dialog component or a custom modal using React.

Notifications and Alerts:

* Display success or error notifications when actions like saving, deleting, or updating testimonies are performed.
* Use a notification library like `react-toastify` to implement this feature.

Autosave Feature (Optional):

* For long testimonies, include an autosave feature that periodically saves progress to prevent data loss.
* Implement this using `useEffect` with a debounce function to trigger autosave at intervals.

Prayer Request Management Page Concepts

1. Scriptural Inspiration:

* Theme Verse: Consider using a verse like Philippians 4:6 ("Do not be anxious about anything, but in every situation, by prayer and petition, with thanksgiving, present your requests to God.") as a guiding theme. This verse can be displayed prominently at the top of the page to remind users of the power of prayer.
* Design Motif: Incorporate visual elements like subtle cross symbols, doves, or light rays to evoke a spiritual atmosphere. Use calm and serene colors like soft blues, whites, and light golds.

2. Overview and Layout:

Header:

* Title: “Manage Prayer Requests” with a subtitle that includes the theme verse.
* Action Buttons: Add buttons like “Add New Request,” “View All Requests,” and “Send Prayer Response” (for acknowledging that a prayer request has been seen or prayed over).

Main Content:

* Request List: Display a list of prayer requests in a card or table format, with options to view, pray, respond, or delete.
* Sidebar (Optional): Include a sidebar for filtering requests (e.g., by urgency, category, or date).

3.Sections and Components:

Prayer Request List:

* Design: Each prayer request can be presented as a card or table row, displaying the requester’s name (if not anonymous), the date of the request, the prayer topic, and a brief description.
* Actions: Include options for marking a request as “Prayed for,” responding with a message, editing, or deleting.
* Implementation: Use Material-UI’s Card or Table components. Implement state management for marking requests as prayed for or responded to, using React’s `useState` or a global state like Redux.

Search and Filter Bar:

* Design: Include a search bar to find specific requests by keywords, and filters to sort by urgency, request date, or categories like “Healing,” “Guidance,” “Thanksgiving,” etc.
* Implementation: Implement with controlled input fields and dropdowns. Update the display with filtered requests using React’s state management.

Add/Edit Prayer Request Form:

Design: A modal or separate page for adding or editing prayer requests. Include fields for the requester’s name, prayer topic, description, and an option for anonymity.

Implementation: Use a form handling library like `Formik` for validations. Include options for media attachments (e.g., images or voice notes) using `react-dropzone` or Material-UI’s file input components.

Prayer Response Section:

* Design: A dedicated area within each prayer request to respond with a prayer or message. Allow for sending an email or SMS notification to the requester (if contact details are provided).
* Implementation: Use a rich text editor like `react-quill` for the response. Integrate email/SMS sending capabilities with a third-party service like SendGrid or Twilio.

Urgency Indicators:

* Design: Add visual indicators (e.g., colored tags) to show the urgency of the prayer request (e.g., critical, high, medium, low).

Implementation: Use CSS classes for different urgency levels and manage these states in React.

4. User Experience Enhancements:

Progress Tracking:

* Design: A progress bar or timeline that shows the status of each prayer request (e.g., submitted, prayed for, responded to).
* Implementation: Use state management to update the progress dynamically. Display this progress visually on each card or row.

Scripture Recommendations:

* Design: Provide scripture suggestions based on the prayer request topic. For example, if the request is for healing, suggest verses like James 5:15 or Isaiah 53:5.
* Implementation: Create a mapping of topics to scripture verses, and display the relevant verses alongside the request. Use `useEffect` to update the verses dynamically based on the prayer topic.

Notification System:

* Design: Notify admins or prayer warriors when a new request is submitted or when a request is marked as urgent.
* Implementation: Integrate with a notification library like `react-toastify` for real-time alerts or send email notifications using a service like SendGrid.

Prayer Request Analytics (Optional):

* Design: Include a dashboard section that shows statistics like the number of requests received, prayed for, and responded to.
* Implementation: Use a chart library like `recharts` or `chart.js` to display data visualizations. Fetch data periodically from the backend or use a state management solution for real-time updates.

