Village Management Project

1. Frontend & Backend Structure

Frontend

• Technologies Used:

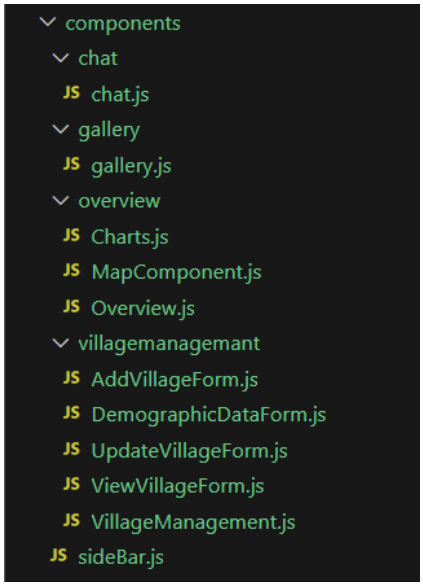
- The frontend is entirely built using React.

- Additional libraries are used to enhance display and interaction, such as react-leaflet and leaflet for embedding interactive maps.

• Component Structure:

- The interface is divided into independent and reusable components.

- Each component serves a specific function and has its own design as in the figure .



- Start with Login and Sign-Up Page:Enables users to log in or create a new account.
- Main Sections of the Project:
  -Overview Section: Contains the map component to display the locations of villages. Uses the **react-leaflet** library to include interactive maps, Places markers to identify village locations.
  -Displays charts and statistics about the villages, Utilizes chart libraries like **Chart.js or Recharts** for attractive information presentation.

Backend

• Technologies Used:

- **Express** with **Node.js** to create the server.

- **Apollo Server** to support **GraphQL** usage.

• Application Layers:

1. Routes:

   - Defines endpoints to route requests.

2. Middleware:

   - Handles authentication using JWT.

3. Controllers:

   - Processes requests and returns appropriate responses.

4. Services:

   - Core business logic such as database interactions.

• Security:

- Uses **JWT** for authentication and authorization.

- Passwords are encrypted using backage **bcrypt**.

- Manages sessions and identifies the current user through the is_logged_in property in the database.

2. Database

• Database Name: village_management

• Database Type: **MySQL**

Main Tables:

1. Gallery Table:

- Purpose: Stores images related to the villages.
- Fields:
  - id: Unique identifier.
  - imgBase64: Base64 encoded string of the image.
  - imgText: Description of the image.
  - createdAt: Creation date.

2. Messages Table:

- Purpose: Stores messages between users.
- Fields:
  - id: Unique identifier.
  - sender: Sender's name.
  - recipient: Recipient's name.
  - text: Message text.
  - timestamp: Time of sending.

3. Users Table:

- Purpose: Manages user data.
- Fields:
  - id: Unique identifier.
  - username: Unique username.
  - password: Encrypted password.
  - full_name: Full name.
  - role: User role (admin or user).
  - is_logged_in: Login status.
  - profile_image: Link to profile image.
  - email: Unique email address.

4. Villages Table:

- Purpose: Stores village data.

- Fields:

  - id: Unique identifier.

  - name: Village name.

  - region: Region.

  - land_area: Land area.

  - latitude and longitude: Geographic coordinates.

  - image: Image link.

  - tags: Keywords.

  - population: Population count.

  - population_distribution: Population distribution (JSON).

  - gender_ratios: Male and female ratios (JSON).

  - population_growth_rate: Population growth rate.

3. Chat System

• System Description:

- A system for exchanging messages between users.

- Relies on the messages table to store messages.

- Supports sending and receiving messages with timestamps and user identification.

- The chat system uses sockets, allowing users to send messages to any chosen admin.

- Messages are stored in the database for preservation, and admins can view users who sent messages upon login and can respond to them.

4.User Gallery:

• Each user will have a personal photo gallery where they can add photos and locations they have visited.

Admin Photo Addition:

• Admins can add photos, which will be visible to all users in the public gallery.

Image Search Feature:

• There will be a search feature within the gallery based on the description of the image to facilitate finding added photos.

5. GraphQL

• Description:

- Integrated GraphQL to enhance APIs.

- Utilizes Apollo Server to manage data queries.

- Allows users to query data flexibly according to needs without fetching unnecessary data.

Additional Notes:

• **Base64:**

- Used for compressing images and uploading them to the database for storage.

• **JWT:**

- Manages authentication and authorization processes.

- Tracks the current user for displaying and managing their personal data.

• **bcrypt:**

- Encrypts passwords to ensure security.

- Validates password matches during login.

• **Express & SQL Integration:**

- Creates a server using Express.

- Integrates with MySQL for database operations.

• **Apollo Server:**

- Used to support GraphQL queries, making data access more efficient and faster.

login and can respond to them.