AI Resume

by

Artificial J.A.M.A

Adriana Altamirano

A’nya Carr

Molisha Khosla

Joshua Roasa

Submitted in partial fulfillment for the requirements of

CPS 4951 - Senior Capstone in Computer Science

Spring 2024

Table of Contents:

1. Topic
   1. Problem
   2. Motivation
   3. Description
   4. Scope
2. Teams and Tasks
3. Project Breakdown

3.1 Architecture

3.2 Data Analysis

1. Challenges and Takeaway
2. Future Work
3. Additional Notes
4. References
5. Topic
   1. Problem

The integration of AI into hiring processes has been met with both enthusiasm and skepticism, raising questions about its effectiveness in reducing bias. While the potential benefits include streamlining recruitment and enhancing objectivity, concerns persist regarding the perpetuation of biases inherent in algorithms. The AI Resume Parser project seeks to address these concerns by mitigating bias and improving efficiency in the hiring process.

Current AI Resume Parsers face challenges related to bias and unfamiliar formats, creating barriers for qualified applicants and impacting the fairness of hiring decisions. These biases can stem from historical data or institutional practices, reinforcing disparities in recruitment outcomes. Despite their potential to streamline hiring, algorithms risk replicating and amplifying existing biases, posing ethical and practical challenges for recruiters and applicants alike.

* 1. Motivation

The motivation behind the AI Resume Parser project stems in the recognition of these challenges and the need to develop a solution that promotes fairness and inclusivity in hiring. Inspired by the work of Miranda Bogen and informed by Phase 1 and Phase 2 Planning, our team aims to empower recruiters to identify and source talented candidates effectively, while also simplifying the application process for job seekers.

Our objective is to create an AI-powered resume parsing system that addresses the limitations of existing tools by providing real-time feedback to applicants and assisting recruiters in making informed decisions. By leveraging AI technology, we aspire to enhance the efficiency and objectivity of the hiring process, fostering better matches between candidates and employers.

* 1. Description

The AI Resume Parser is a software application designed to automate resume screening and evaluation, using AI algorithms to assess candidate qualifications and match them with job requirements. The system features a user-friendly interface that allows applicants to submit their resumes and receive immediate feedback on formatting issues or deficiencies.

On the recruiter's side, the system enables them to post job listings, customize parsing criteria, and review accepted resumes efficiently. By incorporating diverse job categories and customization options, the AI Resume Parser aims to cater to the specific needs of recruiters across various industries.

* 1. Scope

The scope of the project encompasses the development and implementation of the AI Resume Parser, focusing on phases such as analysis, design, implementation, testing, and maintenance. The system will address bias by providing clear feedback to applicants, implementing robust data privacy measures, and allowing recruiters to customize parsing criteria based on job requirements.

Furthermore, the AI Resume Parser will be trained on diverse datasets to mitigate bias. By feeding the system with resumes from a variety of backgrounds and demographics, we aim to reduce the risk of algorithmic bias and ensure fair and equitable outcomes in the hiring process.

While the AI Resume Parser aims to improve the efficiency and objectivity of the hiring process, it acknowledges the limitations of algorithms and the importance of human judgment in recruitment decisions. Continuous evaluation and improvement of the system will ensure its effectiveness in promoting fairness and equity in hiring practices.

Drawing from Phase 1 and Phase 2 Planning, our project timeline aligns with the Software Development Life Cycle, emphasizing iterative development and adaptation to evolving needs and challenges.

1. Teams and Tasks

Collective Contribution during Phase 1 & Phase 2 Planning

During Phase 1 & Phase 2 Planning, the team worked collectively to lay the groundwork for the AI Resume Parser project. Together, they honed their skills in documentation and project planning, applying their collective expertise to identify problems, objectives, and user interactions for the system. Through collaborative discussions and brainstorming sessions, they formulated a comprehensive understanding of the project scope and requirements. Additionally, the team collaborated on planning the project timeline, ensuring that key milestones and deliverables were identified and scheduled effectively. This collaborative effort laid the foundation for the successful execution of subsequent project phases.

#### 

#### Adriana Altamirano

During Phase 3 Analysis, Adriana collaborated with the team to define use cases for the AI Resume Parser application, drawing from her understanding of user needs and behaviors. In Phase 4 Design, she partnered with Joshua to design the User Interface for the web pages, ensuring usability and visual appeal. In Phase 5 Implementation, Adriana took on front-end development tasks for the applicant pages, learning and applying Bootstrap, HTML, and CSS to create the Home Page (login page) and all Applicant pages. She skillfully implemented responsiveness and various Bootstrap components such as buttons, filters, search bars, and accordions, enhancing the user experience across different devices. Adriana's dedication to mastering new skills and attention to detail contributed to the successful implementation of the front-end interface for the AI Resume Parser application. She took the front end development work to her partner Anya who worked on the back end development for the applicant pages.

Anya Carr

During Phase 3, A’nya focused on defining the use cases for the AI Resume Parser, while collaborating with Joshua to create the corresponding Use Case Diagram. Moving into Phase 4, A’nya leveraged her SQL expertise to craft MYSQL Tables for the project and populated them with relevant data entries. In Phase 5, A’nya and Adriana collaborated on developing the Applicant side of our website, where A’nya utilized PHP to establish database connectivity, handle data entry operations, and manage data retrieval, update, and deletion functionalities. Additionally, A’nya and Joshua worked together on debugging PHP files. Molisha played a significant role in the design aspect of the site and conducted thorough testing to ensure our code's seamless integration with the site's interface. Molisha and Joshua joined forces to implement CSS for optimizing the responsiveness of the homepage and ensuring its alignment with our website's design plan. Furthermore, Molisha and Joshua collaborated on the development of the Recruiter side of the website, with Joshua also taking charge of creating and updating the necessary database tables utilizing his expertise in SQL.

Molisha Khosla

For phase 1&2 everyone worked together, we created all use cases and class diagrams and work was equally divided. Starting phase 3, we split in 2 teams. [Adriana Altamirano](mailto:altamiad@kean.edu)and Anya Carr started working on applicant side and I and [Joshua Roasa](mailto:roasaj@kean.edu)started working on recruiter side. I was responsible for all front end and [Joshua Roasa](mailto:roasaj@kean.edu)I worked on the backend using php making sure whatever I designed has proper functionality. Adriana also helped in final documentation, she combined all our phase documentations to a final document.

Joshua Roasa

During Phases 1 and 2, everyone collaborated to determine the topic, identify problems and objectives, determine how users would accomplish their tasks when interacting with our system, and make our system more useful and usable. Additionally, we created a timeline for the project using the Software Development Life Cycle. During this phase, everyone was able to contribute ideas on how to proceed with the project, showing enthusiasm for working together as a group.

In Phase 3, which is the Analysis phase, every team member was tasked with identifying at least three Use Cases. Each of us had to provide descriptions, identify the level of the use case, the primary actor, pre-conditions, post-conditions, and the main scenario. After completing these tasks, Anya and I (Joshua) worked together to showcase the Use Case Diagram. During this phase, everyone demonstrated their analytical skills and collaborated effectively with each other.

Moving on to Phase 4, the Design phase, Molisha created the ER diagram, including tables, attributes, strong/weak relationships, and primary/foreign keys, while I (Joshua) assisted in creating tables that would later be implemented in MySQL by Anya. After creating the tables, Anya utilized her MySQL skills, acquired from her database class, to create the MySQL tables. Additionally, during this phase, we worked on designing the User Interface for webpages. Adriana and I (Joshua) collaborated to design the UI using Figma. After completing these tasks, Anya and Molisha suggested some design changes to improve the visual appeal of the web pages.

Moving to Phase 5, Implementation and Documentation, we divided the members into two teams. Anya and Adriana were assigned to work on the Applicant pages, while Molisha and I (Joshua) were assigned to complete the Recruiter Side. During this phase, Anya demonstrated her PHP and MySQL skills to develop the backend of the Applicant pages, while Adriana showcased her HTML and CSS skills to complete the front-end of the Applicant Pages. For the Recruiter side, Molisha was tasked with completing the front-end using her remarkable skills in HTML and CSS. I (Joshua) was assigned to complete the backend, utilizing PHP, MySQL, JavaScript, and Ajax to accomplish these tasks. While implementing the backend, I (Joshua) encountered some problems and added new MySQL tables and altered some of the tables that were created during Phase 4 by Anya.

