# Project Report: PathFinder - An AI-Powered Career Recommendation System (Iteration 1)

**Course:** DTSC 4565 - Software Engineering

**Institution:** College of Information, University of North Texas

**Submission Date:** October 23, 2025

**Team Members:**

* Michelle Dogo (Frontend Development & UI/UX)
* Spoorthi Hassan Sathyanarayana (Backend & Dashboard Design)
* Teja Sai Srinivas Kunisetty **(Team Lead)** - Machine Learning Specialist
* Murari Nalabothu - Database/DevOps Specialist

Individual Contributions Breakdown

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task/Component | Michelle Dogo | Spoorthi Hassan | Teja Sai K | Murari N |
| Requirement Analsyis | ✓ |  |  | ✓ |
| Use Case Development | ✓ | ✓ | ✓ |  |
| Domain Modeling |  | ✓ |  | ✓ |
| UI/UX Design | Lead | Support | Support |  |
| Frontend Development | Lead | Support | Support |  |
| Backend Development |  | Lead | Support |  |
| ML Module Development |  |  | Lead |  |
| Database Design |  |  |  | Lead |
| Testing & QA | ✓ | ✓ | ✓ |  |
| Documentation | ✓ |  |  | ✓ |

**Responsibility Allocation Chart**

**Michelle Dogo (25%)**

* User-friendly form design and implementation
* Student dashboard interface development
* Frontend JavaScript implementation
* User experience optimization
* Documentation of UI components

**Spoorthi Hassan Sathyanarayana (25%)**

* Backend API development using Flask
* Database schema design and implementation
* Employer and Career Advisor dashboard design
* Role-based access control implementation
* System integration and testing

**Teja Sai Srinivas Kunisetty (25%)**

* Recommendation algorithm development
* Skill gap analysis implementation
* Model training and validation
* Integration with backend services
* Performance optimization of ML components

**Database/DevOps Specialist (25%)**

* Database optimization and management
* Notification service implementation
* System deployment and maintenance
* Security implementation
* Performance monitoring

Contents

[Project Report: PathFinder - An AI-Powered Career Recommendation System (Iteration 1) 1](#_Toc212078268)

[1. Customer Statement of Requirements 2](#_Toc212078269)

[1.1 Problem Domain and Opportunity 2](#_Toc212078270)

[1.2 Stakeholder Goals and Perspectives 2](#_Toc212078271)

[1.3 System Vision and Scope (Iteration 1) 3](#_Toc212078272)

[1.4 Illustrative Use Case Scenario: The Student's Journey 4](#_Toc212078273)

[2. Functional and Non-Functional Requirements Specification 5](#_Toc212078274)

[2.1 Functional Requirements 5](#_Toc212078275)

[2.2 Non-Functional Requirements (NFRs) 7](#_Toc212078276)

[3. System Sequence Diagrams (SSDs) 9](#_Toc212078277)

[3.1 Overview of System Interactions 10](#_Toc212078278)

[3.2 Sequence and Activity Diagram of Student 10](#_Toc212078279)

[3.3 Sequence and Activity Diagram of Employer 11](#_Toc212078280)

3.4 Sequence and Activity Diagram of Advisor………………………………………………………..

3.5 Sequence and Activity Diagram of Administrator……………………………………………

3.6 Sequence Diagram and Use case Diagram for the system

[4. Domain Analysis and Domain Model 13](#_Toc212078281)

[4.1 Identification of Key Domain Concepts 13](#_Toc212078282)

[4.2 Conceptual Domain Model Diagram 14](#_Toc212078283)

[4.3 Data Dictionary 15](#_Toc212078284)

[5. User Interface Design 18](#_Toc212078285)

[5.1 UI/UX Philosophy 18](#_Toc212078286)

[5.2 Student Dashboard Wireframe Concept 18](#_Toc212078287)

[5.3 Employer Dashboard Wireframe Concept 19](#_Toc212078288)

[5.4 Career Advisor Dashboard Wireframe Concept 19](#_Toc212078289)

[6. Class Diagram and Interface Specification 20](#_Toc212078290)

[6.1 High-Level Software Architecture 20](#_Toc212078291)

[6.2 System Design Class Diagram 22](#_Toc212078292)

[6.3 Interface Specifications 22](#_Toc212078293)

Reference

## 1. Customer Statement of Requirements

This section outlines the foundational requirements for the PathFinder system. It defines the problem domain, identifies key stakeholders and their objectives, establishes the system's vision and scope for the initial development iteration, and presents an illustrative use case to contextualize the system's intended functionality.

### 1.1 Problem Domain and Opportunity

In the contemporary professional landscape, students, graduates, and career changers face a significant challenge: navigating an overwhelming number of career options within a rapidly evolving job market.1 The process of aligning personal skills, educational background, and intrinsic interests with viable and fulfilling career paths is often fraught with uncertainty and a lack of clear, data-driven guidance. This challenge is exemplified by the common scenario of a final-year student from the College of Information at the University of North Texas, perhaps majoring in Information Science or Data Science, who is unsure whether to pursue a career as a software developer, a data scientist, or a cybersecurity expert, despite possessing a strong technical foundation.2 The core of the problem is not a scarcity of information, but rather an inability to effectively process and personalize vast amounts of data on job roles, required competencies, and market trends.

This information-to-action gap presents a clear opportunity for a technological solution. The PathFinder system is conceived to address this very problem by leveraging machine learning to provide personalized, intelligent career guidance.[1, 1] The system aims to function as a sophisticated matching engine, connecting the supply side of the labor market (individuals and their unique skill sets) with the demand side (employers and their specific needs). By analyzing a user's comprehensive profile, PathFinder can predict suitable career trajectories, identify critical skill gaps, and recommend relevant job opportunities, thereby transforming a process of anxious speculation into one of confident, strategic planning.1 The system's value proposition is rooted in its ability to create a clear, actionable signal from the noise of the modern job market.

### 1.2 Stakeholder Goals and Perspectives

The PathFinder ecosystem is designed to serve a diverse set of stakeholders, each with distinct yet interconnected objectives. The successful operation of the platform depends on its ability to deliver value to all participating groups.

* **Students and Job Seekers (Primary Customer):** This group represents the core user base. Their primary goal is to gain clarity and direction in their career planning. They need a tool that can translate their academic and personal profiles into tangible career options, highlight areas for professional development, and connect them with relevant employment opportunities. For them, PathFinder is a tool for empowerment, designed to reduce career-related anxiety and provide a competitive edge in the job market.1 For the initial iteration of this project, this group is specifically defined as students from the various departments within the College of Information at the University of North Texas, including Information Science, Data Science, Learning Technologies, and Linguistics.
* **Employers and Recruiters (Participating Actor):** For this stakeholder group, the primary goal is to streamline the recruitment process and improve the quality of candidate sourcing. Employers need an efficient mechanism to identify candidates whose skills and qualifications genuinely align with their job requirements. By providing a pre-vetted pool of candidates matched by a data-driven algorithm, PathFinder aims to reduce the time and resources spent on screening resumes and conducting initial interviews, ultimately leading to better hiring outcomes.
* **Career Advisors and Educational Institutions (Participating Actor):** This group seeks to enhance the quality and efficacy of their advisory services. Their goal is to leverage PathFinder as a supportive, data-rich tool to augment their professional judgment. The system can provide them with objective, data-backed recommendations and skill gap analyses for their students or clients, enabling more informed and impactful counseling sessions. They can use the platform's insights to guide curriculum development and advise on skill acquisition priorities.
* **System Administrators (Participating Actor):** This internal stakeholder group is responsible for the technical health and integrity of the platform. Their goals are centered on maintaining system performance, ensuring data security and privacy, managing user accounts, and periodically updating the underlying machine learning models to maintain their relevance and accuracy. Their work is critical to ensuring a reliable and trustworthy experience for all other users.

