Rudhirsetu Website

A modern web application for Rudhirsetu Seva Sanstha built with React, TypeScript, and Tailwind CSS.

For NGO Workers: Content Management Guide

Accessing Sanity Studio

1. Go to https://rudhirsetu.sanity.studio/
2. Log in with your provided credentials
3. You'll see the main dashboard with different content types

Managing Different Content Types

1. Events

To add or edit events:

1. Click on "Events" in the sidebar
2. Click "Create new Event" to add a new event
3. Fill in the following details:
    - Title: Name of the event
    - Date: Select the event date and time
    - Location: Where the event will take place
    - Expected Participants: Optional estimate of attendees
    - Is Upcoming: Check this for future events
    - Description: Additional details about the event
4. Click "Publish" to make the event live on the website

2. Gallery Images

To manage gallery images:

1. Click on "Gallery Images" in the sidebar
2. Click "Create new Gallery Image" to add new images
3. Fill in:
    - Title: Optional name for the image

- Description: Optional details about the image
- Category: Select from blood-donation, eye-care, cancer-awareness, or thalassemia-support
- Is Featured: Check this to show the image in the homepage carousel
- Image: Upload your image and add alternative text for accessibility

4. Click "Publish" to make the image visible on the website

3. Donation Settings

To update donation information:

1. Click on "Donation Settings" in the sidebar
2. You can edit:
   - UPI ID
   - Account Name
   - Account Number
   - IFSC Code
   - Bank and Branch details
   - QR Code Image
3. Remember to click "Publish" after making changes

4. Contact Settings

To update contact information:

1. Click on "Contact Settings" in the sidebar
2. You can modify:
   - Address
   - Phone Number
   - Email
   - Google Maps URL (get this from Google Maps by clicking "Share" and copying the embed URL)
3. Click "Publish" to update the website

5. Social Media Links

To manage social media links:

1. Click on "Social Media Settings" in the sidebar
2. You can update:

- LinkedIn URL: Your organization's LinkedIn profile link
- Facebook URL: Your Facebook page link
- Instagram URL: Your Instagram profile link
- YouTube URL: Your YouTube channel link
- Description: A brief message about following on social media

3. Click "Publish" to update the social media page
4. All links will automatically update on the website

Best Practices

1. **Images**:
   - Use clear, high-quality images
   - Keep file sizes under 5-10MB
   - Add descriptive alternative text
   - Complusory appropriate categories for gallery images
2. **Text Content**:
   - Keep titles concise and clear
   - Use proper capitalization
   - Check spelling and grammar
   - Include all relevant details in descriptions
3. **Events**:
   - Keep event information up-to-date
   - Mark past events as not upcoming
   - Include complete venue details
   - Update participant numbers after events
4. **Regular Maintenance**:
   - Review and update contact information monthly
   - Remove outdated gallery images
   - Verify donation details are current

Need Help?

If you encounter any issues or need assistance:

1. Contact the website administrator
2. Check the Sanity documentation at https://www.sanity.io/docs
3. Take screenshots of any errors you encounter

Technical Documentation

Tech Stack

- **Frontend**: React with TypeScript
- **Styling**: Tailwind CSS
- **CMS**: Sanity
- **Routing**: React Router
- **Icons**: Lucide React
- **Date Handling**: date-fns

Content Models

1. Events

Events are used in the Impact page to showcase upcoming and past events.

```
interface Event {
  id: number;
  documentId: string;
  title: string;
  date: string;
  location: string;
  expectedParticipants?: string;
  isUpcoming: boolean;
  desc?: string;
  createdAt: string;
  updatedAt: string;
  publishedAt: string;
}
```

API Endpoints:

- Get all events: `GET /api/events?sort=date:desc`
- Get upcoming events: `GET /api/events?filters[isUpcoming][$eq]=true`
- Get past events: `GET /api/events?filters[isUpcoming][$eq]=false`

2. Gallery Images

Used in the Gallery page to display images with categories and featured images.

```
interface GalleryImage {
  id: number;
  documentId: string;
  Title?: string;
  description?: string;
  category: string;
  IsFeatured: boolean;
  image: Array<{
    id: number;
    url: string;
    alternativeText: string | null;
    formats: {
      thumbnail?: ImageFormat;
      small?: ImageFormat;
    };
  }>;
}

interface ImageFormat {
  url: string;
  width: number;
  height: number;
  size: number;
}
```

API Endpoints:

- Get all images: `GET /api/gallery-images?populate=*`
- Get images by category: `GET /api/gallery-images?populate=*&filters[category][$eq]={category}`