Spiritual Reflection (Optional):

* Design: A section where the user can reflect on answered prayers or testimonies linked to prayer requests. Allow users to mark a request as answered and provide a testimony.
* Implementation: Add a button or link on each prayer request card to mark as answered, which opens a form to submit a testimony. Store these in a separate section for future reference.

**Events Page Concepts**

1. Public Events Page:

This page is designed to engage visitors by showcasing upcoming church events, making it visually appealing and easy to navigate.

Overview and Layout:

Header:

A large, visually appealing banner that highlights the most significant upcoming event. This could include a countdown timer or a dynamic image/video background relevant to the event.

* A title like “Upcoming Events” with a scripture that inspires participation, such as Hebrews 10:25 (“Let us not give up meeting together...”).
* -Main Content:
* Event Listings: Display events in a card-based layout or a timeline format. Each card/timeline entry should show the event’s title, date, time, location, and a brief description.
* Call to Action: Each event should have a “Register” or “Learn More” button, leading to a detailed event page or registration form.

Design Elements:

* Visual Appeal: Use vibrant colors and imagery that align with the church’s branding. Consider including photos from past events to build excitement.
* Filter and Search: Provide a search bar and filters (e.g., by category, date, location) to help users find specific events. Implement this using dropdowns and checkboxes for category filtering.

Widgets:

* Calendar View: A calendar widget that highlights dates with upcoming events. Clicking on a date should bring up a list of events happening on that day.
* Implementation: Use a library like `react-calendar` or `FullCalendar` to implement this feature.
* Event Countdown: A countdown timer for the next major event, displayed prominently on the page.
* Implementation: Use `react-countdown` for the countdown timer and customize it with the church’s color scheme.

Event Details Modal:

* When a user clicks on an event, open a modal that provides full details, including images, videos, a map for the location, and a link to register or add the event to their calendar.
* Implementation: Use Material-UI’s Dialog component for the modal and include dynamic content loading based on the selected event.

2. Admin Events Management Page:

This page is designed for administrators to manage events with advanced features for adding, editing, and organizing events.

Overview and Layout:

Header:

* A functional header with buttons like “Add New Event,” “Manage Categories,” and “View All Events.”

Main Content:

* Event Dashboard: A table or grid layout that lists all events with options to edit, delete, or duplicate events. Include columns for event name, date, status (e.g., upcoming, past, draft), and actions.
* Event Status Indicators: Use color-coded tags or badges to indicate the status of each event (e.g., live, draft, canceled).

Event Management Features:

Add/Edit Event Form:

* A comprehensive form where admins can input all details about an event, including title, description, date, time, location, category, and whether the event is public or private.
* Include fields for uploading event images, videos, or other media.
* Implementation:\*\* Use `Formik` for form handling, with a rich text editor like `react-quill` for the event description.

Category Management:\*\*

* A section to manage event categories, allowing admins to add, edit, or delete categories that help organize events.
* Implementation:\*\* Use a modal or a separate page for category management, with CRUD functionality.

Event Analytics (Optional):

* Provide analytics on event participation, including registration numbers, attendance tracking, and user engagement metrics.
* Implementation:\*\* Use data visualization tools like `recharts` or `chart.js` to present these insights.

Admin-Only Features:

* Draft and Approval Workflow:\*\* Allow admins to save events as drafts and include an approval process where certain events must be approved by a higher-level admin before going live.
* Implementation:\*\* Manage the event status with a state machine or use a workflow library like `redux-saga`.

Notification System:

* Notify admins of upcoming events that require attention or when an event is approaching its date.

Implementation: Use a notification library like `react-toastify` for real-time alerts.

Permissions and Access Control:

* Restrict access to the admin events page using role-based access control (RBAC). Ensure that only users with admin privileges can view and manage events.
* Implementation: Implement RBAC using React Router and conditional rendering based on the user’s role.