The platform's design facilitates a symbiotic relationship between these stakeholders. Students provide the rich profile data that powers the recommendation engine. Employers provide the real-world job data that grounds the system's recommendations in market reality. This continuous flow of information creates a dynamic data ecosystem where the system becomes progressively more intelligent and valuable to all participants over time.

### 1.3 System Vision and Scope (Iteration 1)

The overarching vision for PathFinder is to empower individuals to make informed and confident career decisions by providing a comprehensive, AI-driven guidance platform.1 The system seeks to democratize access to personalized career counseling, helping users align their professional aspirations with tangible opportunities in the global job market.

In adherence with sound software engineering methodology, the development of PathFinder will proceed in an iterative fashion. The first iteration is deliberately scoped to establish the core functionality and validate the primary value proposition. The focus will be on delivering a functional product that successfully implements the central features of career recommendation and job matching.1

The scope for Iteration 1 includes:

1. **User Registration and Profile Management:** Implementing secure account creation and profile management for both Students and Employers.1
2. **Core Recommendation Engine:** Developing and deploying a machine learning model (e.g., Decision Tree, Random Forest) capable of generating personalized career path recommendations based on student inputs.[1, 1]
3. **Skill Gap Analysis:** Implementing a feature to compare a student's existing skills against the requirements of a target career or job, highlighting any deficiencies.[1, 1]
4. **Job Matching and Application:** Creating a job board where employers can post openings and students can apply for positions that match their profiles.1
5. **Notification System:** Establishing an email-based notification system to inform students of the status of their job applications.1

Advanced features, such as the college recommendation service, in-depth market trend visualizations, and integration with third-party learning platforms, are explicitly deferred to subsequent iterations.1 This focused approach ensures that the foundational components of the system are built robustly before complexity is increased. The technology stack for this iteration will consist of a Python-based backend using the Flask web framework, a SQL database (such as SQLite or PostgreSQL) for data persistence, and various machine learning libraries for the analysis module.1

### 1.4 Illustrative Use Case Scenario: The Student's Journey

To fully illustrate the intended user experience and the practical application of the system's features, consider the scenario of Rahul Sharma, a hypothetical end-user.1

Rahul is a 22-year-old student in his final year of an Information Science program. He possesses a strong academic record and a range of technical skills but is uncertain about which specific career path to pursue. He registers on the PathFinder platform and begins by completing his profile, inputting details about his coursework, programming languages he knows, project experiences, and personal interests, such as problem-solving and data analysis.1

Upon submitting his profile, Rahul requests career recommendations. The system's machine learning module processes his information and presents him with a ranked list of potential careers. The top recommendations are "Data Scientist," "Cybersecurity Analyst," and "Software Development Engineer," each with a "match score" indicating the degree of alignment with his profile.1

Intrigued by the "Data Scientist" path, Rahul clicks for more details. The system displays a comprehensive overview of the role, including typical responsibilities, salary expectations, and future demand trends. Crucially, it also presents him with a skill gap analysis visualization. The chart clearly shows that while he has strong foundational skills in Python and databases, he is "lagging" in specific machine learning libraries and statistical modeling techniques required for senior data science roles.[1, 1]

This analysis is more than just data; it provides Rahul with a clear and actionable path forward. The system doesn't just identify a problem; it empowers him to solve it. Alongside the skill gap report, PathFinder recommends specific online courses and certifications that can help him acquire the missing competencies.1

Simultaneously, the platform's job matching feature presents him with a list of entry-level data analyst and junior data scientist positions that are a strong fit for his current skill set. He can review these postings, see how his skills match up, and apply directly through the platform. When he applies for a job, he receives an immediate confirmation email. A few days later, when an employer reviews his application and shortlists him for an interview, he receives another automated email notification from PathFinder.1

Through this journey, PathFinder has transformed Rahul's state of uncertainty into one of clarity and confidence. He now has a clear career goal, a concrete understanding of the skills he needs to develop, and a direct channel to relevant job opportunities. He is empowered to "strategize his future actions confidently," having moved from passive confusion to active career development.1

## 2. Functional and Non-Functional Requirements Specification

This section provides a formal specification of the system's requirements, divided into functional requirements, which define what the system must do, and non-functional requirements, which define the quality attributes and constraints under which the system must operate.

### 2.1 Functional Requirements

**2.1. Functional Requirements (What PathFinder Must Do)**

**2.1.1 User Management Subsystem (REQ-F-101 to REQ-F-104)**

This subsystem manages access and security, ensuring every interaction is authenticated and role-specific.

* **REQ-F-101:** The system shall allow new users, categorized as **Students, Employers, or Career Advisors**, to register for an account.
* **REQ-F-104 (RBAC):** The system shall implement **role-based access control** to ensure users can only access data appropriate to their role (e.g., preventing an Employer from viewing competitor postings).

**2.1.2 Student-Facing Subsystem (REQ-F-201 to REQ-F-206)**

This layer focuses on guidance and action—from profile setup to job application.

* **REQ-F-202 (Core AI):** Upon request, the system shall execute the **career recommendation algorithm** and display a ranked list of suitable career paths, with match percentages.
* **REQ-F-205 (Actionable Insight):** For any selected job or path, the system shall display a **skill gap analysis**, visually highlighting required but missing or "lagging" skills.
* **REQ-F-206:** The system shall provide a mechanism for students to **submit an application** for a specific job posting directly through the platform.

**2.1.3 Employer-Facing Subsystem (REQ-F-301 to REQ-F-304)**

This layer provides tools for efficient talent sourcing and application management.

* **REQ-F-301:** The system shall provide functionality for authenticated employers to **create, view, edit, and delete their own job postings**.
* **REQ-F-303:** The system shall allow employers to **view a list of student profiles that are algorithmically determined to be a strong match** for a specific job posting.

**2.1.4 Notification Service (REQ-F-401 to REQ-F-403)**

This service ensures timely communication regarding application status and recommendations.

* **REQ-F-401:** The system shall automatically send an email notification to a student whenever the **status of their job application is updated** by an employer.

**2.2. Use Case Diagram: The PathFinder Ecosystem Map 🗺️**

The Use Case Diagram visually defines the **system boundary** and confirms the **role-based separation of duties** for all four core actors (Student, Employer, Career Advisor, and System Administrator). This diagram serves as the high-level operational blueprint, showing the complete functional scope of PathFinder.

**2.2.1 Use Case Diagram of Student**

**A diagram of a pathfinder system

AI-generated content may be incorrect.**

## Use Case Scenario: Student Workflow (Complete System Interaction)

This scenario details the full journey of the **Student** actor, from initial profile creation and personalized AI guidance to job application submission and status tracking within the **PathFinder - AI-Powered Career Recommendation System**.

| **Component** | **Description** |
| --- | --- |
| **Use Case:** | **Student Workflow (Complete System Interaction)** |
| **Actor:** | **Student** |
| **Input:** | Registration/login credentials, comprehensive profile data (skills, education), request for career recommendation, job search filters, application submission confirmation. |
| **Output:** | Authenticated access, personalized career recommendations, visual skill gap analysis, dynamically matched job postings, application confirmation, and status notifications. |
| **Entry Stage:** | Student navigates to the career counseling system. System displays option to register or login. |
| **Exit Stage:** | All profile data saved; all recommendation requests processed; all applications submitted; application statuses reviewed; Student logs out and session ends. |

