Assignment is to build and complete existing given code of Back End. The system includes side-by-side display of mortgage data and source images, record locking, batch handling, audit trails, and integration with county websites. The specification is structured to cover the functional requirements, front-end UI, backend database, and technical considerations. You are expected to troubleshoot and build this Back End *(and integrate it with given Front End, only for Full Stack Developer)*. You may need to edit and fix some build problem so that Back End will enable below features. You can change only ORM (either existing TypeORM or Drizzle ORM) and rest all technologies and Frame works need to be maintained same.

You are expected to successfully resolve all compile and build problems and expected to complete below requirement. *(If you are Only Back End Developer then you must implement AutoComplete vector search & pagination)*

Back End Code Link https://drive.google.com/file/d/15FUMzaV1J9S3lXKzK43QI4xbG7DfgQ8n

Front End Code Link https://drive.google.com/file/d/1gI-V0n36Bva8jCNBfIPjZC8TeuN5JT1K

You are free to use any AI programming tools to complete this assignment.

Product Specifications: Mortgage Data Entry and Verification System

1. Overview

The system is a web-based application designed to streamline the entry, verification, and auditing of mortgage data. It enables virtual assistants (VAs) to view, edit, and verify mortgage records while ensuring data integrity through record locking, batch handling, and audit trails. The front end is built using ReactJS for a responsive, intuitive user interface, and the backend uses PostgreSQL for reliable data storage and management.

---

2. Functional Requirements

2.1 Core Features

- **Side-by-Side Display:**
    - Display extracted mortgage data and the original source image (tiff image format) side-by-side.
    - Source image is rendered in a browser-accessible format (tiff image format).
    - Provide a direct hyperlink to the county website for each record, pre-filling fields (e.g., APN, address) where possible to access the live source document.
- **Data Entry and Verification:**
    - VAs can enter or edit mortgage data fields.
    - Two action buttons: "Good" (approve record) or "Bad" (flag for correction).
    - Upon clicking "Good" or "Bad", the record is submitted, verified, and locked to prevent further edits.
    - Fields include:

        - Property Address: Text
        - Transaction Date: Date (calendar picker)
        - Borrower/Property Owner Name: Text
        - Loan Officer Name: Text
        - NMLS ID: Number
        - Loan Amount: Number
        - Loan Term (# of Years): Number
        - Down Payment: Calculated (Sales Price - Loan Amount, read-only)
        - APN: Text
        - Entered By: Text (read-only, auto-populated with VA's username)
        - Entered By Date: Date (read-only, auto-populated with timestamp)
        - Reviewed By: Text (read-only, auto-populated with VA's username upon verification)
        - Reviewed By Date: Date (read-only, auto-populated with timestamp upon verification)
- **Record Locking and Reassignment:**
    - Implement a locking mechanism to ensure no two VAs work on the same record simultaneously.
    - If a VA exits the browser or times out (e.g., after 10 minutes of inactivity), the record is unlocked and reassigned to another VA.
    - Timeout detection via session management (e.g., React state or server-side session tracking).
- **Batch Handling:**
    - Records are grouped into daily or weekly batches in the backend for administrative purposes.
    - VAs see only their assigned records in a clean, paginated list.
    - Each record displays a summary of extracted data and the source image.
- **Audit Trail:**
    - Track all actions (entry, edit, verification) with timestamps and VA identifiers.
    - Store audit logs in the database for compliance and review.
- **Accessibility:**
    - Accessible via company-provided devices (e.g., laptops, desktops) with modern browsers (Chrome, Firefox, Edge).

    - Responsive design to support varying screen sizes (minimum 1024x768 resolution).
- **AutoComplete search & pagination** **(must to implement)**

2.2 Non-Functional Requirements

- Performance: Load records and images in <2 seconds (assuming standard internet speeds).
- Scalability: Support up to 100 concurrent VAs with minimal latency.
- Security:
    - Role-based authentication (e.g., VA, Admin) using JWT or OAuth.
    - Encrypt sensitive data (e.g., Borrower Name, NMLS ID) in transit (HTTPS) and at rest (PostgreSQL encryption).
    - Restrict access to records based on VA assignment.
- Reliability: Ensure 99.9% uptime with database backups and failover mechanisms.
- Usability: Intuitive UI with minimal training required (e.g., clear labels, tooltips, and validation messages).