MULTILINGUAL FORUM PLATFORM

University project independently audited, completed, and security-hardened

STATUS
COMPLETED
Re-engineered build

Originally developed by me as a university full-stack forum project: authentication, profiles, forum creation, messaging, and MongoDB persistence formed the academic foundation. After submission, I independently audited and re-engineered the codebase - completing viewer-specific translation and password recovery, hardening sessions and browser rendering, and consolidating the backend into a cleaner, maintainable structure.

BACKEND	DATA	INTEGRATIONS	FRONTEND
Node.js + Express	**MongoDB + Mongoose**	**Google Translate + SMTP**	**Vanilla JS + HTML/CSS**

INDEPENDENT RE-ENGINEERING

VIEWER-SPECIFIC TRANSLATION

- Restored the university project's intended Google Cloud Translation API v2 capability and centralized it in a dedicated service layer.
- Corrected the translation target from the message author to the authenticated viewer's preferredLanguage.
- Kept original message content canonical; display text is translated at read time and safely falls back to the original if translation is unavailable.

COMPLETE PASSWORD RECOVERY

- Finished the incomplete recovery path end to end: forgot-password request, reset email, dedicated reset page, and backend token-consumption endpoint.
- Generated cryptographically random one-hour reset tokens, stored only SHA-256 hashes in MongoDB, and delivered reset links through Nodemailer/SMTP.
- Cleared tokens after successful use and invalidated older sessions after a password reset.

SECURITY HARDENING

- Replaced the original localStorage JWT pattern with an HttpOnly, SameSite cookie while retaining Bearer-token support for API clients.
- Removed user-controlled innerHTML rendering and switched to safe DOM text insertion to close the stored-XSS path.
- Added Content Security Policy/security headers, JSON body limits, stronger password handling, generic auth/reset responses, and explicit 404 behavior.

ARCHITECTURE + CLEANUP

- Consolidated authentication, forum, message, translation, and email responsibilities into clear middleware, controller, service, model, and route layers.
- Removed duplicate routes, obsolete translation paths, unused direct dependencies, hard-coded API origins, redundant message fields, and the empty 404 implementation.
- Added .env.example/.gitignore hygiene plus an optional guarded migration for legacy redundant fields without forcing destructive startup changes.

HOW THE SOLUTION FITS TOGETHER

Browser — Safe DOM UI > **Express** — Routes + middleware > **Controllers** — Business logic > **MongoDB / Services** — Data + Translate + SMTP

ENGINEERING VALUE

What the revision demonstrates beyond the original academic prototype

01	02	03
Full lifecycle ownership	**Security-conscious redesign**	**Maintainable handoff**
I built the original academic forum, then returned to audit and improve my own implementation - moving from feature delivery to correctness, security, and maintainability.	Authentication, account recovery, browser rendering, configuration, and error behavior were revised together rather than patched as isolated symptoms.	The final build has a reproducible lockfile, environment template, documented setup, explicit entry point, cleaner layering, and a guarded legacy-data cleanup path.

SELECTED IMPLEMENTATION DECISIONS

Translation at read time	Original message text remains the source of truth. The server obtains the authenticated viewer's preferred language, calls the translation service, and returns displayText without persisting viewer-specific copies.
Reset tokens hashed at rest	The email carries the raw one-time token, while MongoDB stores only its SHA-256 digest plus expiry. Successful reset clears the token fields and invalidates older JWT sessions.
HttpOnly browser session	The revised browser session no longer exposes its JWT to JavaScript through localStorage. HttpOnly/SameSite cookie attributes reduce script access and cross-site sending; Bearer auth remains available for API testing.
Safe rendering over HTML injection	Forum titles, descriptions, usernames, profile values, and message content are inserted as text rather than interpreted as HTML, closing the stored-XSS route present in the original frontend.
Backward-compatible cleanup	The revised schema stops writing obsolete duplicate message fields while existing MongoDB documents can continue to work. Optional migration performs guarded cleanup only after validating required content.

TECHNOLOGY FOOTPRINT

Node.js 18+ | Express 4 | MongoDB / Mongoose | bcryptjs | JSON Web Tokens | Google Cloud Translation API v2 | Nodemailer / SMTP | Vanilla JavaScript | HTML / CSS

Summary

I built the original forum, identified weaknesses in my own implementation, and independently re-engineered it across feature completion, security, architecture, integrations, and migration - demonstrating the ability to audit, refactor, harden, and finish an existing codebase.