1. Project Breakdown
   1. Architecture

The AI Resume Parser project was built using a combination of languages, frameworks, and platforms to create a user-friendly application. The following technologies were utilized in various aspects of the project:

* HTML: HTML (Hypertext Markup Language) was used for creating the structure and content of webpages. It provided the foundation for building the user interface elements and organizing the layout of the application.
* CSS: CSS (Cascading Style Sheets) was employed for styling and visually enhancing the appearance of the webpages. It allowed for customization of colors, fonts, and layout design, ensuring a cohesive and aesthetically pleasing user experience.
* Bootstrap: Bootstrap, a front-end framework, was utilized to streamline the development process and enhance the responsiveness of the application. It provided a collection of pre-designed components and responsive grid system, enabling rapid prototyping and ensuring compatibility across different devices and screen sizes.
* PHP: PHP (Hypertext Preprocessor) served as the server-side scripting language for the project. It facilitated dynamic content generation, server-side processing, and interaction with databases. PHP enabled seamless integration of frontend interfaces with the MySQL database, allowing for efficient data retrieval, manipulation, and storage. It also enabled the implementation of dynamic features such as user authentication, form submission handling, and data validation.
* MySQL: MySQL, a relational database management system, served as the backend database for storing and managing applicant and job-related data. It facilitated data storage, retrieval, and manipulation through structured query language (SQL), ensuring the integrity and security of the application's data.
* JavaScript: JavaScript, a versatile scripting language, was employed to add interactivity and dynamic behavior to the webpages of the AI Resume Parser application. It facilitated client-side scripting tasks such as form validation, DOM manipulation, and event handling, enhancing the user experience by providing real-time feedback and seamless interactions.
* Ajax: jax (Asynchronous JavaScript and XML) was utilized to enable asynchronous communication between the web browser and the server without requiring a page reload. This technology allowed for the retrieval and updating of data from the server in the background, enhancing the application's responsiveness and efficiency. Ajax was instrumental in implementing features such as live search, auto-complete suggestions, and real-time updates, providing a smoother and more interactive user experience.
* Figma: Figma, a collaborative design tool, was used for creating wireframes and mockups of the user interface. It allowed for seamless collaboration among team members and provided a visual representation of the application's design and layout.
* Replit: Replit, an online integrated development environment (IDE), was utilized for collaborative coding and project management. It provided a centralized platform for team members to collaborate on coding tasks, share code snippets, and track project progress in real-time.
* VSCode: Visual Studio Code, a popular source-code editor, was used for writing and editing code. It provided features such as syntax highlighting, code completion, and debugging tools, enhancing the development workflow and productivity of the team.
* MySQL Workbench: MySQL Workbench, a visual database design tool, was utilized for database modeling, SQL development, and administration tasks. It provided a graphical interface for designing and managing MySQL databases, allowing for easier database schema design, query development, and performance optimization.
* LucidChart: LucidChart, a web-based diagramming application, was used for creating flowcharts, entity-relationship diagrams, and other visual representations of system architecture and database schemas. It enabled the team to visualize and communicate complex concepts and relationships, facilitating better understanding and collaboration among team members.

Conclusion:

The AI Resume Parser project employed a diverse set of technologies to create a robust and user-friendly application. By leveraging HTML, CSS, Bootstrap, PHP, MySQL, JavaScript, Ajax, Figma, Replit, VSCode, MySQL Workbench, and LucidChart, the team successfully developed a dynamic platform that simplifies the hiring process for both applicants and recruiters. From designing the user interface to implementing backend functionality and database management, each technology played a crucial role in realizing the vision of the AI Resume Parser. Through collaborative efforts and efficient utilization of these tools, the team delivered an intuitive and efficient solution that enhances the recruitment experience for all stakeholders involved.

* 1. Data Analysis

Team Timeline

Our team decided to use the Software Development Lifecycle and came up with a timeline for each of the 7 Phases. Below is the timeline the team created for the timeline for our application in the course of 14 weeks.

**Phase 1 Planning ( Week 1) & Phase 2 Planning (Week 1 - Week 2)**

P1 Dates: January 29 - February 5

P2 Dates: January 29 - February 12

1. Phase 1: Identifying Problems (Planning)
2. Phase 2: Determine Human Information Requirements (Planning)

**Phase 3 Analysis (Week 2 - Week 3)**

Dates: February 5 - February 12

1. Analyzing system needs (Analysis)

**Phase 4 Design (Week 3 - Week 4)**

Dates: February 12 - February 19

1. Designing recommended system (Design)

**Phase 5 Implementation (Week 4 - Week 10)**

Dates: February 19 - April 1

1. Developing and Documenting Software (Implementation)

**Phase 6 Testing & Integration (Week 10 - Week 13)**

Dates: April 1 - April 22

1. Testing and Maintaining the system (Testing & Integration)

**Phase 7 Maintenance (Week 13 - Week 15)**

Dates: April 22 - May 6

#### Use Case Descriptions and Diagram

The use case description and diagrams will be presented to depict the interactions between different actors and the system components. This diagram will highlight the various functionalities of the AI Resume Parser application from both the applicant and recruiter perspectives.

The use cases for this application that the team came up with were 14 of the following: apply job, login in, log out, add job information, review resume, filter work model, enter basic information, update job information, upload resume, add keywords, close jobs, create account and update profile.

## Use Case #1

### Use Case Name: Apply Job

* ID:# AI01
* Description: Applicants will enter their basic information such as First Name, Last Name, Phone Number and Email to set up a profile. Then applicants will upload their resume.
* Level: High
* Primary Actor: Applicant
* Pre-condition:
  + Must have a resume compatible with the following types: PDF, docx
* Post-condition:
  + Applicant basic information and resume is stored in a database.
* Main Scenario:
  + User enters basic information in a form
  + User uploads resume

## Use Case #2

### Use Case Name: Log in

* ID:# A102
* Description: Actors will log in into the system using their username and password.
* Level: High
* Primary Actor: Recruiter, Applicant
* Pre-condition:
  + Must have existing account before using Create Account
  + Must click Log in
* Post-condition:
  + Actors gain access to system to Add Job functions
* Main Scenario:
  + Actors enter the correct username and password.
  + Actors are logged in

## Use Case #3

### Use Case Name: Log Out

* ID:# AI03
* Description: Actors will click on their profile and hit log out at the bottom
* Level: High
* Primary Actor: Recruiter, Applicant
* Pre-condition: Must be logged in
* Post-condition: Actors won’t have any access to the system
* Main Scenario: Actors are logged out of the system

## Use Case #4

### Use Case Name: Add Job Information:

* ID:# AI04
* Description: This use case allows a Recruiter to add job information to the system, enabling them to manage and update job postings.
* Level: High-Level
* Primary Actor: Recruiter
* Pre-condition: The Recruiter is authenticated and has access permissions to add job information.
* Post-condition: The job information is successfully added to the system and is available for further management.
* Main Scenario:

1. The Recruiter selects the option to add job information in the system.
2. The system presents a form or interface for the Recruiter to input job details such as title, description, requirements, key words they are looking for in the applicant's resume and other relevant information.
3. The Recruiter fills out the required fields with the job information.
4. The system validates the input and saves the job information to the database.
5. The Recruiter confirms the submission.
6. The system displays a confirmation message indicating that the job information has been successfully added.

* Extensions:

1. Update Job Information: If the Recruiter needs to update the job information after adding it, they can use the "Update Job Information" use case.
2. Add Keywords: To enhance searchability and categorization, the Recruiter can use the "Add Keywords" use case to associate relevant keywords with the job information.
3. Close Job: Once a job posting is no longer available or filled, the Recruiter can use the "Close Job" use case to mark it as closed in the system.

## Use Case #5

### Use Case Name: Review Resume

ID:# AI05

* Description: This use case involves the process of a recruiter reviewing an applicant’s resume within the recruitment system to assess their qualifications for a job position.
* Level: High
* Primary Actor: Recruiter
* Pre-condition: The applicant must be logged into the recruitment system and have access to the applicant’s resume.
* Post-condition: The recruiter completes reviewing the applicant’s resume and can have the option to place the applicant’s resume in the Accepted or Rejected buckets.
* Main Scenario:
  + Recruiter logs into the recruiter system account using their credentials.
  + Recruiter navigates to the review resume button.
  + System displays the applicant’s resume, paying attention to the candidate’s qualifications, skills, experience and any other relevant information.
  + If necessary, the recruiter may take notes or market specific sections of the resume for further consideration or discussion.
  + Recruiter evaluates the candidate’s resume based on the job requirements and criteria set by the hiring team.
  + After completing the review, the recruiter has the option to place the applicant’s resume in the Accepted or Rejected buckets based on their evaluation.
  + If the candidate’s resume is placed in the Rejected bucket, the recruiter may provide feedback to the candidate or simply move on to review other applicants.
  + System records the recruiter’s decision and updates the candidate’s status in the recruitment process accordingly.

## Use Case #6

### Use Case Name: Search Keywords

* ID:# AI06
* Description: This use case involves the process of applicants searching for specific job postings using keywords on the website.
* Level: High
* Primary Actor: Applicant
* Pre-condition: The applicant is on the website's job search page.
* Post-condition: The system displays job postings matching the specified keywords.
* Main Scenario:
  + The applicant navigates to the job search page on the website.
  + The system presents the applicant with a search bar or input field to enter keywords.
  + The applicant enters one or more keywords relevant to the desired job title or description, such as "software engineer."
  + The system processes the keyword input and searches through the database of job postings.
  + The system retrieves job postings containing the specified keywords and displays them to the applicant.
  + The applicant reviews the search results to find relevant job postings matching their criteria.

## Use Case #7

### Use Case Name: Filter Work Model

* + - ID:# AI07
    - Description: Users are able to filter out the job they want to apply with an option of remote, onsite, or hybrid
    - Level: High-level
    - Primary Actor: Applicant, Recruiter
    - Pre-condition: user is on search page and is able to filter results
    - Post-condition: the system displays the results selected by the user
    - Main Scenario:

1. User clicks the filter option
2. User can select their preference
3. User can click apply after and view selective options

## Use Case #8

### Use Case Name: Enter Basic Information

* + - ID:# AI08
    - Description: This use case involves entering basic information such as name, last name, email, and phone number. When the users try applying for a job.
    - Level: High-level
    - Primary Actor: Applicant
    - Pre-condition: User need to click the Apply button in order for the prompt to show.
    - Post-condition: The basic information is properly entered in the system.
    - Main Scenario:

1. The User clicks the Apply button for a job
2. A prompt appears asking for the required basic information such as first name, last name, phone number, and email.
3. The User fills out the required fields with their information.
4. The system validates the information. If any fields are missing or invalid, an error message is displayed.
5. The User confirms the submission.

