

## 1. DATABASE FILES

prisma/schema.prisma

Purpose: Prisma ORM schema defining the complete database structure with 8 tables (student\_user, job\_posting, application\_entry, application\_folder, application\_folder\_assignment, interview, reminder, application\_document) and 4 enums (LocationType, ApplicationStatus, InterviewType, DocumentType). Generates TypeScript types and SQL migrations.

prisma/migrations/20251203220135\_init/migration.sql

Purpose: SQL migration file that creates all database tables, constraints, indexes, and enums in PostgreSQL. Auto-generated by Prisma.

prisma/seed.ts

Purpose: Database seeding script to populate initial test data for development.

lib/prisma.ts

Purpose: Prisma Client singleton configuration for database connections. Ensures one client instance across hot reloads in development.

---

## 2. AUTHENTICATION & SECURITY FILES

lib/auth.ts

Purpose: NextAuth.js v5 configuration implementing JWT-based session management, bcrypt password hashing (10 salt rounds), and credential-based authentication. Handles login/logout logic.

middleware.ts

Purpose: Next.js middleware for route protection. Redirects unauthenticated users to login page when accessing protected routes (/jobs).

---

### 3. API ROUTE FILES (Backend)

`app/api/register/route.ts`

Purpose: POST endpoint for user registration. Validates email uniqueness, hashes passwords with bcrypt, creates new user in database. Returns JSON response.

`app/api/jobs/route.ts`

Purpose:

- POST - Create new job application with job posting (creates/upserts job\_posting and application\_entry)
- GET - Retrieve all applications for authenticated user

`app/api/jobs/[id]/route.ts`

Purpose:

- PATCH - Update application status (supports workflow: INTERESTED → APPLIED → ONLINE\_ASSESSMENT → PHONE\_SCREEN → ONSITE → OFFER/REJECTED)
- DELETE - Delete application entry with authorization check

`app/api/applications/[id]/interviews/route.ts`

Purpose:

- POST - Create new interview record for an application (type, datetime, notes, outcome)
- GET - Retrieve all interviews for an application sorted by datetime
- Validates user owns the parent application

`app/api/applications/[id]/reminders/route.ts`

Purpose:

- POST - Create new reminder for an application (type, datetime, message)
- GET - Retrieve all reminders for an application sorted by datetime
- Validates user owns the parent application

app/api/applications/[id]/documents/route.ts

Purpose:

- POST - Handle file upload (multipart/form-data), save to /public/uploads/, create database record
- GET - Retrieve all documents for an application sorted by upload date
- Validates user owns the parent application
- Supports: PDF, DOC, DOCX, TXT files

app/api/folders/route.ts

Purpose:

- POST - Create new folder for organizing applications
- GET - Retrieve all folders for authenticated user with application counts

app/api/folders/[id]/route.ts

Purpose:

- DELETE - Delete folder (cascade deletes assignments)
- Validates user owns the folder

app/api/folders/[id]/assign/route.ts

Purpose:

- POST - Assign application to folder (creates M:N relationship in application\_folder\_assignment)
- DELETE - Remove application from folder
- Validates user owns both application and folder

---

#### 4. PAGE FILES (Frontend - Server Components)

app/page.tsx

Purpose: Landing page with links to login and register. Public route.

app/login/page.tsx

Purpose: Login page with credential form. Uses NextAuth signIn() action. Redirects to /jobs on success.

app/register/page.tsx

Purpose: Registration page with form (name, email, password, phone). Calls /api/register endpoint. Auto-login after registration.

app/jobs/page.tsx

Purpose: Main dashboard (protected route). Server component that fetches all applications, interviews, reminders, documents, and folders via Prisma. Serializes dates for client. Passes data to JobsPageClient component.

app/layout.tsx

Purpose: Root layout component. Configures fonts, metadata, wraps app with SessionProvider for NextAuth.

