Project Proposal - Team 7

Grader Assignment System

Project Overview

Assigning graders to professors and course sections is a time-consuming and fragmented process. The existing process requires the hiring manager to manually filter candidates based on certain criteria and create spreadsheets to match the candidates to the professors. This is an effort that can take up to one to two weeks to complete because the current system lacks a standardized approach and the information needed to make these decisions are spread across multiple platforms such as Handshake, Excel and e-mail.

The Grader Assignment System aims to automate the process of assigning candidates to professors and courses by implementing a centralized platform. Assignment of graders will be optimized using a matching algorithm that is based on predefined criteria such as qualification, prior experience and availability. This will not only save time but improve the accuracy and fairness in grader assignments. The system will offer a user friendly interface for hiring managers to make the entire process more efficient and less prone to human error.

Project Scope

The Grader Assignment System will be developed as a web application to optimize grader assignments for academic institutions, The core functionality will allow hiring managers to upload candidate information and course requirements. A matching algorithm will be implemented to automate the entire matching process and results can be viewed immediately. Hiring managers will also be able to manually adjust assignments if needed, allowing for flexibility in the decision making process.

Key features of this product:

- ➢ Process and extract CV information from PDF files.
- ➢ Read and generate CSV and Excel files
- ➢ Option to view profile of selected candidate
- ➢ Resume parsing
- ➢ Consider professor recommendations in assignment decisions
- ➢ Allow manual override

Stretch Goals

- ➢ View other/alternative candidates

- ➢ Notification system for candidates for their assignment status
- ➢ Advanced filtering and search options for hiring managers to refine results

Project Objectives

Key Objectives

- ➢ Develop a reliable matching algorithm to assign graders to courses based on qualification, availability, and professor recommendations
- ➢ Integrate ability to read and generate CSV/Excel files and parse PDF files in one centralized area without relying on multiple fragmented systems
- ➢ Design a web application that allows hiring managers to view, adjust, and finalize assignments easily

Measurable Goals

- ➢ Reduce Assignment Time: Cut down the process of grader assignment from 1-2 weeks to less than 3 days
- ➢ Accuracy in Matching
    - How do we measure this
- ➢ File Handling Capacity: Successful process the uploaded PDFs, CSVs, and Excel files without errors

Expected Outcome and Deliverables

- ➢ A fully functional web application with automated assignment capabilities
- ➢ Ability to upload, process, and generate CSV/Excel files and read data from PDF documents
- ➢ A matching algorithm with transparent reasoning for every assignment

Specifications

User Interface (UI) Design

- ➢ The platform will be a website application.
- ➢ Key Pages/Screens + Interactive Elements
    - **Login and Dashboard Page**: Secure login for users with role-based access and an overview of assignments, pending applications, and flagged mismatches.
        - Login Form
            - Log In and Forget Password
            - UT Dallas credentials
                - From the credentials, roles will be assigned
        - Dashboard
            - Grader
                - Summary of Assignments

                - Total Assignments, Mismatched Assignments, and Professor Information
            - Professor
                - View courses eligible for grader, matches available, and grader information
            - Admin
                - Add/remove professors, conflict resolution override, manual assignment, delete/archive previous applications and resumes
        - Navigation Menu
            - Allows you to access the other pages
            - Logout option
- **Candidate Management Page**: Upload and manage grader applications, extract resume details, and filter candidates based on experience, qualifications, and availability.
    - Upload Section
        - Upload CSV (Batch Uploads) or PDF of resumes of applicants
    - Candidate List Table
        - Contains Name, Grade, Availability, etc of each applicant
        - Other buttons such as view more, delete, edit, etc.
    - Filtering/Search Bar
        - Search by name or filter by certain stat (Grade, Experience, etc.)
- **Course & Professor Assignment Page**: Display available courses, match graders using an algorithm, and allow manual assignment overrides.
    - Course List Table
        - Contains Class Name, Prof, # of Open Positions, Assigned Graders
        - Action Buttons
            - Assign Graders
            - Edit Assignment (Manual override for assignments)
            - View Prof recommendations
    - Grader Assignment Form
        - Dropdown for Selecting Course
        - Dropdown for Selecting certain Applicant (shows overall ‘fit’)
        - Checkbox to show recommendation by prof
        - Submit Button
- **Assignment Review Page**: Provide an overview of grader assignments, highlight conflicts, and allow final confirmation before submission.
    - Assignment Summary Table
        - Displays Course Name, Professor, Assigned grader, Conflicts (if they exist)
        - Action Buttons
            - Confirm Assignment
            - Edit Assignment
            - View Conflict (if they exist)

