ClarksvilleThreads

CommonThread Volunteer Platform

Codebase Overview & FBLA Rubric Analysis

FBLA Coding & Programming 2025-2026

"Byte-Sized Business Boost"

February 2026

1. Project Overview

CommonThread is a full-stack volunteer management platform that connects volunteers with local businesses and nonprofits. It serves as a centralized marketplace where volunteers discover opportunities matching their skills, apply, track hours, and earn recognition -- while business owners post opportunities, review applications, and manage their volunteer roster.

Core User Flows

- Volunteers: Browse businesses, filter opportunities by skills/urgency, apply, track hours on a calendar, bookmark favorites, leave reviews.
- Business Owners: Register a business, post volunteer opportunities with perks, review/approve applications, manage a volunteer roster.
- AI Chatbot: Gemini-powered Q&A helps users navigate the platform.

FBLA Topic Alignment: Byte-Sized Business Boost

The prompt requires a tool that helps users discover and support small, local businesses. CommonThread addresses this by letting users:

- Sort businesses by category (food, retail, services, nonprofit)
- Leave reviews and star ratings on businesses
- Sort businesses by rating, review count, or name
- Bookmark/favorite businesses for later
- Display special deals via "Volunteer Perks" on each opportunity

2. Tech Stack

Layer	Technology
Frontend	React 18 + Vite 6 (JSX)
Routing	React Router DOM 6
UI Components	Radix UI (shadcn/ui) + Tailwind CSS
Data Fetching	TanStack React Query 5
Forms	React Hook Form + Zod validation
Animation	Framer Motion
Maps	React Leaflet
Backend	Node.js 22 + Express (Cloud Functions)
Database	Google Firestore (NoSQL)
Auth	Firebase Auth (Google OAuth 2.0)
File Storage	Google Cloud Storage (signed URLs)
AI / Chat	Vertex AI - Gemini 1.5 Flash
Email	SendGrid (transactional)
Secrets	Google Cloud Secret Manager
Hosting	Firebase Hosting
CI/CD	GitHub Actions (auto deploy + PR previews)

The frontend is a React single-page application built with Vite, styled with Tailwind CSS, and using 20+ accessible Radix UI primitives. The backend is an Express REST API deployed as a Firebase Cloud Function with JWT-based auth. Data lives in Firestore with security rules enforcing user-scoped access. File uploads use signed URLs to Google Cloud Storage. The AI chatbot uses Vertex AI's Gemini 1.5 Flash model for interactive Q&A.

3. Main Entry Points

Frontend

- index.html -> src/main.jsx -> src/App.jsx (React root with AuthProvider, QueryClient, Router)
- src/Layout.jsx wraps all pages with desktop sidebar + mobile hamburger navigation

Page Routes (src/pages.config.js)

Route	Component	Purpose
/	Home.jsx	Landing page with hero, features, stats
/explore	Explore.jsx	Browse businesses, smart recommendations
/opportunities	Opportunities.jsx	Filter volunteer opportunities
/calendar	Calendar.jsx	Calendar view of committed dates
/favorites	Favorites.jsx	Bookmarked businesses
/profile	Profile.jsx	Edit skills, interests, track hours
/business-dashboard	BusinessDashboard.jsx	Owner: manage opps & apps
/business-detail	BusinessDetail.jsx	Single business view + apply
/business-signup	BusinessSignup.jsx	2-step business registration

Backend API (functions/)

Express app exported as Firebase Cloud Function. Base URL: /api

Route File	Key Endpoints	Purpose
auth.js	GET/PUT /auth/me, POST /register	User profile & registration
businesses.js	CRUD + filter /businesses	Business management
opportunities.js	CRUD + filter /opportunities	Opportunity listings
commitments.js	CRUD + filter /commitments	Volunteer applications
notifications.js	CRUD /notifications	In-app notifications
favorites.js	CRUD /favorites	Bookmarked businesses
reviews.js	CRUD /reviews	Ratings & reviews
chat.js	POST /chat/invoke, /conversation	AI chatbot (Gemini)
uploads.js	POST /uploads/signed-url	File storage