3. Donation Settings

Single type for managing donation-related information.

```
interface DonationSettings {
  id: number;
  documentId: string;
  upiId: string;
  accountName: string;
  accountNumber: string;
  ifscCode: string;
  bankAndBranch: string;
  qrCodeImage: {
    url: string;
    alternativeText: string | null;
    formats: {
      thumbnail: ImageFormat;
      small: ImageFormat;
      medium: ImageFormat;
    };
  };
}
```

API Endpoint:

- Get donation settings: `GET /api/donation-setting?populate=*`

4. Contact Settings

Single type for managing contact information.

```
interface ContactSettings {
  id: number;
  documentId: string;
  address: string;
  phone: string;
  email: string;
  url: string; // Google Maps iframe HTML
}
```

API Endpoint:

- Get contact settings: `GET /api/contact-setting?populate=*`

Migration Notes for Sanity

Schema Transformations

1. **Events Collection**:

```
// Sanity schema
export default {
  name: 'event',
  title: 'Events',
  type: 'document',
  fields: [
    {
      name: 'title',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'date',
      type: 'datetime',
      validation: Rule => Rule.required()
    },
    {
      name: 'location',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'expectedParticipants',
      type: 'string'
    },
    {
      name: 'isUpcoming',
      type: 'boolean',
      initialValue: true
    },
    {
      name: 'desc',
      type: 'text'
    }
  ]
}
```

2. **Gallery Images Collection**:

```
// Sanity schema
export default {
  name: 'galleryImage',
  title: 'Gallery Images',
  type: 'document',
```

```
  fields: [
    {
      name: 'title',
      type: 'string'
    },
    {
      name: 'description',
      type: 'text'
    },
    {
      name: 'category',
      type: 'string',
      options: {
        list: [
          'blood-donation',
          'eye-care',
          'cancer-awareness',
          'thalassemia-support'
        ]
      }
    },
    {
      name: 'isFeatured',
      type: 'boolean',
      initialValue: false
    },
    {
      name: 'image',
      type: 'image',
      options: {
        hotspot: true
      },
      fields: [
        {
          name: 'alt',
          type: 'string',
          title: 'Alternative Text'
        }
      ]
    }
  ]
}
```

3. **Donation Settings Singleton**:

```
// Sanity schema
export default {
```

```
  name: 'donationSettings',
  title: 'Donation Settings',
  type: 'document',
  __experimental_actions: ['update', 'publish'], // Prevents creating multiple instances
  fields: [
    {
      name: 'upiId',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'accountName',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'accountNumber',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'ifscCode',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'bankAndBranch',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'qrCodeImage',
      type: 'image',
      options: {
        hotspot: true
      },
      fields: [
        {
          name: 'alt',
          type: 'string',
          title: 'Alternative Text'
        }
      ]
    }
  ]
}
```

4. **Contact Settings Singleton**:

```
// Sanity schema
export default {
  name: 'contactSettings',
  title: 'Contact Settings',
  type: 'document',
  __experimental_actions: ['update', 'publish'],
  fields: [
    {
      name: 'address',
      type: 'text',
      validation: Rule => Rule.required()
    },
    {
      name: 'phone',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'email',
      type: 'string',
      validation: Rule => Rule.required().email()
    },
    {
      name: 'googleMapsUrl',
      type: 'url',
      validation: Rule => Rule.required()
    }
  ]
}
```

Key Differences between Strapi and Sanity

1. **Image Handling**:
   - Strapi: Stores images on local filesystem or external providers, returns URLs
   - Sanity: Uses Sanity Image Pipeline, requires GROQ queries
2. **API Structure**:
   - Strapi: REST API with filters and populate
   - Sanity: GROQ queries for flexible data fetching
3. **Authentication**:

- Strapi: JWT-based auth
- Sanity: Token-based auth with CDN

4. **Deployment**:
   - Strapi: Requires separate backend hosting
   - Sanity: Hosted service, only frontend deployment needed

Environment Variables

```
# Current Strapi Setup
VITE_STRAPI_API_URL=http://localhost:1337
VITE_STRAPI_URL=http://localhost:1337

# For Sanity Migration
VITE_SANITY_PROJECT_ID=your-project-id
VITE_SANITY_DATASET=production
VITE_SANITY_API_VERSION=2023-05-03
```

Pagination

All list endpoints support pagination with the following parameters:

- `page` : Current page number
- `pageSize` : Number of items per page (default: 6 for events, 12 for gallery)

Response includes metadata:

```
interface Pagination {
  page: number;
  pageSize: number;
  pageCount: number;
  total: number;
}
```

Folder structure:

1. src/components - For reusable UI components
2. src/pages - For different pages/routes
3. src/assets - For images and other static assets

4. src/styles - For global styles
5. src/layouts - For layout components
6. src/utils - For utility functions
7. src/types - For TypeScript type definitions