## Use Case #9

### Use Case Name: Update Job Information

* + - ID:# AI09
    - Description: This use case allows a Recruiter to update job information in the system, enabling them to manage and modify existing job postings.
    - Level: High-level
    - Primary Actor: Recruiter
    - Pre-condition: The Recruiter is authenticated and has access permissions to update job information.
    - Post-condition: The job information is successfully updated in the system.
    - Main Scenario:

1. The Recruiter selects the option to update job information.
2. The system presents a list of existing job postings for the Recruiter to choose from.
3. The Recruiter selects the job posting they want to update.
4. The system displays the current details of the selected job posting along with options to edit.
5. The Recruiter makes the necessary updates to the job information.
6. The Recruiter confirms the update.
7. The system displays a confirmation message indicating that the job information has been successfully updated.

## Use Case #10

### Use Case Name: Upload Resume

* + - ID:# AI10
    - Description: In this use case applicant will upload a resume for AI to review and provide a feedback
    - Level: High
    - Primary Actor: Applicant
    - Pre-condition: required well-formatted resume as pdf or .docx file
    - Post-condition: applicant will receive a feedback from AI after uploading their result
    - Main Scenario:

1. Applicant clicks on upload resume
2. Dialogue box appears
3. Applicant selects their resume from the dialogue box
4. Upload the file

## Use Case #11

### Use Case Name: Add Keywords

* + - ID:# AI11
    - Description: The recruiter will add keywords about the job they crucially want. The AI will consider these keywords highly.
    - Level: High
    - Primary Actor: Recruiter
    - Pre-condition:
      * Recruiter must log in
      * Job must be added
      * Must be within a certain number of characters
    - Post-condition
      * Keywords for Job skills and qualifications are saved
    - Main Scenario:
      * Recruiter clicks Add Job Information
      * Recruiter clicks Add Keywords
      * Recruiter enters skills and qualifications.
      * Recruiter clicks submit.
      * Keywords are stored.

## Use Case #12

### Use Case Name: Close Jobs

* + - ID:# A12
    - Description: This use case involves the process of closing job postings by the recruiter once the positions have been filled or are no longer available.
    - Level: High
    - Primary Actor: Recruiter
    - Pre-condition: The recruiter must be logged into the system and have the necessary permissions to close the job posting(s).
    - Post-condition: The closed job postings are no longer visible to applicants and are marked as filled or closed in the system.
    - Main Scenario:

1. Recruiter logs into the system and navigates to the list of active job postings.
2. Recruiter selects the job postings they wish to close.
3. System prompts the recruiter to confirm the action.
4. Recruiter confirms the closure of the selected job posting(s).
5. System updates the status of the closed job postings, marking them as filled or closed.
6. Closed job postings are removed from the list of active job postings and are no longer visible to applicants.

## Use Case #13

### Use Case Name: Create Account

* + - ID:# A13
    - Description: Actors will create an account on our website.
    - Level: High
    - Primary Actor: Recruiter and Applicant
    - Pre-condition:
      * Recruiter must have access to platform’s registration page to enter first name, last name, email, strong password, phone number, company\_ID
      * Applicant must be on the website’s page to create account to enter first name, last name, email, strong password, phone number and upload pdf format file.
    - Post-condition:
      * Recruiter will have a recruiter account, opening them to access to other use cases like Add Job, Add Keywords, Update Job Information etc.
    - Main Scenario Recruiter :
      * Recruiter clicks Create Account
      * Recruiter enters first name, last name, email, strong password, phone number, and company\_ID
      * Recruiter hits the register button.
    - Main Scenario Applicant :
      * Applicant clicks Create Account
      * Applicant enters first name, last name, email, password, phone number and upload pdf format resume.
      * Applicant hits register button.

## Use Case #14

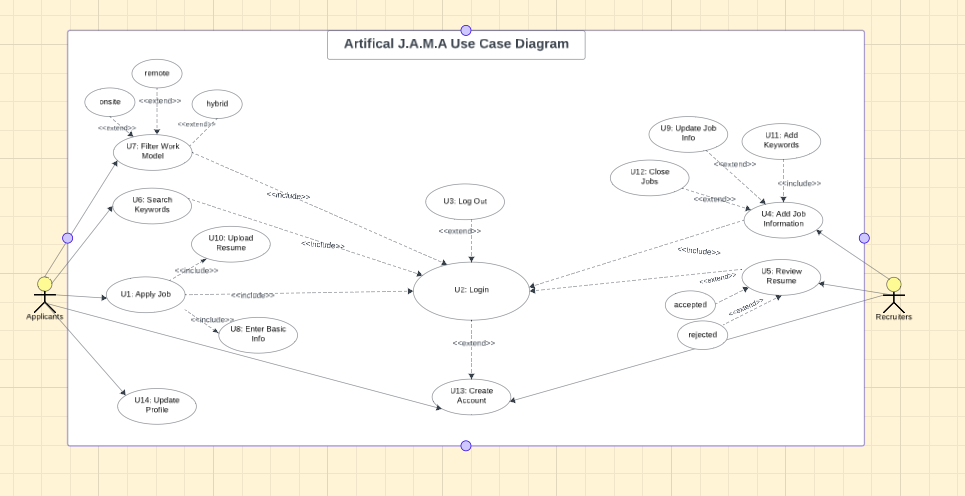
### Use Case Name: Update profile

* + - ID:# A14
    - Description: Applicant will be able to update their profile and any personal information.
    - Level: High
    - Primary Actor: Applicant
    - Pre-condition: applicant must have an account
    - Post-condition: N/A
    - Main Scenario Applicant:

Applicants will go to their profile.

1. Click on edit profile.
2. Click on email or phone number or profile picture to edit.
3. Click save and profile is updated.

The use case diagram is as shown below:



To explain the use case diagram for our AI Resume Parser in depth, there are two actors in our use case diagram. For the applicant, this actor has access to five use cases. The five use cases the applicant can do are log in, filter work model, search keywords, apply job and update profile. Applicants are **required** to log in before they can filter work models, search keywords, apply for jobs and update profiles. Login allows Applicant to access the other functions in the AI Resume System. Filter Work Model allows the applicant to filter the job postings through the nature of the job as in Remote, In-Person, and Hybrid. Search Keywords allows the applicants to search for job postings through the keywords they select. Apply Job allows applicants to enter basic info and upload resumes to set up a profile. After they select Apply Job, they are **required** to Enter Basic Info and Upload Resume. Enter Basic Info allows applicants to enter their basic information such as First Name, Last Name, Phone Number and Email. Upload Resume allows applicants to upload their resume. For the recruiter, this actor has access to eight use cases. The eight use cases the recruiter can do are login, log out, create account, add job information, add keywords, review resume, update job info, and close jobs. Recruiters are required to login before they can log out, add job information, add keywords, review resume, update job info, and close jobs. Login allows the recruiter to access the other functions in the AI Resume System. Log out allows the recruiter to log out of the system. This is **optional**. Create Account allows Recruiter to create an account on the AI Resume System and get access to functions within the system. It is **optional** to create an account, in the case that the recruiter already has an existing account. Add Job Information allows the Recruiter to add a job and its job details. This is **optional** right after login. It is **required** to Add Keywords. Recruiter also has the **option** to Update Job Information and Close Job. Review Resume allows the Recruiter to review successful candidates’ resumes. They can organize them in the folders Accepted or Rejected.

Below is code snippets of the mysql database table creation.

MYSQL Table Creation Code

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema ResumeParser

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `ResumeParser`;

CREATE SCHEMA IF NOT EXISTS `ResumeParser` DEFAULT CHARACTER SET utf8;

USE `ResumeParser`;

-- -----------------------------------------------------

-- Table `ResumeParser`.`Applicants`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `ResumeParser`.`Applicants`;

CREATE TABLE IF NOT EXISTS `ResumeParser`.`Applicants` (

`Applicant\_ID` INT NOT NULL AUTO\_INCREMENT,

`FirstName` VARCHAR(45) NOT NULL,

`LastName` VARCHAR(45) NOT NULL,

`Password` VARCHAR(45) NOT NULL,

`Email` VARCHAR(45) NOT NULL,

`PhoneNumber` VARCHAR(45) NOT NULL,

`ResumeFile` BLOB NOT NULL,

PRIMARY KEY (`Applicant\_ID`)

) ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `ResumeParser`.`Recruiters`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `ResumeParser`.`Recruiters`;

CREATE TABLE IF NOT EXISTS `ResumeParser`.`Recruiters` (

`Recruiter\_ID` INT NOT NULL AUTO\_INCREMENT,

`FirstName` VARCHAR(45) NOT NULL,

`LastName` VARCHAR(45) NOT NULL,

`Email` VARCHAR(45) NOT NULL,

`Password` VARCHAR(45) NOT NULL,

`PhoneNumber` VARCHAR(45) NULL,

`Company\_ID` INT NOT NULL,

PRIMARY KEY (`Recruiter\_ID`)

) ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `ResumeParser`.`Job`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `ResumeParser`.`Job`;

CREATE TABLE IF NOT EXISTS `ResumeParser`.`Job` (

`Job\_ID` INT NOT NULL AUTO\_INCREMENT,

`JobTitle` VARCHAR(50) NOT NULL,

`JobLogo` BLOB NOT NULL,

`JobDescription` VARCHAR(100) NOT NULL,

`JobQualifications` VARCHAR(100) NOT NULL,

`Keywords` VARCHAR(100) NOT NULL,

`JobLocation` VARCHAR(100) NOT NULL,

`Filter` VARCHAR(100) NOT NULL,

PRIMARY KEY (`Job\_ID`)

) ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `ResumeParser`.`Resume`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `ResumeParser`.`Resume`;

CREATE TABLE IF NOT EXISTS `ResumeParser`.`Resume` (

`Resume\_ID` INT NOT NULL AUTO\_INCREMENT,

`JobTitle` VARCHAR(50) NOT NULL,

`Job\_ID` INT NOT NULL,

`Applicant\_ID` INT NOT NULL,

PRIMARY KEY (`Resume\_ID`),

FOREIGN KEY (`Job\_ID`) REFERENCES `ResumeParser`.`Job` (`Job\_ID`),

FOREIGN KEY (`Applicant\_ID`) REFERENCES `ResumeParser`.`Applicants` (`Applicant\_ID`)

) ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

MYSQL Data Insertion Statement Code

-- -----------------------------------------------------

-- INSERT Statements

-- -----------------------------------------------------

INSERT INTO Applicants VALUES

(DEFAULT, "Alice", "Jones", "ajones@hotmail.com","908-111-1111","/drive/AJResume.pdf"),

(DEFAULT, "Ray", "Smith", "crazyace@gmail.com","908-222-2222","/drive/myResume.pdf"),

(DEFAULT, "Rico", "Sanchez", "sanchez\_rico@msn.com","908-333-3333","/drive/Resume.pdf")

INSERT INTO Recruiters VALUES

(DEFAULT, "Greg", "Heff", "gheff@hotmail.com","ice\_cream@@","908-444-4444","123"),

(DEFAULT, "Bella", "Moon", "moon@yahoo.com","rice\_cakes","908-555-5555","808"),

(DEFAULT, "Yoly", "Eclair", "monsta@gmail.com","smile","908-666-6666","323")

INSERT INTO Job VALUES

(DEFAULT, "Summer Intern",img,"This job consists...","php,bachelor's degree ...","Community, HTML, Future","123 Union Ave",”Remote”),

(DEFAULT, "Full Stack Job",img,"This job consists...","masters degree ...","Node.js,Agile,Google Cloud",,"321 Clark Drive","Hybrid"),

(DEFAULT, "Sales Associate",img,"This job consists...","associates degree ...","Customer Service, Passion, Lead","1000 Michael St.","Onsite")

INSERT INTO Resume VALUES

(DEFAULT,” Full Stack Job”, 123, “002”,

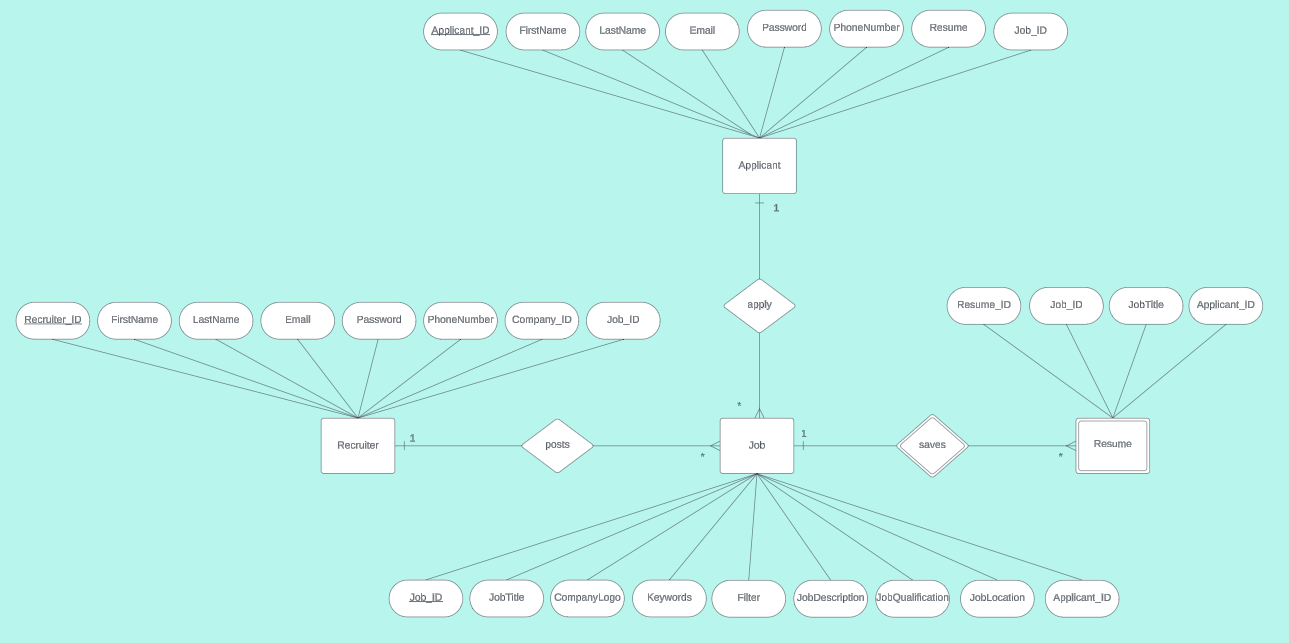
Entity Relationship Diagram

In terms of how our application’s infrastructure will consist of, we decided to use a relational database called MySQL. Our team came up with an entity relationship diagram. Before we designed our entity relationship diagram we laid out our entities to be able to reference when creating our ER diagram. Our entities below lay out the table name, attributes, primary key and foreign key.

* Applicants
  + Applicant\_ID (PK) auto increment
  + FirstName
  + LastName
  + Email
  + Password
  + PhoneNumber
  + Resume
  + Job\_ID (FK)
* Recruiters
* Recruiter\_ID (PK) auto increment
* FirstName (varchar)
* LastName (varchar)
* Email (varchar)
* Password (varchar)
* PhoneNumber (varchar)
* Company\_ID (integer)
* Job\_ID (FK)
* Job
* Job\_ID (PK)
* JobTitle
* CompanyLogo (FK)
* JobDescription
* JobQualification
* Keywords
* JobLocation
* Filter
* Applicant\_ID (FK)
* Resume
  + Resume\_ID(PK)
  + Job\_ID(FK)
  + JobTitle
  + Applicant\_ID(FK)

The entity relationship diagram below is referenced from the layout we

used above.



To explain the ER diagram for our AI Resume Parser application in depth, we will start by explaining our entities. We have 3 strong entities, 1 weak entity, 2 strong relationships and 1 weak relationship. An applicant has a strong relationship with the job. An applicant can apply to many jobs while a job posting can be applied to by an applicant. A recruiter and job have a strong relationship. A recruiter can post many job postings while a job posting can be posted one at a time by a recruiter. Job and Resume have weak relationships meaning that we need a job posting in order to save a resume. A job posting can save many resumes while resumes are saved per job posting. The primary key for applicants is Applicant\_ID. The primary key for Recruiter\_ID. The primary key for Job is Job\_ID. The composite key(primary key) for the Resume table cannot have a primary key on it’s own since it’s a weak entity therefore we created a composite key for Resume to take the primary key from Job\_ID and combine it with the identifier for the Resume table which is Resume\_ID. To conclude, the composite primary key for the Resume table is Job\_ID, Resume\_ID.

In Phase 5 Implementation, several modifications were necessitated and implemented, following recommendations provided by Joshua. Primarily, a new table, denoted as AppliedJobsApplicant, was incorporated. This table serves as a pivotal linkage point among various existing tables within the database schema, notably Applicant, Recruiter, Job, and Resume.

CREATE TABLE `AppliedJobsApplicant` (

`AppliedJobs\_ID` int(11) NOT NULL AUTO\_INCREMENT,

`Job\_ID` int(11) NOT NULL,

`Job\_Title` varchar(50) NOT NULL,

`CompanyName` varchar(100) NOT NULL,

`Applicant\_ID` int(11) NOT NULL,

`ApplicantName` varchar(45) NOT NULL,

`LastName` varchar(45) NOT NULL,

`Email` varchar(45) NOT NULL,

`PhoneNumber` varchar(45) NOT NULL,

`Resume` blob NOT NULL,

`Recruiter\_ID` int(11) NOT NULL,

`Status` varchar(20) DEFAULT 'In Progress',

PRIMARY KEY (`AppliedJobs\_ID`),

KEY `fk\_AppliedJobs\_Job` (`Job\_ID`),

KEY `fk\_AppliedJobs\_Applicant` (`Applicant\_ID`),

KEY `fk\_AppliedJobs\_Recruiter` (`Recruiter\_ID`),

CONSTRAINT `fk\_AppliedJobs\_Applicant` FOREIGN KEY (`Applicant\_ID`) REFERENCES `Applicants` (`Applicant\_ID`),

CONSTRAINT `fk\_AppliedJobs\_Job` FOREIGN KEY (`Job\_ID`) REFERENCES `Job` (`Job\_ID`),

CONSTRAINT `fk\_AppliedJobs\_Recruiter` FOREIGN KEY (`Recruiter\_ID`) REFERENCES `Recruiters` (`Recruiter\_ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=55 DEFAULT CHARSET=latin1

After altering the tables, these are now the current MySQL tables:

CREATE TABLE `Applicants` (

`Applicant\_ID` int(11) NOT NULL AUTO\_INCREMENT,

`FirstName` varchar(45) NOT NULL,

`LastName` varchar(45) NOT NULL,

`Password` varchar(45) NOT NULL,

`Email` varchar(45) NOT NULL,

`PhoneNumber` varchar(45) NOT NULL,

`ResumeFile` longblob,

PRIMARY KEY (`Applicant\_ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=47 DEFAULT CHARSET=latin1

| CREATE TABLE `Recruiters` ( `Recruiter\_ID` int(11) NOT NULL AUTO\_INCREMENT, `FirstName` varchar(45) NOT NULL, `LastName` varchar(45) NOT NULL, `Email` varchar(45) NOT NULL, `Password` varchar(45) NOT NULL, `PhoneNumber` varchar(45) DEFAULT NULL, `Company\_ID` int(11) NOT NULL, PRIMARY KEY (`Recruiter\_ID`) ) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=latin1 |
| --- |
|  |

| CREATE TABLE `Job` ( `Job\_ID` int(11) NOT NULL AUTO\_INCREMENT, `JobTitle` varchar(255) DEFAULT NULL, `JobLogo` longblob, `JobDescription` varchar(255) DEFAULT NULL, `JobQualifications` varchar(255) DEFAULT NULL, `Keywords` varchar(100) NOT NULL, `JobLocation` varchar(100) NOT NULL, `Filter` varchar(100) NOT NULL, `CompanyName` varchar(100) NOT NULL, `Recruiter\_ID` int(11) DEFAULT NULL, PRIMARY KEY (`Job\_ID`), KEY `fk\_Recruiter\_ID` (`Recruiter\_ID`), CONSTRAINT `fk\_Recruiter\_ID` FOREIGN KEY (`Recruiter\_ID`) REFERENCES `Recruiters` (`Recruiter\_ID`) ) ENGINE=InnoDB AUTO\_INCREMENT=12 DEFAULT CHARSET=latin1 |
| --- |
|  |

| CREATE TABLE `AppliedJobsApplicant` ( `AppliedJobs\_ID` int(11) NOT NULL AUTO\_INCREMENT, `Job\_ID` int(11) NOT NULL, `Job\_Title` varchar(50) NOT NULL, `CompanyName` varchar(100) NOT NULL, `Applicant\_ID` int(11) NOT NULL, `ApplicantName` varchar(45) NOT NULL, `LastName` varchar(45) NOT NULL, `Email` varchar(45) NOT NULL, `PhoneNumber` varchar(45) NOT NULL, `Resume` blob NOT NULL, `Recruiter\_ID` int(11) NOT NULL, `Status` enum('In Progress','Accepted','Rejected') DEFAULT 'In Progress', PRIMARY KEY (`AppliedJobs\_ID`), KEY `fk\_AppliedJobs\_Job` (`Job\_ID`), KEY `fk\_AppliedJobs\_Applicant` (`Applicant\_ID`), KEY `fk\_AppliedJobs\_Recruiter` (`Recruiter\_ID`), CONSTRAINT `fk\_AppliedJobs\_Applicant` FOREIGN KEY (`Applicant\_ID`) REFERENCES `Applicants` (`Applicant\_ID`), CONSTRAINT `fk\_AppliedJobs\_Job` FOREIGN KEY (`Job\_ID`) REFERENCES `Job` (`Job\_ID`), CONSTRAINT `fk\_AppliedJobs\_Recruiter` FOREIGN KEY (`Recruiter\_ID`) REFERENCES `Recruiters` (`Recruiter\_ID`) ) ENGINE=InnoDB AUTO\_INCREMENT=56 DEFAULT CHARSET=latin1 |
| --- |

| CREATE TABLE `Resume` ( `Resume\_ID` int(11) NOT NULL AUTO\_INCREMENT, `JobTitle` varchar(50) DEFAULT NULL, `Job\_ID` int(11) NOT NULL, `Applicant\_ID` int(11) NOT NULL, `Status` enum('In Progress','Accepted','Rejected') DEFAULT 'In Progress', PRIMARY KEY (`Resume\_ID`), KEY `fk\_Resume\_Job` (`Job\_ID`), KEY `fk\_Resume\_Applicants` (`Applicant\_ID`), CONSTRAINT `fk\_Resume\_Applicants` FOREIGN KEY (`Applicant\_ID`) REFERENCES `Applicants` (`Applicant\_ID`), CONSTRAINT `fk\_Resume\_Job` FOREIGN KEY (`Job\_ID`) REFERENCES `Job` (`Job\_ID`), CONSTRAINT `Resume\_ibfk\_1` FOREIGN KEY (`Job\_ID`) REFERENCES `Job` (`Job\_ID`), CONSTRAINT `Resume\_ibfk\_2` FOREIGN KEY (`Applicant\_ID`) REFERENCES `Applicants` (`Applicant\_ID`) ) ENGINE=InnoDB AUTO\_INCREMENT=22 DEFAULT CHARSET=latin1 |
| --- |

1. Challenges and Takeaway

Adriana

The challenges I faced with the team at the beginning of the project were how to guide ourselves in terms of starting, documentation, and how development would work. To start, during our first team meeting, I had technical difficulties with our first established communication channel, and we switched to another platform. The challenges I faced with the team at the beginning of the project were valuable learning experiences for both me and the team. We learned the importance of effective communication channels and adaptability when faced with technical difficulties. Switching to another platform for communication taught us to be flexible and proactive in finding solutions to problems.

Another issue we encountered as a team was figuring out how we could reduce AI bias when we start our AI implementation. Regarding AI bias reduction, we learned the significance of addressing ethical considerations early in the project. This challenge prompted us to explore strategies for mitigating bias in AI implementations, enhancing our understanding of ethical AI practices.

Some more challenges we faced as a team were being unsure about one of the use cases we came up with during Phase 3 Analysis. We were uncertain because the use case was more of an internal system use case. For example, we had a use case for AI giving feedback to the applicant; however, we were uncertain if this should be a use case or not because there’s no specific button in our system that will do this. The feedback would occur automatically when the applicant uploads their resume. Uncertainty surrounding use cases during Phase 3 Analysis taught us the importance of clarity and specificity in defining project requirements. It underscored the need for thorough analysis and consensus-building within the team to ensure alignment on project objectives.

Our next challenge was issues with identifying tables and primary/foreign keys for our database design. With this challenge, we decided to talk as a team and reference our prior database management knowledge and also researched to determine what necessary database tables were needed. Issues with database design highlighted the importance of leveraging collective knowledge and conducting thorough research. Collaborative problem-solving and referencing prior knowledge enabled us to overcome challenges in identifying tables and primary/foreign keys.

My next challenge regarding front-end development was having creativity issues when coming up with the design mockups for our web interface designs in Phase 4 Design. By collaborating with another peer, Joshua, we were able to exchange ideas through our initial ideas and then improve our mockups from the initial designs. Creativity issues during front-end development emphasized the value of collaboration and peer feedback. Working with my colleague Joshua allowed us to exchange ideas, iterate on designs, and ultimately enhance the quality of our mockups.

The next challenge we faced as a team in Week 5 when we were starting our Phase 5 Implementation/Development. We were trying to figure out how to get the database tables that were created in Phase 4 Design on everyone’s laptop, but we decided to break up the work into front-end development and back-end development; therefore, only two people working on the backend development would need the database tables on their laptop. We resolved this issue by figuring out it would be redundant to have everyone have the database tables since we split the work into front and back-end development. Managing the distribution of workload between front-end and back-end development taught us the importance of efficient resource allocation. We realized the benefits of breaking down tasks and optimizing resource utilization to streamline project progress.

The next challenge in Week 7 for the front-end development part for the applicant homepage was integrating the Bootstrap accordion into our website and making it work. What I did was take the Bootstrap code from the Bootstrap documentation online and then customize it to add an apply button and make sure it fits inside our website. This was a challenge because the accordion code from Bootstrap doesn’t know how to fit on the website, so I had to make sure it fits centered on the page and has the content that we wanted it to have for the applicant homepage. To add onto the challenges for Week 7, changing and adding code to make the accordion work, the navbar would get messed up, so there was lots of debugging and fixing my code to make sure this issue was resolved. Integrating Bootstrap components into the website showcased the significance of persistence and attention to detail in web development. Debugging issues with the navigation bar underscored the importance of thorough testing and iterative refinement of code. Lastly, for Week 7, we also had a team challenge of having issues with integrating PHP into the applicant and recruiter homepage but was resolved. Our HomePage front-end accordion did end up getting integrated with PHP database development. Challenges with integrating PHP into homepage development highlighted the importance of comprehensive understanding and troubleshooting of technology stacks. Overcoming these challenges reinforced our knowledge of web development frameworks and languages.

For Week 8, we were facing the challenge of figuring out how to store the PDF of a resume into the MySQL database and how we would be able to see it on our side. The difficulty in storing PDF resumes in the database prompted us to deepen our understanding of database management and file storage mechanisms. It underscored the importance of robust data handling practices and prompted us to explore alternative solutions.

For Week 9, we were facing the challenge of attempting to accept an applicant and add them to the resume table, but this was not fully working yet. This not working led to us still not figuring out how to store the PDF of a resume into the MySQL database and be able to see it on our side.

For Week 10, we faced the challenge of attempting to fetch the data and output it on the recruiter’s update page and being able to update the job. We still continue to face issues with storing the PDF. Struggles with data fetching and updating processes underscored the complexity of backend development tasks. Overcoming these challenges enhanced our proficiency in database querying and data manipulation, facilitating smoother project implementation.

Overall, each challenge presented opportunities for learning and growth, reinforcing our teamwork, problem-solving skills, and technical capabilities throughout the project lifecycle.

Anya

During the backend development of the Applicant pages, I encountered several challenges stemming from my limited experience with PHP. However, I was fortunate to have support from my team members who had prior experience in web development using PHP, and I also supplemented my knowledge through a database course I was concurrently taking. One significant difficulty was implementing a dynamic search bar in PHP, particularly due to the complexity of creating prepared statements dynamically.