### Normal Operations

#### 1. Registration and Authentication Phase:

| **Step** | **Description** |
| --- | --- |
| **If New User (Registration):** |  |
| 1.1 | Student selects "Register as Student." |
| 1.2 | System displays registration form (Name, UNT Email, Password, Major). |
| 1.5 | **System creates student account** (status: unverified). |
| 1.7 | Student clicks verification link; System activates account and redirects to dashboard. |
| **If Existing User (Login):** |  |
| 1.9 | Student enters registered email and password. |
| 1.10 | System authenticates credentials, generates session token, and redirects to dashboard. |

#### 2. Profile Completion and Management Phase:

| **Step** | **Description** |
| --- | --- |
| 2.1 | Student clicks "Complete Profile" or "Edit Profile." |
| 2.2 | System displays profile sections: Education, Skills, Projects, Documents. |
| 2.3 | **Student inputs skills** (e.g., Python, SQL) from predefined lists, indicating proficiency (1-5). |
| 2.4 | Student uploads required documents (Resume, Cover Letter). |
| 2.6 | **System calculates Profile Completion Percentage** and displays it visually. |
| 2.7 | System updates database; displays "Profile saved successfully." |

#### 3. Core AI Guidance Phase (Recommendation & Skill Gap):

| **Step** | **Description** |
| --- | --- |
| 3.1 | Student clicks **"Get Career Recommendations"** (UC-001). |
| 3.2 | System validates Profile Completion Percentage (must be $\ge 80\%$). |
| 3.3 | **System sends complete Profile data to the ML Module.** |
| 3.5 | **System displays recommendations** with Match Score (0-100) and link to detail view. |
| 3.6 | Student selects a recommended career (e.g., "Data Scientist"). |
| 3.7 | System displays **Skill Gap Analysis** (UC-002) for the selected role. |
| 3.8 | System visually highlights **missing or lagging skills** required for the role. |
| 3.9 | System suggests **recommended online courses/certifications** to close the gap. |

#### 4. Job Search and Application Phase:

| **Step** | **Description** |
| --- | --- |
| 4.1 | Student clicks **"Job Matching/Board."** |
| 4.2 | **System displays dynamically matched job postings** based on the Student's profile. |
| 4.4 | Student selects a job posting to view full description. |
| 4.5 | System displays the Job Description, Employer details, and a **Student-Job Match Score.** |
| 4.6 | Student clicks **"Apply Now"** (UC-003). |
| 4.7 | System prompts Student to confirm application documents (Resume, Cover Letter). |
| 4.9 | **System creates an Application record** (status: Submitted). |
| 4.11 | **System triggers Notification Service** (REQ-F-402). |
| 4.12 | System sends automated **email acknowledgment** to the Student. |
| 4.13 | System displays confirmation: "Application submitted successfully." |

#### 5. Application Status Tracking Phase:

| **Step** | **Description** |
| --- | --- |
| 5.1 | Student clicks "My Applications" on the dashboard. |
| 5.2 | System retrieves and displays a list of all submitted applications with current status. |
| 5.3 | System receives an **automated notification** (triggered by Employer decision). |
| 5.4 | **System displays in-app notification** alert (e.g., "Update on Junior Analyst Application"). |
| 5.5 | Student reviews the status change (e.g., "Shortlisted") and employer feedback notes. |

#### 6. Continue or Exit Phase:

| **Step** | **Description** |
| --- | --- |
| 6.1 | Student can continue refining profile, requesting recommendations, or applying for jobs. |
| 6.2 | Student clicks "Logout." |
| 6.3 | System destroys session token and logs out the student. |

### Exceptions (E):

| **ID** | **Description** | **Resolution/System Action** |
| --- | --- | --- |
| **E1:** | Student profile completion is less than $80\%$ when requesting a recommendation. | **System blocks request** and displays a warning: "Profile must be complete to run the AI." |
| **E2:** | Student attempts to upload a document in an invalid format (e.g., .txt). | **System displays error:** "Please upload documents in PDF or DOCX format only." |
| **E3:** | No jobs match the student's current profile or selected filters. | System displays: **"No matching jobs found."** Suggestions to broaden filters or check skill gaps are provided. |
| **E4:** | Student attempts to apply for a job that has passed its application deadline. | **System disables the "Apply Now" button** and displays: "Application window closed on [Date]." |
| **E5:** | The ML module fails to return a result during recommendation generation. | System displays a **technical error** and advises the student to try again later. |
| **E6:** | Notification email delivery fails (e.g., mailbox full). | System logs failure and **prioritizes the in-app notification** to ensure status delivery. |
| **E7:** | Invalid login credentials. | System shows "Invalid email or password" and applies a retry limit. |
| **E8:** | Student tries to save a skill that is not on the predefined list without approval. | System saves the skill as **"Pending Verification"** and excludes it from the ML until validated. |
| **E9:** | Network interruption occurs during profile saving. | System displays a connectivity error and attempts to **preserve user input** for retry. |

**2.2.2 Use Case Diagram of Employer**

**A diagram of a career counseling system

AI-generated content may be incorrect.**

**Use Case Scenario:**  
  
**Use Case: Employer Workflow (Complete System Interaction)**

* **Actor:** Employer
* **Input:** Registration/login credentials, company profile details, job posting information, application status decisions
* **Output:** Authenticated access, company profile created/updated, job postings published, matched candidates viewed, application statuses updated with student notifications
* **Entry Stage:**
  + Employer navigates to career counseling system
  + System displays option to register or login as employer
* **Normal Operations:**
  + **Registration/Authentication Phase:**
    - **If New User (Registration):**
      * Employer selects "Register as Employer"
      * System displays registration form
      * Employer enters company name, industry, email, and password
      * Employer provides company description
      * Employer accepts terms and conditions
      * System validates email format and password strength
      * System creates employer account in database
      * System sends verification email
      * Employer clicks verification link
      * System activates account
    - **If Existing User (Login):**
      * Employer enters registered email and password
      * System authenticates credentials
      * System generates session token
    - System redirects to employer dashboard
  + **Manage Company Profile Phase:**
    - Employer clicks "Company Profile" or "Edit Profile"
    - System retrieves and displays current company information
    - Employer updates company name, industry, description
    - Employer uploads company logo (optional)
    - Employer updates contact details (phone, website, address)
    - Employer clicks "Save Changes"
    - System validates all fields
    - System updates database with new information
    - System displays "Profile updated successfully"
  + **Manage Job Postings Phase:**
    - **Create New Job:**
      * Employer clicks "Post New Job"
      * System displays job creation form
      * Employer enters job title
      * Employer writes detailed job description
      * Employer specifies required skills from predefined list or adds custom skills
      * Employer adds preferred qualifications, location, salary range
      * Employer sets application deadline
      * Employer clicks "Publish Job"
      * System validates required fields
      * System saves job posting to database
      * AI matching system begins analyzing and matching students based on skills
      * System displays "Job posted successfully" with job ID
    - **Edit Existing Job:**
      * Employer selects job from list
      * System displays job in edit mode
      * Employer modifies desired fields
      * System saves updates and re-runs matching if skills changed
    - **Remove Job:**
      * Employer selects job to delete
      * System requests confirmation
      * System archives job and notifies applicants
  + **View Matched Candidates Phase:**
    - Employer clicks "View Matched Candidates" for specific job
    - System retrieves job requirements and required skills
    - AI matching algorithm queries student database
    - System calculates match scores based on skill overlap
    - System ranks students by match percentage
    - System displays matched candidates with: name, match score, key skills, profile link
    - Employer can sort and filter results
    - Employer reviews list of AI-matched students
  + **Manage Applications Phase:**
    - Employer clicks "View Applications" for job posting
    - System displays list of students who applied
    - Employer selects candidate profile to review
    - System fetches complete student profile with resume, skills, assessments
    - System displays AI match analysis and skill-gap report
    - Employer reviews candidate qualifications against job requirements
    - Employer clicks "Update Status" button
    - System presents status options: "Shortlisted" or "Rejected"
    - Employer selects status and optionally adds feedback notes
    - Employer clicks "Confirm"
    - System validates status selection
    - System updates application status in database with timestamp
    - System records employer ID and decision notes
    - System triggers notification service
    - Notification service retrieves student email from database
    - System sends automated email to student (job title, company, status, notes)
    - System creates in-app notification for student
    - System displays confirmation: "Application status updated, student notified"
    - Employer continues reviewing other candidates or returns to dashboard
  + **Continue or Exit Phase:**
    - Employer can manage more jobs, review more applications, or logout
    - System maintains session until logout or timeout
* **Exit Stage:**
  + All company profile updates saved
  + All job postings published/updated/removed
  + All application statuses updated
  + All students notified of status changes
  + Employer logs out and session ends
  + All changes persisted in database
* **Exceptions:**
  + **E1:** Email already registered → System displays "Account exists, please login"
  + **E2:** Weak password during registration → System shows password requirements
  + **E3:** Email verification timeout → System offers to resend verification email
  + **E4:** Invalid login credentials → System shows "Invalid credentials" with retry limit
  + **E5:** Invalid logo file format → System prompts "Please upload JPG, PNG, or GIF"
  + **E6:** Profile image too large → System displays size limit
  + **E7:** Missing required job fields → System highlights fields and prevents submission
  + **E8:** No skills specified for job → System warns "At least one skill required for matching"
  + **E9:** Invalid application deadline date → System shows date picker
  + **E10:** Delete job with pending applications → System requires confirmation
  + **E11:** No matching candidates found → System displays "No candidates match required skills yet"
  + **E12:** Matching algorithm failure → System shows all relevant students as fallback
  + **E13:** No application status selected → System prompts "Please select a status"
  + **E14:** Database update fails → System shows error with retry option and data preservation
  + **E15:** Notification service down → Status saved but notification queued; system warns employer
  + **E16:** Student account deactivated → Status updated but notification cannot be sent
  + **E17:** Concurrent status update → System detects conflict and shows current status
  + **E18:** Email delivery failure → System logs failure and creates in-app notification backup
  + **E19:** Session expired → System prompts re-login while preserving work
  + **E20:** Network interruption → System displays connectivity error with retry option

