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 professors 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 and Excel.

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:

- ➢ Get CV data from a different website source, such as Handshake
- ➢ Read and generate CSV and Excel files
- ➢ Process and extract necessary information from PDF files.
- ➢ Option to view profile of selected candidate
- ➢ Resume parsing
- ➢ Consider professor recommendations into assignment

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
- ➢ 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
    - ○ **Login and Dashboard Page**: Secure login for users with role-based access and an overview of assignments, pending applications, and flagged mismatches.
    - ○ **Candidate Management Page**: Upload and manage grader applications, extract resume details, and filter candidates based on experience, qualifications, and availability.
    - ○ **Course & Professor Assignment Page**: Display available courses, match graders using an algorithm, and allow manual assignment overrides.
    - ○ **Assignment Review Page**: Provide an overview of grader assignments, highlight conflicts, and allow final confirmation before submission.
    - ○ **Report and Analytics Page**: Generate visual reports on grader demand, assignment trends, and historical data.
- ➢

Backend & API

- ➢ Node.js
- ➢ Express.js
- ➢ MongoDB

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 deployment phase
- Github Actions: Automate the software workflows and implemented during 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
Phase 3: Initial Development	2 Weeks	Focus on backend
Phase 4: Front End Development	2 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 Week	
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

Links

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