// Handling form submission

if ($\_SERVER["REQUEST\_METHOD"] == "GET") {

// SQL query to select job data from the database

$sql = "SELECT Job\_ID,JobTitle, JobDescription, JobQualifications, JobLocation, Filter, CompanyName,Recruiter\_ID FROM Job";

// Sanitize and validate form data

$filter1 = isset($\_GET["filter1"]) ? sanitize\_input($\_GET["filter1"]) : null;

$filter2 = isset($\_GET["filter2"]) ? sanitize\_input($\_GET["filter2"]) : null;

$filter3 = isset($\_GET["filter3"]) ? sanitize\_input($\_GET["filter3"]) : null;

$searchKeyword = isset($\_GET["search"]) ? sanitize\_input($\_GET["search"]) : null;

// Check if search keyword is provided

if (!empty($searchKeyword)) {

// If search keyword is not empty, filter based on it

$searchKeyword = '%' . $searchKeyword . '%';

$sql .= " WHERE JobTitle LIKE ?";

} else {

// If search keyword is not provided, show all jobs

$searchKeyword = "%"; // Match all job titles

#echo 'search keyword is ' . $searchKeyword;

}

// Check if filter is provided

$filter\_array = array($filter1, $filter2, $filter3);

$checked\_array = [];

foreach ($filter\_array as $e) {

if (!is\_null($e) && !empty($e)) {

$checked\_array[] = $e;

#echo 'added' . $e;

}

}

//print\_r($checked\_array);

// If both search keyword and some filter values are given

if ($searchKeyword!='%' && count($checked\_array) > 0) {

$sql .= " AND Filter IN (";

$placeholders = implode(",", array\_fill(0, count($checked\_array), "?"));

$sql .= $placeholders;

$sql .= ")";

} elseif ($searchKeyword=='%' && count($checked\_array) > 0) {

$sql .= " WHERE Filter IN (";

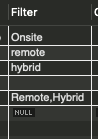
$placeholders = implode(",", array\_fill(0, count($checked\_array), "?"));

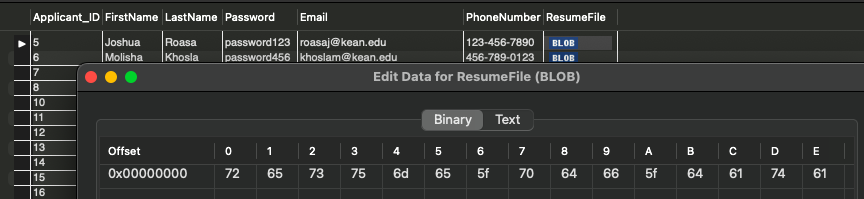
$sql .= $placeholders;

$sql .= ")";

}

Additionally, adapting to changes in the project structure posed ongoing challenges, especially concerning the evolution of our filter system. Originally, our Job table filter column only accommodated onsite, hybrid, or remote options, but as our project progressed, we needed to support multiple filter selections.



Debugging PHP code was another major hurdle, exacerbated by the necessity of safeguarding sensitive user data used for authentication. ChatGPT proved to be a valuable resource in overcoming some of these hurdles. Another obstacle was figuring out how to store PDF files in our MYSQL Database. Initially, I attempted to store PDFs in a format that could be directly viewed from the database, akin to an image file. However, MYSQL stores PDFs as binary data, making us unable to recover it as a PDF .

Molisha

The main hurdle I faced was when I got stuck dealing with the view jobs row, especially when it needed to work dynamically. But that wasn't the only challenge. Each problem I faced taught me something new about HTML5 and CSS3, showing me the importance of always learning and keeping up with what's happening in web design.

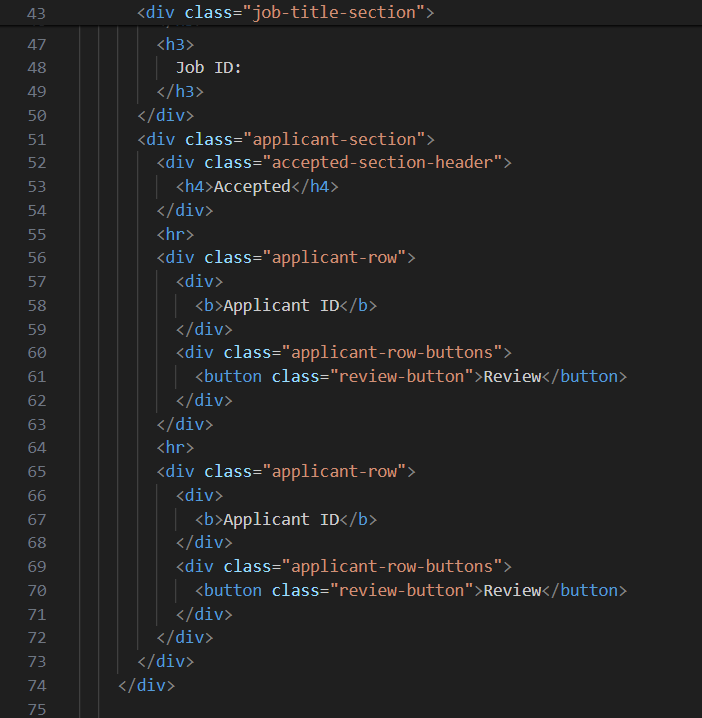
I learned much by working on all these things. I designed the pages to work dynamically and learned to handle a lot of CSS classes as well. Learned to avoid repeating the code. Meeting a tight deadline for designing all front-end pages underscored the importance of efficient time management and prioritization to deliver high-quality work on schedule.

Utilizing HTML5 and CSS3, I designed every page from the ground up, covering a range of essential pages including the home page, register page, review pages for applicants and applicant information, review accepted applications, review applicant page, view job page, update job page, and add job pages for recruiters. Despite the daunting deadline, I tackled the challenge with determination, ensuring timely delivery of all front-end pages.

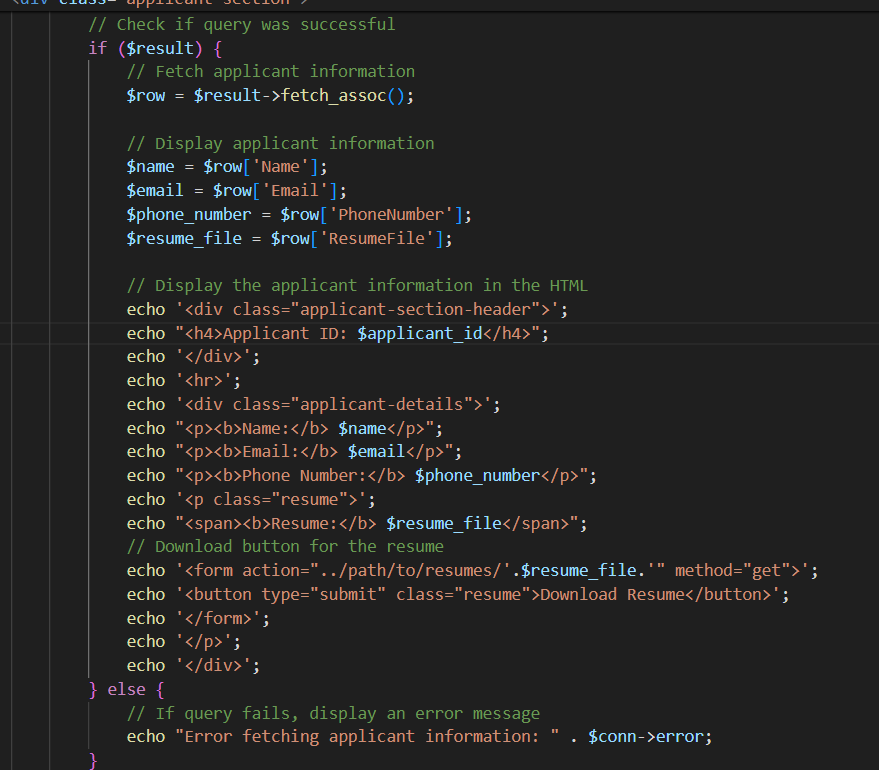
Joshua

The first challenge we encountered was determining how to access the same database. Initially, our plan involved using XAMPP or AMP to store all the tables, with each of us updating changes on our respective laptops. However, recalling the advice of Dr. Huang, my professor in Database Management during the Fall 2023 semester, I reached out to him to inquire about creating a database for us using obi2. Following my email, Dr. Huang graciously assisted us in setting up our server. Since then, we've been utilizing the obi2 web server and database server to host our website live and store the database. This arrangement enables each team member to access a unified database, thereby saving time.

The second challenge I encountered involved converting all the HTML and CSS that my partner, Molisha, had created for our tasks on the Recruiter Side Pages. Molisha would design the HTML and CSS components initially, and then I would convert them into PHP and ensure their functionality. This task presented a new experience for me, as in my database class, we were primarily tasked with retrieving information from databases and displaying it in simple HTML tables without any CSS or additional styling. To overcome this challenge, I researched examples online and consulted with ChatGPT, ultimately finding solutions to implement the necessary changes.