                - Conflict Resolution Page
                    - Auto Re-assign (re-run the algo)
                    - Manually pick another grader
                    - Flag for Review
    - **Report and Analytics Page**: Generate visual reports on grader demand, assignment trends, and historical data.
        - Filter Date and Range of Analytics
            - Select date range
            - Filter by course
            - Filter by professor
        - Reports
            - Generate reports button
                - Export Button (CSV or PDF)
        - Charts and Graph
            - Assignment Trends
            - Grader Demand Analysis
            - Historical Data Comparisons

Backend & API

- Database and API Structure: Authentication and dashboard API, and manages user authentication, secure access, and dashboard statistics.
    - Secure login for users (role-based access)
        - Roles: Admin, Professor, Grader
    - Reset password
    - Dashboard Overview
        - Pending applications
    - Upload & Manage Candidates
        - Upload CSV (batch upload) or PDF resumes, fetch list of candidates with filters, edit candidate details, delete candidate
        - Manages course grader assignments, recommendations, and manual overrides.
    - Course List
        - Fetch available courses with open positions, auto assign graders using an algorithm
    - Database Structure
        - Stores class name, professor, open positions, assigned graders
- Grader Management API
    - Handling grader applications and availability tracking
    - Upload and Manage Graders
        - Input: grader applications and resume (PDF or CSV)
        - Output: Fetch list of available graders
        - Control: Grader information, GPA, experience
        - Test: Availability, preferences
- Grader Assignment Review and Conflict Resolution API

        - Re-running algorithm in event grader assigned creates merge between another potential grader match
        - Manually reassigning grader
        - Flagging assignment issues
- Reporting and Analytics
    - Filters & Data Selection
        - Fetching course specific statistics
        - Retrieving specific statistics guided by professor
    - Database Structure
        - Report collection to store grader trends and historical comparisons
- Infrastructure
    - Framework: Express.js (Node.js)
    - Database: MongoDB (NoSQL)
    - Authentication: JWT-based security
    - File Storage: Cloud/local storage for resumes (PDF or CSV file)
    - Deployment: Dockerized backend with API routes

Tech Stack

- Frontend: (React.js, HTML, CSS, JavaScript)
- Backend: (Node.js, Express.js)
- Database: (MongoDB)
- Cloud and Hosting: UT Dallas Linux VM

Hardware Requirements

- Sufficient RAM to hold the data in memory while processing
- Basic storage capability to store CSV and Excel files of grader resumes
- Modern CPU with at least one core

Software Requirements

- Docker: Used in development & deployment phase
- Github Actions: Automate the software workflows and used to implement our CI/CD pipeline
- Linux VM that supports the toolchains we're using (so just not an extremely old OS)

Project Timeline

(either in phases or week by week schedule)

Phase	Duration	Tasks (define frontend/backend/general)
Phase 1: Planning and Research	1 Week	
Phase 2: Design	1 Week	Plan out the visual aspects of the website, create architecture design. Acquire data formats so we can start prototyping our matching algorithm. Decide on documentation / specification tools and strategy for API.
Phase 3: Initial Development	2 Weeks	Start design of frontend and organize backend implementation.
Phase 4: Front End Development & Algorithm Implementation/Integration	3 Weeks	Focus on front end, continue back end
Phase 5: Algorithm Implementation/Integration	2 Weeks	
Phase 6: Testing and Debugging	1-2 Weeks	
Phase 7: Final Adjustments	1-2 Weeks	
Documentation and Final Review	1 Week	
Presentation and Delivery	May 2	Demo, Q&A

Project Team

Role	Team Member	Responsibilities
Backend Development	Solomon Pierce	API and CI/CD Pipeline Architect
Backend Development	Arsal Hussain	Database design and API
Frontend Development	Ji Min Yoon	Design and Prototype UI
Frontend Development	Sophie Tran	Design and building interactive UI

Full Stack Engineer	Rayyan Waris	Implement UI and API Integration

Rotation Leader

Will loop each cycle for even distribution of leadership and participation of meeting minutes

Dates	**Leader**
2/11, 3/25, 4/29	Solomon Pierce
2/18, 4/1	Arsal Hussain
2/25, 4/8	Ji Min Yoon
3/4, 4/15	Sophie Tran
3/11, 4/22	Rayyan Waris

Links

- Github Repository: https://github.com/cs4485-s25-alagar-t7