**2.2.3 Use Case Diagram of Advisor**

A screenshot of a computer

AI-generated content may be incorrect.

Use Case Scenario:

**CAREER ADVISOR - COMPREHENSIVE USE CASE**

**Use Case: Career Advisor Workflow (Complete System Interaction)**

* **Actor:** Career Advisor
* **Input:** Login credentials, student selection, personalized guidance text, course recommendations, professional advice
* **Output:** Authenticated access, student reports displayed, guidance saved and student notified
* **Entry Stage:**
  + Advisor navigates to the career counseling system login page
  + System displays login form for advisors
* **Normal Operations:**
  + **Authentication Phase:**
    - Advisor enters username/email and password
    - System validates credentials against database
    - Authentication service generates session token
    - System grants access to advisor-specific dashboard
  + **View Assigned Students Phase:**
    - System automatically displays dashboard with list of assigned students
    - System shows student names, IDs, last activity date, and report status
    - Each student entry has clickable link to detailed profile
    - Advisor reviews the list and selects a student to mentor
  + **Access Student Reports Phase:**
    - Advisor clicks on specific student name/profile
    - System captures selected student ID
    - System queries database for complete student profile information
    - System retrieves AI-generated career recommendations
    - System fetches skill-gap analysis reports
    - System retrieves any previous advisor feedback/comments
    - System compiles all data into comprehensive report view
    - Report displays: demographics, assessment results, recommended careers, skill gaps, previous guidance
  + **Provide Guidance Phase:**
    - Advisor reviews all AI-generated recommendations and analyses
    - Advisor clicks "Add Guidance" or "Provide Feedback" button
    - System displays guidance input form with multiple text areas
    - Advisor types personalized comments based on AI recommendations
    - Advisor adds specific course suggestions to address skill gaps
    - Advisor provides professional advice and career development next steps
    - Advisor clicks "Submit Guidance" button
    - System validates input (non-empty, appropriate length)
    - System saves guidance to database with student ID, advisor ID, and timestamp
    - System triggers notification service
    - System sends email and in-app notification to student
    - System displays success message: "Guidance saved and student notified"
  + **Continue or Exit Phase:**
    - Advisor returned to assigned students list
    - Advisor can select another student to review or logout
    - System maintains session until advisor logs out or session expires
* **Exit Stage:**
  + All guidance successfully saved across all reviewed students
  + Students notified of new advisor feedback
  + Advisor logs out and session ends
  + All changes persisted in database
* **Exceptions:**
  + **E1:** Invalid login credentials → System displays "Invalid username or password" with retry option
  + **E2:** Account locked/suspended → System shows "Contact administrator to unlock account"
  + **E3:** No students assigned → System displays "No students currently assigned to you"
  + **E4:** Student data incomplete → System shows available information with "Incomplete profile" warning
  + **E5:** AI recommendations not generated → System displays "Analysis in progress, check back later"
  + **E6:** Empty guidance submission → System prompts "Please enter guidance before submitting"
  + **E7:** Text exceeds character limit → System shows character count warning and prevents submission
  + **E8:** Database save failure → System displays error and offers to save draft locally
  + **E9:** Notification service unavailable → Guidance saved but student not notified; system logs for retry
  + **E10:** Student account deactivated → System warns advisor before saving guidance
  + **E11:** Session timeout during work → System prompts re-login while preserving unsaved guidance
  + **E12:** Network interruption → System displays connectivity error with data preservation and retry option
  + **E13:** Database connection lost → System uses cached data with "Offline mode" indicator
  + **E14:** Concurrent advisor access to same student → System handles with last-write-wins, shows warning

**2.2.4 Use Case Diagram of Administrator**

**A diagram of a person with text

AI-generated content may be incorrect.**

**Use Case Scenario: System Administrator Workflow (Maintenance & Governance)**

This scenario details the full lifecycle of system governance and critical maintenance tasks performed by the **System Administrator** actor, focusing on user management, reporting, and the essential process of retraining the Machine Learning (ML) Recommendation Model.

| **Component** | **Description** |
| --- | --- |
| **Use Case:** | **System Administrator Workflow (Maintenance & Governance)** |
| **Actor:** | **System Administrator** (Highest authority internal user) |
| **Goal:** | To maintain system integrity, manage all user accounts securely, and ensure the relevance and accuracy of the core AI recommendation model. |
| **Input:** | Administrator credentials, user ID/role for management, access to the ML Model Management Interface, new training dataset/version number. |
| **Output:** | Successful authentication, updated user accounts (status/role), generated system reports, updated and redeployed ML model version, and audit logs of all changes. |
| **Entry Stage:** | Administrator navigates to the secure System Admin console URL. System displays restricted login portal. |
| **Exit Stage:** | All system changes are audited and persisted; the ML model is stable or successfully updated; Administrator logs out and session ends. |

**Normal Operations**

**1. Authentication and Dashboard Access Phase:**

| **Step** | **Description** |
| --- | --- |
| 1.1 | Administrator enters unique credentials and Two-Factor Authentication (2FA) code. |
| 1.2 | System authenticates credentials and checks for Administrator role privileges. |
| 1.3 | System generates a privileged session token. |
| 1.4 | **System redirects to the secure Admin Dashboard**, displaying system health metrics (uptime, user load) and critical alerts (model stability, database warnings). |