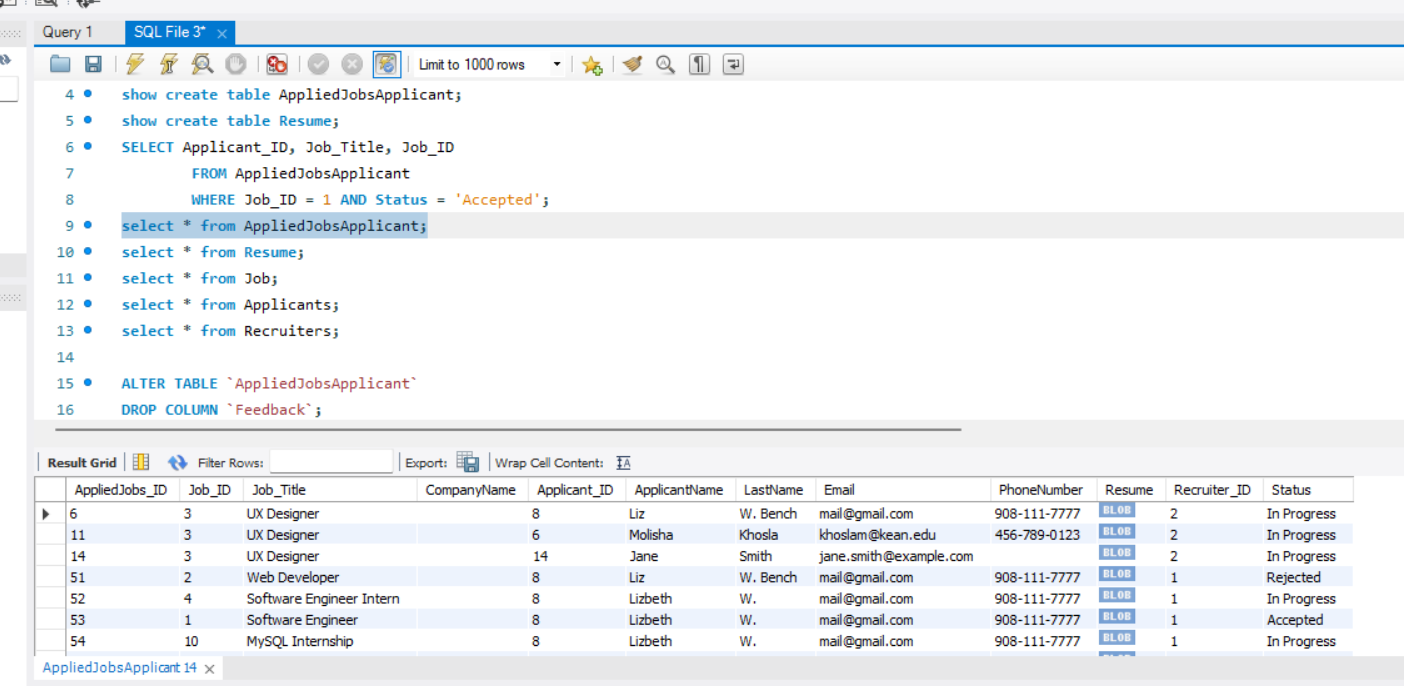


to php file.

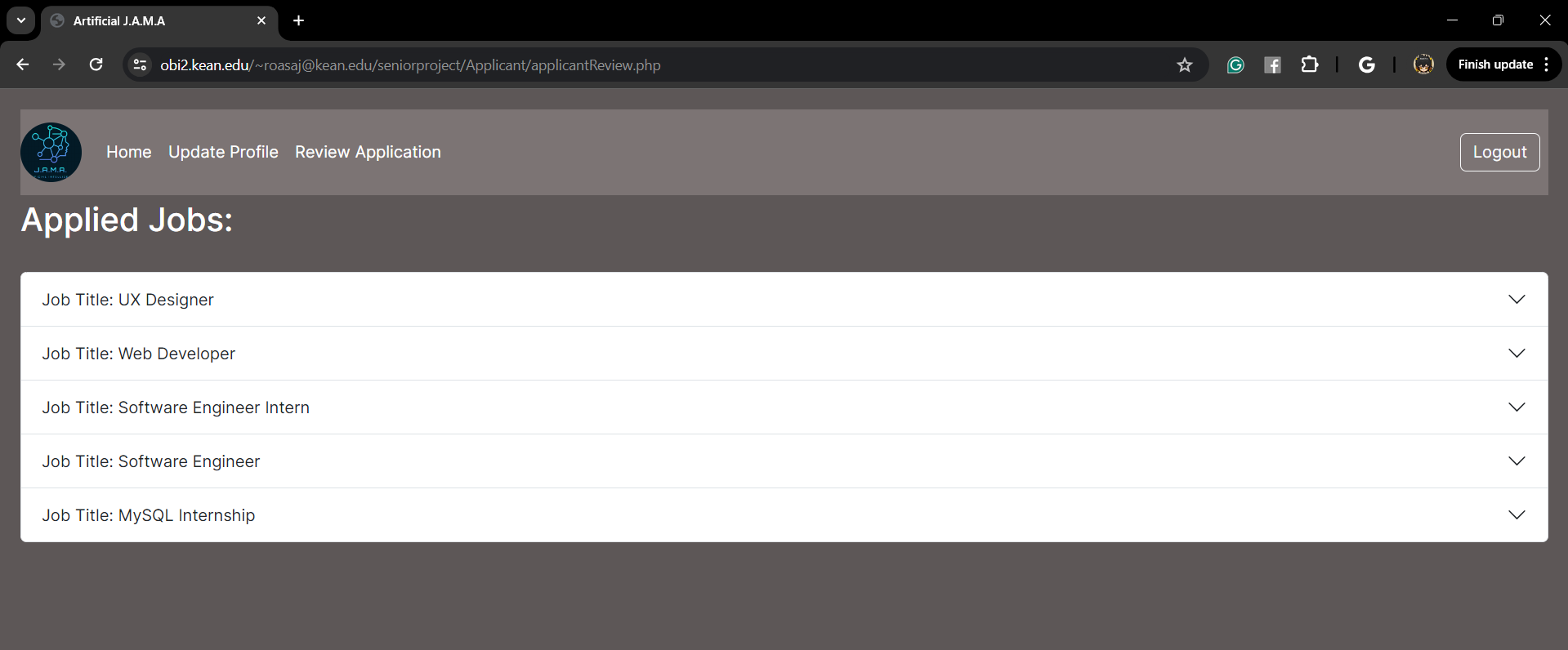


The third challenge I encountered arose during the creation of the backend. It dawned on me that we lacked a centralized table connected to every other table. Specifically, we faced the question of where to store all the Applicant Applications. Initially, we intended to house this information within the Job Application table. However, I soon realized that this approach would not only clutter the Job table with redundant information but also impede efficient data retrieval for Job-specific values.

To address this issue, I devised a solution by introducing a new table: AppliedJobApplications. This table serves the crucial function of storing all Applicant applications that are still in progress. It acts as a hub, facilitating the showcasing of Applicants who have applied for specific Jobs. Furthermore, it provides essential feedback to Applicants regarding the status of their applications, whether they are In Progress, Accepted, or Rejected. Essentially, this table serves as the linchpin connecting all our other tables, streamlining data management and enhancing accessibility.



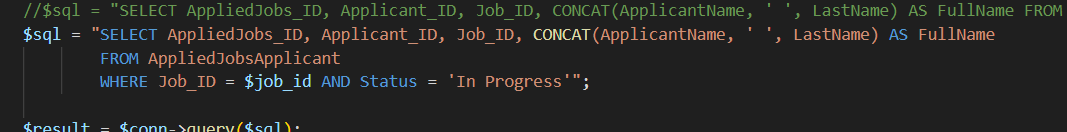
Review Application Page:



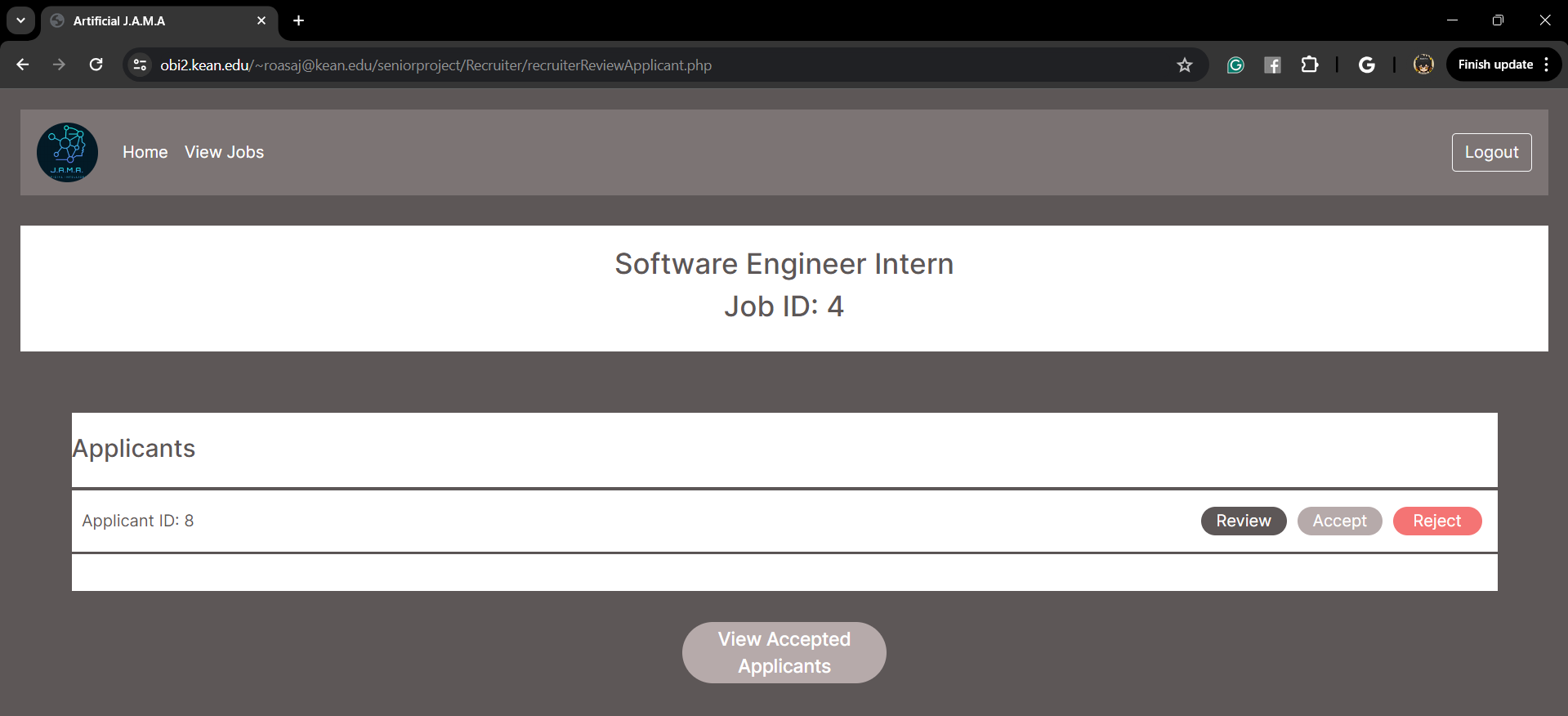
The final challenge I encountered stemmed from the initial setup of AppliedJobApplicant, which did not include a Status field (In Progress, Accepted, Rejected). Previously, when a Recruiter accepted or rejected an Applicant's application, the data would be deleted from the AppliedJobApplicant table. However, upon collaboration with Anya, we recognized that deleting these Applicant values from the table would also remove them from the Review Application page on the Applicant side, which showcases the Jobs the Applicant has applied for along with feedback.