Evangelism Management Page Concepts

1. Purpose and Overview:

This internal page is designed for church leaders, evangelism teams, and volunteers to coordinate and track evangelism efforts. It should provide tools for planning, executing, and monitoring evangelism activities. This internal evangelism management page will provide a robust tool for organizing, tracking, and enhancing evangelism efforts within the church, ensuring that every step from planning to follow-up is well-coordinated and effective.

Page Structure:

Header:

* A title like “Manage Evangelism” with an inspiring verse, such as Matthew 28:19 (“Go therefore and make disciples of all nations…”).
* Action buttons for adding new evangelism initiatives, viewing all efforts, and generating reports.

Main Content:

* Dashboard Overview: Display key metrics such as the number of people reached, decisions made for Christ, follow-up status, and upcoming events.
* Evangelism Activity Feed: A timeline or list of recent evangelism activities, showing brief descriptions, dates, and responsible team members.
* Calendar: A calendar view to schedule and keep track of evangelism events, visits, and follow-up activities.

2. Sections and Components:

Evangelism Campaign Management:

Campaign List:

* Design: Each campaign is represented as a card or table row, displaying the campaign name, dates, goals, and status (e.g., ongoing, completed, upcoming).
* Actions: Options to edit, duplicate, delete, or view detailed reports of each campaign.
* Implementation: Use Material-UI’s Card or Table components for the layout. Manage campaign status and actions with React’s state management.

Add/Edit Campaign Form:

* Design: A form to input campaign details, including title, description, target audience, location, duration, goals, and team members.
* Implementation: Use a form library like `Formik` for validation and field management. Include options to assign roles and responsibilities within the campaign.

Volunteer Coordination:

Volunteer List:

* Design: A list or table showing all volunteers involved in evangelism, with contact details, assigned roles, and participation history.
* Actions: Assign volunteers to specific campaigns, track their participation, and send notifications or reminders.
* Implementation: Manage volunteer data using state or Redux, with forms for assigning roles and updating participation.

Role Assignment:

* Design: Assign specific roles to volunteers, such as “Team Leader,” “Speaker,” “Prayer Support,” etc. Roles can be displayed on the campaign page and managed through a drag-and-drop interface.
* Implementation: Use a library like `react-beautiful-dnd` for drag-and-drop functionality, and manage assignments through state updates.

Follow-Up Management:

* Design: Track the follow-up process for those who made decisions for Christ during evangelism efforts. This section could include lists of names, contact information, follow-up dates, and notes.
* Implementation: Create a form to log follow-up interactions and a dashboard to monitor progress. Use state management to handle data updates and persistence.

Resource Library:

* Design: A section where team members can access evangelism resources like tracts, scripts, training videos, and spiritual growth materials. This library should be easily searchable and categorized.
* Implementation: Store resources in a database or file system and use a search/filter system to help users find what they need. Integrate file uploads with a component like `react-dropzone`.

3. Analytics and Reporting:

Campaign Analytics:

* Design: Provide detailed analytics on each evangelism campaign, including the number of people reached, decisions made, follow-ups completed, and overall campaign effectiveness.
* Implementation: Use data visualization libraries like `recharts` or `chart.js` to present these metrics. Pull data from the backend or state management for real-time updates.

Volunteer Performance Reports:

* Design: Generate reports on volunteer participation, showing their involvement across different campaigns and their impact on evangelism outcomes.
* Implementation: Create customizable reports that can be exported to PDF or CSV for further analysis.

4. Permissions and Access Control:

Restricted Access:

Ensure that this page is only accessible to authorized users, such as church leaders, evangelism team members, and select volunteers.

* Implementation: Use role-based access control (RBAC) to restrict access, implemented with React Router and conditionally rendering components based on user roles.

5. Notifications and Alerts:

* Design: Implement a notification system that alerts team members about upcoming events, pending follow-ups, or new volunteer assignments.
* Implementation: Use a notification library like `react-toastify` for real-time alerts, and manage notifications through state or Redux.