**2. User Account Management Phase:**

| **Step** | **Description** |
| --- | --- |
| 2.1 | Administrator clicks **"User Management"** (UC-006). |
| 2.2 | System displays a searchable list of all users (Students, Employers, Advisors). |
| 2.3 | Administrator searches for a specific user ID or role (e.g., a problematic Employer). |
| 2.4 | Administrator selects the target user account. |
| 2.5 | System displays user details and management options: Reset Password, Deactivate Account, Change Role. |
| 2.6 | Administrator clicks **"Deactivate Account"** and confirms the action with a brief, required note for the audit log. |
| 2.7 | System immediately updates the user status to *Deactivated* and terminates all active sessions for that user. |
| 2.8 | System records the action, timestamp, and Admin ID in the security audit log. |

**3. System Reporting and Audit Phase:**

| **Step** | **Description** |
| --- | --- |
| 3.1 | Administrator clicks **"System Reports"** $\rightarrow$ **"Generate Placement Metrics."** |
| 3.2 | System prompts Admin for report parameters (e.g., Date Range, College/Major Filter). |
| 3.3 | System executes a complex query against the student and application database. |
| 3.4 | **System generates a CSV or graphical report** detailing student placement rates, most active employers, and successful applications. |
| 3.5 | Administrator downloads or views the generated report. |

**4. Core ML Model Governance Phase:**

| **Step** | **Description** |
| --- | --- |
| 4.1 | Administrator clicks **"ML Model Management"** (UC-007). |
| 4.2 | System displays current model version, last training date, and performance metrics (accuracy, confidence levels). |
| 4.3 | Administrator selects **"Initiate Model Retraining"** and provides the path/version number for the new dataset. |
| 4.4 | System verifies the new dataset and confirms maintenance window is open (low traffic). |
| 4.5 | **System executes the ML Module Retraining Pipeline** on the dedicated processing server. |
| 4.6 | Retraining Pipeline completes and **System runs Validation Tests** (e.g., against a holdout set). |
| 4.7 | Validation Passes: System prompts Administrator to **"Confirm Deployment of New Model."** |
| 4.8 | Administrator confirms; System redirects recommendation requests to the new model version (Blue/Green Deployment). |
| 4.9 | System logs the new model version and deployment timestamp as the official current version. |
| 4.10 | Validation Fails: System aborts deployment, logs the failure, and **reverts to serving the previous stable model.** |

**5. Exit Phase:**

| **Step** | **Description** |
| --- | --- |
| 5.1 | Administrator clicks **"Logout."** |
| 5.2 | System destroys the privileged session token and records the logout timestamp in the audit log. |
| 5.3 | System redirects to the main public login page. |

**Exceptions (E):**

| **ID** | **Description** | **Resolution/System Action** |
| --- | --- | --- |
| **E1:** | Administrator login attempt fails due to invalid credentials. | System shows "Access Denied" and locks the account after 5 failed attempts, requiring manual reset (E4 fallback). |
| **E2:** | Administrator attempts to change a user's role without proper secondary authorization (if required). | System rejects the change and prompts for the secondary authentication code. |
| **E3:** | New training dataset provided for ML model is corrupted or incomplete. | System halts the retraining pipeline immediately and logs an **"ML Data Integrity Error."** |
| **E4:** | Validation Tests on the new ML model fail to meet the minimum required accuracy threshold ($>85\%$). | System aborts the deployment process and continues serving the *previous, stable* model (E12 fallback). |
| **E5:** | Administrator tries to generate a report on an unknown date range. | System displays date input validation error. |
| **E6:** | Concurrent user update conflict (two Admins edit the same account). | System detects conflict and prompts the Administrator to reload the user's data to see the other Admin's latest changes (E17 fallback). |

### 2.2 Non-Functional Requirements (NFRs)

Non-functional requirements define the essential quality attributes of the system. While not explicitly enumerated in the source documents, they are strongly implied by statements regarding security, performance, and usability.1 The following table formalizes these critical constraints.

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **ID** | **Requirement** | **Justification / Source** |
| **Security** | REQ-NF-SEC-01 | All user passwords must be stored in the database only after being hashed using a modern, salted, one-way hashing algorithm (e.g., bcrypt, Argon2). | Derived from the requirement for "password hashing" to ensure secure data storage.1 |
|  | REQ-NF-SEC-02 | All data transmitted between the client web browser and the server must be encrypted using Transport Layer Security (TLS) 1.2 or a more recent version. | A standard best practice for web applications handling personally identifiable information (PII). |
|  | REQ-NF-SEC-03 | The system must enforce authorization checks on all server-side operations to prevent unauthorized data access or modification. | Derived from the need for "secure access control" for different user roles.1 |
| **Performance** | REQ-NF-PERF-01 | The Career Recommendation Service API endpoint shall return results to the user within 3 seconds under a simulated load of 100 concurrent users. | Quantifies the requirement for a "quick and dependable response to the recommendations".1 |
|  | REQ-NF-PERF-02 | The employer's "View Matching Candidates" page shall load and display the list of students within 5 seconds for a job posting with up to 500 applicants. | Ensures an efficient workflow for recruiters, a key stakeholder group. |
| **Usability** | REQ-NF-USA-01 | The initial student profile creation and data input process shall be completable by a first-time, non-technical user in under 10 minutes. | Based on the stated focus on "user-friendly forms" to encourage user adoption and data completeness.1 |
|  | REQ-NF-USA-02 | The web application interface must be responsive and render correctly on the latest versions of major web browsers, including Google Chrome, Mozilla Firefox, and Apple Safari. | A standard requirement to ensure accessibility for the majority of the target audience. |
| **Reliability** | REQ-NF-REL-01 | The system shall maintain an availability (uptime) of 99.5% during standard business hours (9:00 AM to 5:00 PM in the primary target time zone). | A standard requirement for a production-level web service to ensure it is available when users need it. |
| **Data** | REQ-NF-DATA-01 | The initial dataset for training and system validation shall be exclusively composed of profiles representing students from the majors within the College of Information at the University of North Texas. | To narrow the project scope for Iteration 1 and focus on a specific, well-defined user group as per project guidelines. |

## 3. System Sequence Diagrams (SSDs)

System Sequence Diagrams are used to model the dynamic behavior of the system by illustrating the sequence of interactions between external actors and the system for a specific use case. The system is treated as a "black box," with the diagrams focusing solely on the input and output events that cross its boundary. This section provides textual descriptions of SSDs for two primary use cases.

### 3.1 Overview of System Interactions

The interactions with the PathFinder system are initiated by its external actors: the Student, the Employer, and the Career Advisor. These actors send messages (requests) to the system, and the system responds with information or a confirmation of an action. The following diagrams model the logical flow of these interactions, clarifying the system's responsibilities and defining its public interface from a user's perspective.

### 

### 3.2 Sequence and Activity Diagram of Student

**3.2.1 Sequence Diagram: Student Applies for a Job**

A diagram of a workflow

AI-generated content may be incorrect.

## Sequence Diagram: Student Applies for a Job (UC-003)

The following diagram details the sequence of interactions when a **Student** initiates a job application, modeling the communication flow between the actor and the necessary internal components: the UI, the Application Controller, the Database, and the Notification Service.

| **Participant** | **Description** |
| --- | --- |
| **Student** | The external actor initiating the action. |
| **UI: JobBoardPage** | The user interface displaying job details and the submission form. |
| **Ctrl: ApplicationController** | The system component managing application logic and transaction coordination. |
| **DB: JobPosting, Application** | The database layer storing job and application records. |
| **Service: Notification** | The component responsible for sending automated application confirmation emails. |

**3.2.2 Activity Diagram of Student**  
A diagram of a job application

AI-generated content may be incorrect.

### 3.3 Sequence and Activity Diagram of Employer

**3.3.1 Sequence Diagram of Employer**

A screenshot of a computer screen

AI-generated content may be incorrect.