To resolve this issue, I made adjustments to the AppliedJobApplication table by adding a new column named Status, with a default value of In Progress and options for Accepted and Rejected. Additionally, modifications were made to the recruiterReviewApplicant.php page, which serves as the platform for Recruiters to review all Applicant Applications. Here, I implemented a WHERE condition to only display applications with a Status of In Progress, ensuring that only relevant applications are showcased.

Solution:

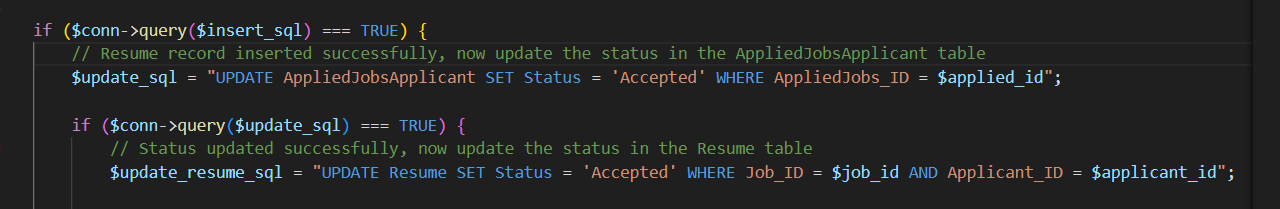


recruiterReviewApplicant.php

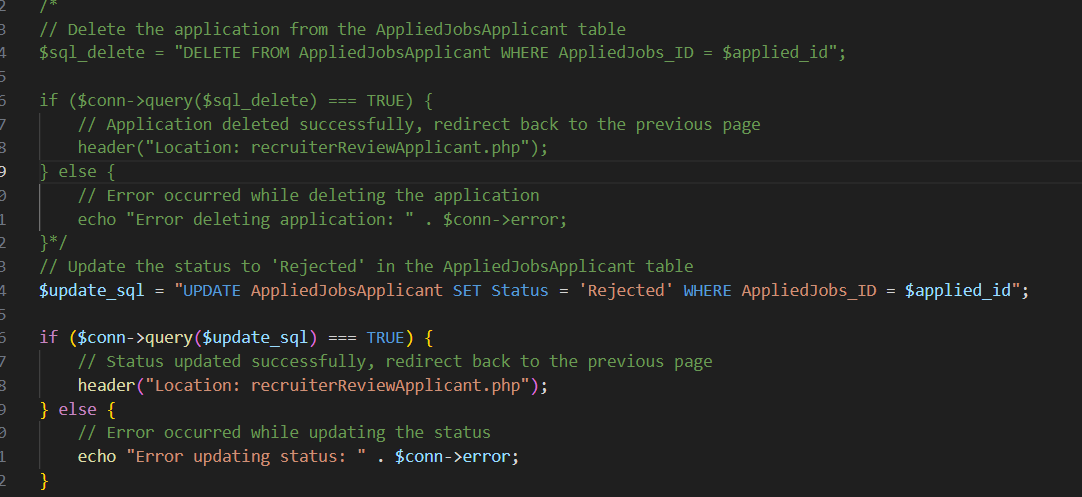


Another significant change I implemented was altering the method of handling Applicant Applications within the appliedJobsApplicant table. Instead of outright deleting the Applicant Application upon acceptance or rejection, I modified the statement to perform an update operation instead. This adjustment ensures that the historical record of Applicant Applications remains intact within the table, providing a comprehensive view of all past interactions and decisions.

acceptedApplicant.php



rejectApplicant.php



As a team, we faced several challenges for which we were unable to find immediate solutions. Two notable obstacles included the inability to upload logos for Jobs and enable Recruiters to download the resume PDFs uploaded by Applicants. Additionally, we ran out of time to implement the AI Resume parser itself. These challenges remain unresolved as we continue to explore potential solutions and allocate resources accordingly.

My biggest takeaway from this project is realizing the significance of teamwork. I'm sincerely grateful for every team member who approached their tasks with enthusiasm and dedication. Furthermore, this project provided a valuable opportunity to apply the knowledge I accumulated from various computer science courses here at Kean University.

1. Future work

In our roadmap for future enhancements, we will prioritize several key areas to elevate the project's capabilities. Firstly, we will embark on refining the user experience by transitioning the multi-value filter into an intuitive table format, ensuring seamless navigation for both recruiters and applicants. Consistency in code implementation across all aspects of the platform will be a cornerstone of our development efforts, facilitating easier maintenance and scalability as the project evolves.

A significant advancement on our horizon involves optimizing the storage and retrieval of PDFs and DOCX files within the database. This initiative will not only improve system performance but also lay the groundwork for future enhancements in data management and analysis.

Our vision for the Recruiter side encompasses cutting-edge AI-driven features. We will harness the power of AI to analyze resumes based on specialized keywords, empowering recruiters with actionable insights to make informed decisions swiftly and accurately. Additionally, implementing AI-driven feedback mechanisms will provide real-time guidance to recruiters, enhancing the efficiency of the hiring process.

Looking ahead, our commitment to innovation extends to the application side, where we aim to seamlessly integrate the AI Resume Parser directly into webpages, offering a frictionless user experience. Addressing the challenge with footers not adhering to desired CSS standards will ensure a polished and cohesive interface, enhancing user engagement. Moreover, we will focus on making the website responsive, enabling users to access and utilize the platform seamlessly across various devices, including mobile phones and tablets.

Furthermore, enhancing recruiter professionalism through seamless logo upload and retrieval will be a priority. Streamlining the process of adding job descriptions and postings will contribute to a more efficient recruitment workflow, benefiting recruiters and applicants alike.

As we continue to invest in the project, increased resources will unlock opportunities for further advancements. Advanced analytics capabilities will enable deeper insights into candidate profiles and recruitment trends, empowering stakeholders with data-driven decision-making tools. With our collective commitment to innovation, the future of the project is bright, poised to revolutionize the recruitment landscape.

1. Additional Notes

All files: Including Phases, Videos, Zip file code.

<https://drive.google.com/drive/u/3/folders/103ijrckpW5Vm50xFYVkoMLqErkk9Q42Y>

1. References

[1] Bootstrap. 2023. Accordion. Bootstrap. https://getbootstrap.com/docs/5.3/components/accordion/

[2] W3Schools. n.d. <button> Tag. W3Schools. https://www.w3schools.com/tags/tag\_button.asp

[3] W3Schools. n.d. PHP MySQL Prepared Statements. W3Schools. https://www.w3schools.com/php/php\_mysql\_prepared\_statements.asp

[4] W3Schools. n.d. PHP Tutorial. W3Schools. https://www.w3schools.com/php/default.asp

[5] W3Schools. n.d. JavaScript AJAX Introduction. W3Schools. https://www.w3schools.com/js/js\_ajax\_intro.asp

[6] Lucidchart. n.d. Lucidchart. Lucidchart. https://lucid.app/lucidchart/d10e90bc-72bb-4573-94bb-eb8e427a059d/edit?invitationId=inv\_4c15306a-5044-4ad7-ab98-cd2936377d39&page=0\_0#

[7] Figma. n.d. Figma. Figma. https://www.figma.com/file/vvYq4GSc7AHTakg3Obj4By/A.I-J.A.M.A?type=design&node-id=0-1&mode=design&t=LyU1YjhtwYpI6A3T-0

[8] ChatGPT. n.d. ChatGPT. ChatGPT. https://chatgpt.com/

[9] Programming with Mosh. 2018. Learn JavaScript in One Hour. YouTube. https://www.youtube.com/watch?v=8fVEMdHKmqM

[10] Bogen, M. 2019. All the Ways Hiring Algorithms Can Introduce Bias. Retrieved from https://hbr.org/2019/05/all-the-ways-hiring-algorithms-can-introduce-bias on April 16, 2024.