Jaeger.AI - Team Contributions Summary
Course Project - Part 3: Web Application
Date: December 4, 2025

TEAM MEMBERS

Name: Leo Yu
NetID: zy3707
Role: Backend Development

Name: Aadi
NetID: arn8074
Role: Frontend Development

WORK DISTRIBUTION (approximate split, both took on a lot of hats)

Leo Yu (zy3707) - Backend Development

Primary Responsibilities:

- Database design and implementation
- API route development
- Query optimization
- Security implementation

Specific Tasks:

1. Database Design & Schema

- Designed Prisma schema with 8 tables (student_user, job_posting, application_entry, application_folder, application_folder_assignment, interview, reminder, application_document)

- Defined 4 enums (LocationType, ApplicationStatus, InterviewType, DocumentType)
- Implemented relationships (1:N, M:N)
- Created database migrations
- Configured PostgreSQL connection (Neon Cloud)

2. API Route Implementation (All 12+ Endpoints)

- POST /api/register - User registration with bcrypt password hashing
- POST /api/jobs - Create application with job upsert logic
- GET /api/jobs - Fetch all applications with nested relations
- PATCH /api/jobs/[id] - Update application status
- DELETE /api/jobs/[id] - Delete application with cascade
- POST /api/applications/[id]/interviews - Create interview
- GET /api/applications/[id]/interviews - Fetch interviews
- POST /api/applications/[id]/reminders - Create reminder
- GET /api/applications/[id]/reminders - Fetch reminders
- POST /api/applications/[id]/documents - File upload handling
- GET /api/applications/[id]/documents - Fetch documents
- POST /api/folders - Create folder
- GET /api/folders - Fetch folders with counts
- DELETE /api/folders/[id] - Delete folder
- POST /api/folders/[id]/assign - Assign application to folder (many-to-many)
- DELETE /api/folders/[id]/assign - Remove application from folder

3. Authentication & Security

- NextAuth.js v5 configuration (lib/auth.ts)
- Credentials provider setup with bcrypt password verification
- Session management with JWT strategy
- Middleware for route protection (middleware.ts)
- Authorization checks in all API routes (verify user ownership)
- Prepared statement implementation via Prisma ORM

4. Database Query Optimization

- Designed efficient queries with proper indexing
- Implemented eager loading for nested relations
- Optimized batch queries for applications list
- Added proper ordering (for example, interviews by datetime)

5. File Upload System

- Implemented multipart/form-data handling
- File storage in /public/uploads/ directory
- Unique filename generation to prevent collisions
- File type validation (PDF, DOC, DOCX, TXT)
- Database record creation for uploaded files

6. Prisma Configuration

- Prisma Client setup (lib/prisma.ts)
- Database seeding script (prisma/seed.ts)
- Managed migrations and schema changes

7. Bug Fixes & Debugging

- Fixed enum case mismatch issues (uppercase conversion)
- Resolved Turbopack caching problems
- Fixed status counter logic (include rejected in "Applied")
- Debugged authorization check edge cases
- Fixed Next.js 16 async params compatibility

Files Owned by Leo:

- /prisma/schema.prisma
- /lib/auth.ts
- /lib/prisma.ts
- /middleware.ts
- All files in /app/api/ (all API route files)
- Database migration files
- All database queries implementation

Aadi (arn8074) - Frontend Development

Primary Responsibilities:

- UI/UX design and implementation
- React component development
- Client-side state management
- User interface interactions

Specific Tasks:

1. ER Diagram Design

- Created Entity-Relationship diagram for Part 2
- Designed database schema visual representation
- Documented relationships between entities
- Defined cardinality (1:N, M:N)

2. Page Components (Next.js App Router)

- app/page.tsx - Landing page
- app/login/page.tsx - Login page with NextAuth integration
- app/register/page.tsx - Registration page with form validation
- app/jobs/page.tsx - Main dashboard (server component)
- app/layout.tsx - Root layout with SessionProvider

3. React Client Components (15 Components)

- components/JobsPageClient.tsx - Main dashboard with stats, filtering, search
- components/CreateJobForm.tsx - Application creation modal form
- components/StatusDropdown.tsx - Status update dropdown with color coding
- components/ScheduleInterviewButton.tsx - Interview scheduling modal
- components/SetReminderButton.tsx - Reminder creation modal
- components/UploadDocumentButton.tsx - Document upload modal with file input
- components/DeleteButton.tsx - Delete confirmation and action
- components/CreateFolderDialog.tsx - Folder creation dialog
- components/FolderList.tsx - Folder sidebar with counts and selection
- components/AddToFolderButton.tsx - Multi-select folder assignment dropdown

- components/ui/button.tsx - Reusable button component (shadcn/ui)
- components/ui/input.tsx - Form input component
- components/ui/label.tsx - Form label component
- components/ui/select.tsx - Dropdown select component
- components/ui/textarea.tsx - Multi-line text input

4. UI/UX Features

- Dashboard statistics cards (Total, Applied, Interviewing, Offers)
- Application cards with expandable details
- Color-coded status badges
- Folder sidebar with filtering
- Search bar with real-time filtering
- Modal forms with validation
- Responsive layout with Tailwind CSS
- Loading states and error handling

5. Client-Side Logic

- Filter applications by folder (state management)
- Search functionality (company name and job title)
- Combine folder and search filters
- Overdue reminder detection with visual indicators (red or yellow backgrounds)
- Form state management for modals
- File input handling for uploads

6. Styling & Design

- Tailwind CSS configuration (tailwind.config.ts)
- Custom color scheme
- Responsive grid layouts
- Card designs with hover effects
- Badge and tag styling
- Modal/dialog styling
- Button variants (primary, outline, ghost)

7. Integration with Backend

- API calls from client components (fetch)
- Form submission handling
- Error handling and user feedback (alerts)
- Page refresh after mutations
- Session data display (user name, email)

Files Owned by Aadi:

- All files in /app/ (except API routes)
- All files in /components/
- /tailwind.config.ts
- Part 2 ER diagram file

COMMUNICATION & COLLABORATION

Development Approach:

- Leo built backend APIs first

- Aadi built frontend components concurrently
- Integration testing done together
- Bug fixes coordinated as needed

WORK DIVISION SUMMARY

Division Summary:

- Leo Yu (zy3707): Backend (Database, API, Security, Queries) ~ 50%
- Aadi (arn8074): Frontend (UI, Components, ER Diagram) ~ 50%