**3.3.2 Activity Diagram of Employer**

**A diagram of activity diagram

AI-generated content may be incorrect.**

### 3.4 Sequence and Activity Diagram of Advisor

**3.4.1 Sequence Diagram of Advisor**

**A screenshot of a diagram

AI-generated content may be incorrect.**

**3.4.2 Activity Diagram of Advisor**

A diagram of a company

AI-generated content may be incorrect.

### 3.5 Sequence and Activity Diagram of Administrator

**3.5.1 Sequence Diagram of Administrator**

**A diagram of a software system

AI-generated content may be incorrect.**

**3.5.2 Activity Diagram of Administrator**

**A diagram of a system

AI-generated content may be incorrect.**

**3.6 Sequence and Class Diagram for system:**

**3.6.1 Class Diagram for System:**

**A screenshot of a chat

AI-generated content may be incorrect.**

**3.6.2 Sequence Diagram for System:**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

## 4. Domain Analysis and Domain Model

This section provides a conceptual model of the problem domain. It identifies the key entities, their attributes, and the relationships between them. This domain model serves as a static blueprint of the system's universe, providing a common vocabulary and understanding for both stakeholders and developers, independent of any specific implementation technology.

### 4.1 Identification of Key Domain Concepts

Through an analysis of the project's goals and requirements, several core concepts (or entities) have been identified. These concepts form the building blocks of the PathFinder system.

* **User:** A generalized concept representing any person who interacts with the system. It has specialized types (subclasses).
  + **Student:** A type of User who seeks career guidance. For the scope of this project, a Student is defined as an individual enrolled in a major within the College of Information at the University of North Texas. A student possesses a Profile.
  + **Employer:** A type of User representing a company or recruiter who posts job openings.
  + **CareerAdvisor:** A type of User who mentors students and reviews their progress.
* **Profile:** A comprehensive collection of a Student's professional and personal attributes. This is the primary input for the recommendation engine. It contains information about Education, Skills, and Interests.
* **JobPosting:** An advertisement for a job vacancy created by an Employer. Each posting has a specific set of required skills.
* **Skill:** A discrete competency, such as "Python programming," "Project Management," or "Data Visualization." The Skill entity is the fundamental unit of currency within the system. It is the critical link that allows the system to match a Student's Profile with an Employer's JobPosting. The consistency and quality of the skill vocabulary are paramount for the system's effectiveness, suggesting the need for a standardized taxonomy, potentially derived from a source like O\*NET.1
* **Application:** An associative entity that represents the act of a Student applying for a JobPosting. It links one Student to one JobPosting and has attributes of its own, such as applicationDate and status.
* **CareerRecommendation:** A data object representing a career path suggested to a student by the system. It includes the career title and a match score.

### 4.2 Conceptual Domain Model Diagram

The relationships between these key concepts can be described textually, outlining their connections and multiplicities (the number of instances of one class that can be related to one instance of another class).

* A User can be one of Student, Employer, or CareerAdvisor (an inheritance or specialization relationship).
* A Student *has exactly one* Profile. A Profile *belongs to exactly one* Student. (A one-to-one relationship).
* An Employer *posts zero or more* JobPostings. A JobPosting *is posted by exactly one* Employer. (A one-to-many relationship).
* A Student *submits zero or more* Applications. An Application *is submitted by exactly one* Student. (A one-to-many relationship).
* A JobPosting *receives zero or more* Applications. An Application *is for exactly one* JobPosting. (A one-to-many relationship).
* A Profile *possesses many* Skills. A Skill *can be possessed by many* Profiles. (A many-to-many relationship).
* A JobPosting *requires many* Skills. A Skill *can be required by many* JobPostings. (A many-to-many relationship).

### 4.3 Data Dictionary

To further elaborate on the domain model, the following data dictionary defines the essential attributes for each key concept. This provides a more detailed specification that will directly inform the design of the database schema.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Concept (Class)** | **Attribute** | **Data Type** | **Description** | **Source** |
| **Student** | studentID | UUID | A system-generated unique identifier for the student. | Inferred |
|  | name | String | The full name of the student. | 1 |
|  | email | String | The student's primary email address, used for login and notifications. | 1 |
|  | contactNumber | String | The student's phone number. | 1 |
|  | hashedPassword | String | The salted hash of the user's password. | 1 |
| **Profile** | profileID | UUID | A system-generated unique identifier for the profile. | Inferred |
|  | educationHistory | Text | A description of the student's academic background. | 1 |
|  | interests | Text | A description of the student's career interests. | 1 |
| **JobPosting** | jobID | UUID | A system-generated unique identifier for the job posting. | Inferred |
|  | title | String | The official title or designation of the job. | 1 |
|  | description | Text | A detailed description of the role, responsibilities, and requirements. | 1 |
|  | location | String | The physical or remote location of the job. | 1 |
|  | salary | String | The optional salary or compensation range for the position. | 1 |
|  | datePosted | DateTime | The timestamp when the job was posted. | Inferred |
| **Skill** | skillID | UUID | A system-generated unique identifier for the skill. | Inferred |
|  | name | String | The standardized name of the skill (e.g., "Python," "Data Analysis"). | 1 |
|  | category | String | A category for the skill (e.g., "Programming Language," "Soft Skill"). | Inferred |
| **Application** | applicationID | UUID | A system-generated unique identifier for the application. | Inferred |
|  | applicationDate | DateTime | The timestamp when the application was submitted. | Inferred |
|  | status | Enum | The current status of the application (e.g., "Submitted," "Viewed," "Accepted," "Rejected"). | 1 |

5. **Nonfunctional Requirements — FURPS + Model**

The PathFinder system’s quality attributes are categorized using the **FURPS + framework**, which stands for *Functionality, Usability, Reliability, Performance, Supportability,* and the “ + ” extension that includes *design, implementation, interface, physical, and legal constraints.*  
This section describes the FURPS + requirements and identifies which will be implemented in **Iteration 1**.

**5.1 Functionality**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **Iteration 1** | **Verification Method** |
| F-SEC-01 | Passwords must be salted and hashed using Argon2 or bcrypt. | ✅ | Code review and database inspection. |
| F-SEC-02 | All data transmission must use TLS 1.2 or higher. | ✅ | Environment check and SSL security test. |
| F-AUTH-01 | Role-based access control (RBAC) separating Student, Employer, and Advisor roles. | ✅ | API role tests and integration tests. |
| F-AUD-01 | Authorization checks enforced on all server-side operations. | ✅ | Penetration test and negative test cases. |

**5.2 Usability**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **Iteration 1** | **Verification Method** |
| U-ONB-01 | Onboarding and profile completion must take ≤ 10 minutes for non-technical users. | ✅ | User testing and timed task analysis. |
| U-RESP-01 | Interface must render responsively on Chrome, Firefox, and Safari. | ✅ | Cross-browser testing. |
| U-ACC-01 | Forms must be keyboard navigable and WCAG 2.1 compliant. | ✅ | Accessibility scan and manual review. |
| U-CONT-01 | System must display clear match explanations and tooltips. | ✅ | UX review and user survey (CSAT ≥ 4 / 5). |

**5.3 Reliability**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **Iteration 1** | **Verification Method** |
| R-AVAIL-01 | System uptime of ≥ 99.5 % during business hours. | ⏳ | Uptime monitoring report. |
| R-RECOV-01 | Graceful error handling with user-friendly messages. | ✅ | Fault-injection tests and UI review. |
| R-RETRY-01 | Idempotent background email jobs with retry logic. | ✅ | Queue tests under failure simulation. |

