FusionMail Documentation

1. Project Overview

FusionMail is a modern, Gmail-inspired webmail client built with React, TypeScript, and a real (external) backend service. It supports advanced email management, real-time updates, and a rich set of productivity features for both received and sent mail. The UI and UX are designed to closely mirror Gmail, with enhancements for real-time feedback and customization.

2. Key Features

- **Conversation View:** Threaded email conversations, with per-message actions.
- **Star, Archive, Spam, Trash, Mute, Snooze, Add to Tasks:** Gmail-like actions for both individual messages and entire threads.
- **Labels:** Assign, remove, and create labels with color coding. Real-time label updates for both received and sent mail.
- **Rich Compose Modal:**
    - Font selection (with Google Fonts), color, alignment, and formatting.
    - Attachments, emoji, and signature support.
    - Save as draft, schedule send, and send mail with label assignment.
- **Real-Time UI Updates:**
    - Optimistic updates and React Query cache invalidation for instant feedback.
    - All actions (star, archive, etc.) update the UI immediately across Inbox, Sent, Drafts, and Label views.
- **Show Original Message:**
    - View the raw source of any message (received or sent), with header parsing and RFC 2047 decoding.
- **User Authentication:** JWT-based login, with user email extracted from token.
- **Mobile Responsive:** Fully responsive design for desktop and mobile.
- **Accessibility:** Keyboard navigation and ARIA support for major actions.

3. Supported Actions

- **Inbox Management:**
    - Star/unstar, archive/unarchive, mark as spam/not spam, move to trash/restore, mute/unmute, snooze/unsnooze, add/remove from tasks.
    - Bulk actions via toolbar.
- **Label Management:**
    - Assign/remove labels to any message (received or sent).
    - Create new labels with random hex color.
    - Manage labels from both the sidebar and message menus.
- **Compose & Drafts:**
    - Compose new mail, reply, reply-all, forward.
    - Save as draft, schedule send, attach files, insert signature, use rich formatting.
- **Conversation Actions:**
    - All actions (star, archive, etc.) can be applied to any message in a thread, and can affect the whole conversation as per Gmail logic.
- **Show Original:**
    - View and download the raw EML source of any message, with headers and body.
- **Search & Filter:**
    - Search by subject, sender, or label. Filter by status (inbox, sent, drafts, spam, etc.).

4. User Interaction Flows

- **Viewing a Conversation:**
    - Click any email in the list to open the conversation view.
    - Use the toolbar or per-message menus for actions (star, label, archive, etc.).
    - Click the three-dot menu on any message to access "Show original" and other actions.
- **Composing Mail:**
    - Click "Compose" to open the rich modal.
    - Select font, color, and formatting; attach files; assign labels.
    - Save as draft or send immediately.
- **Label Assignment:**
    - Open the label menu from the sidebar, toolbar, or message menu.
    - Assign or remove labels; create new labels with color.
    - Changes are reflected in real time across all views.
- **Show Original Message:**
    - Click "Show original" in any message menu to view the raw EML source.
    - If the backend API is unavailable, the frontend will use the locally cached rawEml for accurate display.
- **Bulk Actions:**

  - Select multiple emails in the list and use the toolbar for bulk star, archive, spam, etc.

5. Technical Highlights

- **Frontend:** React (TypeScript), TanStack Query (React Query), Tailwind CSS, custom hooks and contexts for font, color, translation, etc.
- **Backend:**
  - FusionMail uses a real/external backend service for all email, label, and user management APIs.
  - The backend is not part of this repository; the `server/` directory here is not used in production.
  - All API interactions in the frontend (`client/`) are directed to the external backend.
- **Shared Types:**
  - The `shared/` directory contains TypeScript types and schemas for data models (emails, labels, etc.) used by the frontend.
  - This ensures type safety and consistency with the backend API contracts.
- **Data Model:**
  - Emails have `emailUniqueId` (received) or `sendMail_Id` (sent), and are grouped by `threadId` for conversations.
  - Labels have `labelUniqueId`, name, and color.
- **API Endpoints:**
  - `/email/allmails`, `/mails/get-sendmail`, `/email/getEmailsByLabel`, `/label/getLabels`, `/label/createLabel`, `/email/updateEmail`, `/mails/conversation`, `/mails/getRawEmailFile`, etc.
- **Cache & Real-Time:**
  - Uses React Query for data fetching and cache management.
  - All actions trigger cache invalidation for real-time UI updates.
- **Raw Message Handling:**
  - When viewing "Show original", the frontend will use the `rawEml` from localStorage if the API fails, and will parse headers for display.
  - RFC 2047 decoding is applied for human-readable headers.

6. How Files Are Saved & Documentation Location

- **Source Code:**
  - Frontend: `client/`
  - Shared types: `shared/` (for TypeScript models used by the frontend)
  - The `server/` directory is not used for the deployed application.
- **Documentation:**
  - This `README.md` is the canonical documentation for the project.
  - For PDF export, use any Markdown-to-PDF tool (e.g., VSCode extension, `pandoc`, or online converter).
  - All new features and changes should be documented here.

For further details, see the codebase and inline comments. For developer onboarding, start with this README and explore the `client/src/components/gmail/` directory for UI logic, and the `shared/` directory for data models and types.