4. File Structure

```
ClarksvilleThreads/
|-- src/                             FRONTEND
|   |-- main.jsx                     React entry point
|   |-- App.jsx                      Root: auth + routing + providers
|   |-- Layout.jsx                   Nav shell (sidebar + mobile menu)
|   |-- pages.config.js              Route definitions
|   |-- api/gcpClient.js             REST client with auth token injection
|   |-- lib/
|   |   |-- FirebaseAuthContext.jsx    Auth state: login/logout, tokens
|   |   |-- firebase-config.js       Firebase SDK initialization
|   |   |-- query-client.js          React Query config
|   |   |-- PageNotFound.jsx         404 page
|   |-- pages/                       9 page components (see routes above)
|   |-- components/
|   |   |-- ui/                      20+ shadcn/Radix UI primitives
|   |   |-- business/                OpportunityManager, ReviewSection, etc.
|   |   |-- explore/BusinessCard.jsx  Business listing card
|   |   |-- NotificationBadge.jsx     Bell icon with unread count
|   |   |-- QASection.jsx            AI chatbot widget
|
|-- functions/                       BACKEND (Cloud Functions)
|   |-- index.js                     Express app + route mounting
|   |-- src/routes/                  9 route files (REST endpoints)
|   |-- src/services/                7 service files (business logic)
|   |   |-- userService.js           User CRUD
|   |   |-- businessService.js       Business CRUD + rating calc
|   |   |-- opportunityService.js     Opportunity CRUD + slot tracking
|   |   |-- commitmentService.js      Application CRUD
|   |   |-- notificationService.js    Notification CRUD
|   |   |-- emailService.js          SendGrid emails
|   |   |-- llmService.js            Gemini AI invocation
|   |-- src/middleware/auth.js       JWT verification
|   |-- src/config/                  Firebase Admin + Vertex AI init
|
|-- firebase.json                    Hosting/functions/storage config
|-- firestore.rules                  DB security rules
|-- storage.rules                    File upload rules
|-- .github/workflows/              CI/CD (deploy + PR previews)
|-- tailwind.config.js               Tailwind theme
|-- vite.config.js                   Vite build config
```

5. How the Code Works (End-to-End)

Example: Volunteer Applies for an Opportunity

1. AUTH: User clicks Google Sign-In. FirebaseAuthContext runs signInWithPopup(). Firebase returns a JWT stored in React context.

2. BROWSE: User visits /explore. Explore.jsx calls gcpClient.entities.Business.list(). The API client attaches the JWT as a Bearer token and hits GET /businesses.

3. BACKEND: routes/businesses.js receives the request. optionalAuth middleware verifies the token via Firebase Admin SDK. businessService.listBusinesses() queries Firestore and returns JSON.

4. RECOMMEND: Explore.jsx scores businesses by matching user interests/skills to business categories and opportunity requirements. Top matches shown first.

5. APPLY: User clicks Apply on an opportunity, fills the form. Frontend calls entities.VolunteerCommitment.create() which POSTs to /commitments.

6. AUTO-ACCEPT: Backend checks age/hour requirements. If met, status = confirmed; otherwise pending. Increments slots_filled on the opportunity.

7. NOTIFY: Backend creates a Firestore notification doc and sends a SendGrid confirmation email to the volunteer.

8. UPDATE: React Query invalidates the cache. UI refreshes showing the new commitment on the Calendar page.

Data Flow Diagram

```
React Pages  -->  gcpClient.js  -->  Cloud Functions (Express)
     |                                      |
React Query          JWT Bearer         Firestore DB
(cache)              Auth Token         (NoSQL docs)
     |                                      |
Firebase Auth  <--  Auth Middleware  <--  Security Rules
```

6. FBLA Rubric Mapping (110 pts)

Code Quality (20 pts)

Language Selection (5 pts)

STRONG - React/JSX frontend, Node.js/Express backend

Industry-standard choices. React for component-based UI, Express for REST APIs, Firestore for scalable NoSQL. Vite for fast builds. Can explain selection using industry terminology.

Comments & Formatting (5 pts)

GAP - ~7% comment density, inconsistent across files

gcpClient.js has good JSDoc comments. Layout.jsx and most pages have zero comments. Naming conventions are consistent (camelCase). Formatting is clean. Recommend adding JSDoc to all major functions and components.

Modular & Readable (10 pts)

STRONG - Clean separation of concerns

Frontend: pages / components / api / lib layers. Backend: routes / services / middleware / config layers. Each service handles one domain. Components are small and focused. React Query manages server state cleanly.

User Experience (25 pts)

UX Design, Journey & Accessibility (10 pts)

GOOD - Radix UI provides accessibility foundation

Responsive design (mobile + desktop). Radix UI provides keyboard navigation and ARIA attributes. Custom accessibility labels are sparse. No skip-nav or explicit a11y audit. Recommend adding aria-labels to business-specific components.

Intuitive UI / Instructions (5 pts)

STRONG - Clear navigation, well-labeled forms

Sidebar navigation with icons and labels. Mobile hamburger menu. Form fields have labels and placeholders. Error states shown via toast notifications.

Navigation + Intelligent Feature (5 pts)

STRONG - AI chatbot + smart recommendations

QASection.jsx provides Gemini-powered interactive Q&A. Explore.jsx has a recommendation engine matching user skills/interests to opportunities. Both qualify as intelligent features.

Input Validation (5 pts)

GAP - Basic HTML5 validation only

Required field checks exist but no semantic validation. Zod is installed but unused. No server-side schema validation. No specific error messages for edge cases. Recommend implementing Zod schemas on both frontend and backend.