---

## 5. REACT COMPONENTS (Frontend - Client Components)

components/JobsPageClient.tsx

Purpose: Main client component for dashboard. Displays application cards with stats counters (Total, Applied, Interviewing, Offers), folder sidebar, search functionality, and renders all sub-components. Implements filtering by folder and search query.

components/CreateJobForm.tsx

Purpose: Modal form for creating new job application. Fields: company name, job title, job URL, location, location type, salary range, status, notes. Auto-populates user email from session.

components/StatusDropdown.tsx

Purpose: Dropdown component to update application status. Calls PATCH /api/jobs/[id] and refreshes page on success.

components/ScheduleInterviewButton.tsx

Purpose: Modal form for scheduling interviews. Fields: interview type (phone/video/onsite/technical/behavioral), datetime, notes, outcome. Converts enum to uppercase before API call.

components/SetReminderButton.tsx

Purpose: Modal form for creating reminders. Fields: reminder type

(deadline/follow\_up/interview\_prep/document\_upload/other), datetime, message. Shows overdue detection with red/yellow backgrounds.

components/UploadDocumentButton.tsx

Purpose: Modal form for uploading documents. Fields: document type (resume/cover\_letter/other), file input. Uses FormData API for multipart upload. Accepts .pdf, .doc, .docx, .txt files.

components/DeleteButton.tsx

Purpose: Delete button for applications. Calls DELETE /api/jobs/[id] with confirmation, refreshes page on success.

components/CreateFolderDialog.tsx

Purpose: Dialog/modal for creating new folders. Field: folder name. Calls POST /api/folders.

components/FolderList.tsx

Purpose: Sidebar component displaying all folders with application counts. Handles folder selection for filtering and folder deletion.

components/AddToFolderButton.tsx

Purpose: Dropdown button to assign/remove applications from folders. Shows current folders with checkmarks. Calls POST/DELETE /api/folders/[id]/assign.

---

## 6. UI COMPONENT LIBRARY (shadcn/ui)

components/ui/button.tsx

components/ui/input.tsx

components/ui/label.tsx

components/ui/select.tsx

components/ui/textarea.tsx

Purpose: Shared UI primitives used across modals and forms.

---

## 7. UTILITY & CONFIGURATION FILES

lib/utls.ts

Purpose: Utility functions for className merging (clsx + tailwind-merge). Used across all components.

next.config.ts

Purpose: Next.js configuration. Enables Turbopack, sets logging options.

tailwind.config.ts

Purpose: Tailwind CSS configuration with custom colors, border radius, and CSS variables for theming.

tsconfig.json

Purpose: TypeScript compiler configuration with path aliases (@/\* imports).

package.json

Purpose: Node.js dependencies and scripts. Includes Next.js 16.0.3, Prisma 6.19.0, NextAuth v5, bcrypt, Tailwind CSS.

components.json

Purpose: shadcn/ui configuration for component installation.

postcss.config.mjs

Purpose: PostCSS configuration for Tailwind CSS processing.

.env

Purpose: Environment variables (DATABASE\_URL for PostgreSQL connection, AUTH\_SECRET for JWT signing). NOT INCLUDED IN SUBMISSION.

.gitignore

Purpose: Git ignore rules. Excludes node\_modules, .next, .env, /public/uploads/.

## FILE STRUCTURE SUMMARY

jaeger-ai/

app/

api/

register/

jobs/

[id]/

applications/[id]/

interviews/

reminders/

documents/

folders/

[id]/assign/

jobs/

login/

register/

page.tsx

layout.tsx

components/

ui/

lib/

auth.ts  
prisma.ts  
utils.ts  
prisma/  
  schema.prisma  
  migrations/  
  seed.ts  
public/  
  uploads/  
Configuration files (package.json, tsconfig.json, etc.)

---

#### TOTAL FILE COUNT

TypeScript/TSX files: 30+

Configuration files: 8

Database files: 3

All source code uses:

- Prepared statements (Prisma ORM parameterized queries)
- Session management (NextAuth JWT)
- Password hashing (bcrypt with 10 salt rounds)
- Authorization checks (every API validates ownership)
- XSS prevention (React auto-escaping)