**5.4 Performance**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **Iteration 1** | **Verification Method** |
| P-API-01 | Career recommendation endpoint ≤ 3 s (p95) under 100 concurrent users. | ✅ | Load test (JMeter or Locust). |
| P-EMP-01 | Employer “View Candidates” loads ≤ 5 s for ≤ 500 applications. | ✅ | Dataset simulation and timing test. |
| P-SRCH-01 | Job search results first paint ≤ 2 s (p95). | ⏳ | Lighthouse and RUM metrics. |
| P-RES-01 | API error rate < 1 % under expected load. | ✅ | Load test summary and logs. |

**5.5 Supportability**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **Iteration 1** | **Verification Method** |
| S-TEST-01 | Automated unit tests ≥ 80 % coverage; CI runs on every commit. | ✅ | CI coverage report. |
| S-LOG-01 | Structured JSON logs with correlation IDs enabled. | ✅ | Log review and trace validation. |
| S-DOC-01 | REST API documented using OpenAPI 3 / Swagger UI. | ✅ | Documentation review. |
| S-I18N-01 | UI text externalized for future localization. | ⏳ | Code inspection of string files. |

**5.6 “+” Constraints and External Requirements**

**5.6.1 Design Constraints**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **ID** | **Requirement** | **Iteration 1** | **Verification Method** | | --- | --- | --- | --- | | D-ARCH-01 | Three-tier architecture (Web UI / Flask API / SQL DB) with RecommendationService façade. | ✅ | Architecture review and code inspection. | | D-SCHEMA-01 | Relational schema normalized with join tables (Profile–Skill, Job–Skill). | ✅ | Database migration review. |   **5.6.2 Implementation Constraints** |
| | **ID** | **Requirement** | **Iteration 1** | **Verification Method** | | --- | --- | --- | --- | | I-LANG-01 | Python 3.11 + Flask + SQLAlchemy + scikit-learn. | ✅ | Environment file and pip freeze. | | I-CI-01 | CI pipeline includes linting, tests, and security scan. | ✅ | CI config and run logs. |   **5.6.3 Interface Requirements** |
| | **ID** | **Requirement** | **Iteration 1** | **Verification Method** | | --- | --- | --- | --- | | X-API-01 | RESTful endpoints with JWT authentication (/api/v1/...). | ✅ | API spec and token validation test. | | X-MAIL-01 | Email notifications via queue-backed SMTP service. | ✅ | Email integration test. |   **5.6.4 Physical / Operational / Data Requirements** |
| | **ID** | **Requirement** | **Iteration 1** | **Verification Method** | | --- | --- | --- | --- | | PHY-ENV-01 | Single-node containerized deployment (MVP). | ✅ | Dockerfile and compose run. | | DATA-SCOPE-01 | Training dataset restricted to UNT College of Information students. | ✅ | Dataset audit and script review. | | BACKUP-01 | Nightly DB backups retained 7 days (non-prod). | ⏳ | Backup job logs. |   **5.6.5 Legal and Privacy Requirements** |
| | **ID** | **Requirement** | **Iteration 1** | **Verification Method** | | --- | --- | --- | --- | | L-PII-01 | PII minimization (collect only necessary fields). | ✅ | Data inventory review. | | L-CONSENT-01 | User consent and privacy notice on registration form. | ✅ | UI inspection and stored flag. | | L-GDPR-01 | Data export/delete endpoints planned for Iteration 2. | ⏳ | API design doc and backlog ticket. | |

**5.7 Summary**

**Implemented in Iteration 1 (✅):**  
Password hashing and TLS; RBAC and authorization; responsive UI and 10-minute onboarding; error handling; email queue; core performance targets; automated tests; logging; OpenAPI docs; three-tier architecture; REST + JWT; containerized deployment; UNT-scoped dataset; PII minimization; privacy consent.

**Deferred (⏳):** Uptime monitoring, 2 s job search target, localization, nightly backups, data export/delete APIs.

**5.8 Measurement and Acceptance**

* **All performance SLOs at p95** for 100 concurrent users.
* **Security and privacy tests** validated via automated CI checks and manual audits.
* **Deferred features** are tracked in the Iteration 2 backlog with assigned owners.

## 6. User Interface Design

This section outlines the high-level design concepts for the system's user interfaces (UIs). It describes the layout and key components of the dashboards for each primary actor, translating the functional requirements into a conceptual user experience (UX).

### 5.1 UI/UX Philosophy

The design philosophy for PathFinder is centered on clarity, empowerment, and efficiency. The UI must be intuitive and accessible, catering to the distinct needs and goals of each user persona. The design will be guided by the specific responsibilities of the development team members: Michelle Dogo's focus on "user-friendly forms" and a polished student interface, and Spoorthi Hassan Sathyanarayana's focus on functional "dashboard design" and role-based accessibility for employers and advisors.1

The system's UI design is predicated on the understanding that it must provide three different "lenses" through which to view the same core data.

* For the **Student**, the UI is a tool for *exploration and personal development*. It must be visually engaging and encouraging, designed to answer the question, "What can I become, and how do I get there?"
* For the **Employer**, the UI is a tool for *evaluation and filtering*. It must be data-dense and efficient, designed to answer the question, "Who is the best candidate for this role?"
* For the **Career Advisor**, the UI is a tool for *oversight and mentorship*. It must provide a consolidated view of student progress, designed to answer the question, "How are my students progressing, and where do they need guidance?"

### 5.2 Student Dashboard Wireframe Concept

The Student Dashboard is the central hub for the primary user. It is designed to be a personalized and dynamic space that guides the student through their career discovery journey.

* **Layout:** A modern, two-column layout. A fixed navigation sidebar on the left provides access to major sections, while the main content area on the right displays the relevant information.
* **Left Navigation Pane:** Contains links to "Dashboard Home," "My Profile," "Career Recommendations," "Job Search," and "My Applications."
* **Main Content Area (Dashboard View):**
  + **Welcome Banner:** A personalized greeting and a "Profile Completeness" progress bar to encourage users to provide comprehensive information.
  + **Top Career Recommendations:** A horizontally scrolling section with visually appealing "cards." Each card represents a recommended career, displaying the job title, a match score, and a brief description. Clicking a card navigates to a detailed view with the skill gap analysis.
  + **Skill Gap Visualization:** On the detailed career view page, a dynamic chart (e.g., a radar chart or a side-by-side bar chart) will visually compare the student's current skills against the skills required for the selected career. Missing skills will be clearly highlighted.
  + **Recommended Jobs Feed:** A scrollable list of job postings that are a strong match for the student's current profile, with options to save or apply directly from the list.

### 5.3 Employer Dashboard Wireframe Concept

The Employer Dashboard is designed for efficiency and task-completion. The interface prioritizes clear information hierarchy and quick access to applicant management tools.

* **Layout:** A professional, data-centric dashboard.
* **Main View:** A summary view showing key metrics (e.g., total active postings, new applicants today). The primary feature is a table of all active job postings.
* **Job Postings Table:** Columns for "Job Title," "Date Posted," "Status" (e.g., Open, Closed), and "Number of Applicants." Each row is clickable.
* **Job Posting Creation/Edit View:** A clean, multi-step form for creating or editing a job posting. This includes fields for title, description, and a tag-based input field with auto-completion for adding required skills from the system's standardized list.
* **Applicant Management View:** Accessed by clicking a job posting. This view displays a table of all applicants for that job, pre-sorted by their algorithm-generated match score. Each applicant row displays their name, match score, and current application status. An employer can click on a candidate to view their full profile and use simple buttons ("Shortlist," "Reject") to update the application status.

### 5.4 Career Advisor Dashboard Wireframe Concept

The Career Advisor Dashboard is a portfolio management tool designed for oversight and guidance. It provides a read-only, consolidated view of the students under the advisor's purview.