Functionality (25 pts)

Addresses All Parts of Prompt (10 pts)

MOSTLY MET - 5 of 6 topic requirements covered

Category sorting: YES. Reviews/ratings: YES. Sort by rating: YES. Favorites/bookmarks: YES. Special deals (volunteer perks): YES. Bot verification: MISSING. Recommend adding reCAPTCHA or email verification.

Presentable Report / Data Analysis (10 pts)

MISSING - No export or analytics features

Dashboard shows basic stat cards (opportunity count, applications, volunteers). No PDF/CSV export despite jsPDF and html2canvas being in package.json. No charts despite recharts being installed. Recommend implementing CSV export for volunteer rosters and a PDF report for business dashboards.

Data Storage (5 pts)

STRONG - 8 Firestore collections, proper variable scope

Uses arrays for skills, interests, and notifications. Firestore collections: users, businesses, volunteer_opportunities, volunteer_commitments, notifications, favorites, reviews, monthly_availability. Variable scope is logical throughout services and components.

Presentation Delivery (30 pts)

These 30 points are earned during the live presentation and cannot be assessed from code alone. Categories: well-organized statements (10 pts), confidence/body language/eye contact/voice (10 pts), answering questions effectively (10 pts).

Presentation Protocols (10 pts)

10 points for following guidelines (max 3 devices, topic alignment, no QR scanning, etc.). Ensure compliance during your presentation.

7. Estimated Score & Gap Analysis

Rubric Item	Max	Est.	Notes
Language Selection	5	5	React + Node.js, strong industry choice
Comments & Formatting	5	3	Clean formatting, sparse comments
Modular & Readable	10	9	Excellent separation of concerns
UX Design & Accessibility	10	7	Good UX, basic a11y via Radix
Intuitive UI	5	4	Clear nav, labeled forms
Navigation + Intelligent	5	5	AI chatbot + recommendations
Input Validation	5	2	Basic only, Zod unused
Addresses Prompt	10	7	5/6 features, missing bot check
Reports / Data Analysis	10	2	Basic stats only, no exports
Data Storage	5	5	8 collections, arrays, proper scope
Presentation Delivery	30	?	Depends on live presentation
Protocols	10	10	Follow guidelines at competition

Estimated Technical Score: 59 / 70 (before presentation)

With the recommended fixes below, potential score: 68-70 / 70.

8. Priority Recommendations

CRITICAL (Highest Rubric Impact)

- Add bot verification: Integrate reCAPTCHA v3 on business signup and review forms. This is an explicit prompt requirement worth significant points.
- Implement report generation: Use the already-installed jsPDF + html2canvas to export business dashboards and volunteer hour summaries as PDFs. Add CSV export for volunteer rosters.
- Activate Zod validation: Zod is already in package.json. Add schemas to all backend routes and frontend forms for syntactic + semantic validation with helpful error messages.

HIGH (Moderate Rubric Impact)

- Increase code comments: Add JSDoc comments to all major functions, components, and services. Target 15%+ comment density. Focus on explaining WHY, not WHAT.
- Enhance README.md: Add project overview, architecture diagram, feature list, setup instructions, and third-party attribution (Radix UI, shadcn/ui, Firebase, etc.).
- Add accessibility labels: Add custom aria-label and aria-describedby to business cards, opportunity listings, and dashboard components.

MEDIUM (Polish)

- Add charts to dashboard: Use the already-installed recharts library to visualize volunteer hours, application trends, or rating distributions.
- Add email verification step: Require email confirmation before allowing reviews to prevent spam.

9. Reference: California State Champions Presentation

The Westlake Tennis Academy team (Spencer & Jay) won the California state competition with a tennis lesson scheduling platform. Key takeaways from their presentation style:

Presentation Structure They Used

- Page-by-page walkthrough: Homepage -> Mission -> Learn More -> Join Lesson -> Account -> Instructors -> Volunteer -> Admin
- Highlighted responsive/mobile design and accessibility (color contrast, collapsing nav)
- Showed live user flows: creating accounts, joining lessons, submitting applications
- Demonstrated admin flow: reviewing applications, editing lessons, sending emails
- Showed backend code organization (models, controllers, routers)
- Demonstrated automated features (reminder emails via job scheduler)
- Emphasized: 'All designs were original, no templates, designed in Figma then coded in React'

How CommonThread Compares

- Your stack is MORE sophisticated: Firebase + Vertex AI + SendGrid + Firestore security rules vs. their React + Node + MongoDB
- You have an AI chatbot (Gemini) which they lacked -- this is a strong differentiator
- Your smart recommendation engine matches their 'no templates' originality claim
- Consider structuring your presentation similarly: page walkthrough, live demo, code tour, then unique features

Generated February 2026 | ClarksvilleThreads FBLA Analysis