* **Layout:** A clean, professional interface focused on student tracking.
* **Main View:** A searchable and sortable list of all students assigned to the advisor. Each row displays the student's name, their primary career interest (if saved), and a timestamp of their last activity.
* **Student Detail View:** Clicking on a student's name opens a detailed view. This view is essentially a read-only replica of that student's dashboard. The advisor can see the exact same career recommendations, skill gap analyses, and job matches that the student sees. This shared context is critical for enabling the advisor to provide relevant, specific, and actionable guidance during counseling sessions. The interface would also include a section for the advisor to log private notes on the student's progress.

## 6. Class Diagram and Interface Specification

This final section transitions from conceptual analysis to a concrete software design blueprint. It outlines a high-level software architecture, proposes a design-level class diagram, and specifies the key software interfaces, providing a clear plan for the development team.

A screenshot of a computer program

AI-generated content may be incorrect.

### 6.1 High-Level Software Architecture

The PathFinder system will be implemented using a classic three-tier architecture, which separates concerns and promotes modularity. This architecture aligns with the proposed technology stack of a Python backend (Flask/Django), a SQL database, and a modern web frontend.1

1. **Presentation Tier (Frontend):** This tier is responsible for rendering the user interface and capturing user input. It will be built using HTML, CSS, and JavaScript. To create a dynamic and responsive user experience, it will use AJAX (Asynchronous JavaScript and XML) to communicate with the backend without requiring full page reloads. This tier is the primary responsibility of the frontend development specialists.1
2. **Application Tier (Backend):** This is the core of the system, containing the business logic. It will be developed using the Flask web framework in Python. This tier is responsible for handling incoming HTTP requests from the frontend, processing user data, enforcing business rules, interacting with the database, and orchestrating calls to the machine learning module. It exposes a set of RESTful API endpoints for the frontend to consume. This tier is the responsibility of the backend and database specialists.1
3. **Data Tier:** This tier consists of two main components:
   * **Persistent Storage:** A relational database (e.g., PostgreSQL or MySQL) will be used to store all user data, job postings, applications, and other persistent information. An Object-Relational Mapper (ORM) like SQLAlchemy will be used to abstract database interactions.
   * **Analysis Module (Machine Learning):** This component contains the logic for career prediction and job matching. It will be developed using Python libraries such as scikit-learn. While it runs on the same backend server in Iteration 1, it is architecturally distinct. The dataset used for training and validating these models will be specifically curated to represent students from the College of Information at the University of North Texas, encompassing majors such as Information Science, Data Science, Learning Technologies, and Linguistics.2

A diagram of a computer

AI-generated content may be incorrect.

This architecture deliberately isolates the complex and potentially volatile machine learning component from the more stable application and web logic. The application tier communicates with the analysis module through a well-defined internal interface.

### 6.2 System Design Class Diagram

The following is a textual description of a design-level class diagram, which models the key software classes that will be implemented in the application tier.

* **Controller Classes:**
  + UserController: Handles HTTP requests related to user management, such as register(), login(), and get\_profile().
  + JobController: Manages requests related to job postings and applications, with methods like create\_job(), get\_job\_by\_id(), apply\_to\_job(), and get\_applicants\_for\_job().
  + RecommendationController: Exposes the functionality of the analysis module via API endpoints, with methods like get\_career\_recommendations().
* **Service Classes:**
  + RecommendationService: This is a crucial service class that encapsulates all machine learning logic. It acts as a facade, providing a simple interface to the complex underlying models. It will have methods like predict\_careers(profile) and match\_jobs(profile). This class is the core responsibility of the machine learning specialist.1
  + NotificationService: A service responsible for sending emails. It will have a method like send\_email(to\_address, subject, body).
* **Repository/Data Access Classes:**
  + UserRepository, JobRepository, ApplicationRepository: These classes implement the repository pattern, abstracting all database operations for their respective entities. For example, UserRepository would have methods like find\_by\_email(email) and save(user).
* **Model Classes (Data Structures):**
  + User, Profile, JobPosting, Skill, Application: These are plain data classes (often mapped to database tables by an ORM) that represent the core entities of the domain.

### 6.3 Interface Specifications

To ensure clear contracts between system components and facilitate parallel development, the following software interfaces are specified.

#### 6.3.1 RecommendationService Interface

This interface defines the contract between the main application logic and the machine learning module.

Python

class RecommendationService:  
 """  
 Defines the contract for the career and job recommendation engine.  
 """  
  
 def get\_career\_recommendations(self, student\_profile: Profile, top\_n: int = 5) -> list:  
 """  
 Analyzes a student's profile and returns the top\_n most suitable career paths.  
 Each CareerRecommendation object contains the career title and a match score.  
 """  
 pass  
  
 def get\_job\_matches(self, student\_profile: Profile, filters: dict = None) -> list[JobMatch]:  
 """  
 Finds and ranks job postings from the database that match a student's profile.  
 Each JobMatch object contains the JobPosting and a match score.  
 """  
 pass  
  
 def analyze\_skill\_gap(self, student\_profile: Profile, target\_career\_id: str) -> SkillGapReport:  
 """  
 Compares a student's skills to those required by a target career path.  
 Returns a SkillGapReport object detailing possessed skills and missing skills.  
 """  
 pass

#### 6.3.2 Backend API Endpoint Specification (Conceptual)

This section outlines the key RESTful API endpoints that the presentation tier will use to interact with the application tier.

* **User Management:**
  + POST /api/v1/users/register: Creates a new user account. Request body contains user details.
  + POST /api/v1/auth/login: Authenticates a user with email and password, returning a JWT (JSON Web Token) for session management.
* **Student-Facing Endpoints (Requires Authentication):**
  + GET /api/v1/student/profile: Retrieves the profile for the currently logged-in student.
  + PUT /api/v1/student/profile: Updates the profile for the currently logged-in student.
  + GET /api/v1/student/recommendations/careers: Fetches personalized career recommendations.
  + GET /api/v1/student/recommendations/jobs: Fetches matched job postings.
  + POST /api/v1/applications: Submits a new job application. Request body contains jobID.
* **Employer-Facing Endpoints (Requires Authentication):**
  + POST /api/v1/employer/jobs: Creates a new job posting.
  + GET /api/v1/employer/jobs/{job\_id}: Retrieves details for a specific job posting.
  + GET /api/v1/employer/jobs/{job\_id}/applicants: Retrieves the list of applicants for a specific job.
  + PUT /api/v1/employer/applications/{app\_id}: Updates the status of a specific application.

#### Works cited

1. Project proposal DTSC4565.docx
2. Academic Departments - the College of Information - University of North Texas, accessed on October 22, 2025, <https://ci.unt.edu/academic-programs/departments.html>
3. Welcome to the College of Information | University of North Texas, accessed on October 22, 2025, <https://ci.unt.edu/>

**7.REFERENCE:**

The following list contains academic, technical, and web resources that provided foundational concepts, architectural models, or technical guidance used in the design and specification of the PathFinder system.

A. Academic and Methodological References

Sommerville, I. (2021). Software Engineering (10th ed.). Pearson Education. (Referenced for concepts on requirements engineering, functional/non-functional requirements, and project management.)

Larman, C. (2015). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd ed.). Prentice Hall. (Referenced for Use Case modeling, Sequence Diagrams, and Domain Modeling concepts.)

Kendall, K. E., & Kendall, J. E. (2019). Systems Analysis and Design (10th ed.). Pearson Education. (Referenced for methodologies used in the Customer Statement of Requirements and stakeholder analysis.)

B. Machine Learning and Algorithm References

Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). The MIT Press. (Referenced for foundational concepts in predictive modeling and algorithm selection.)

Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. (Referenced for the implementation framework of the core Recommendation Engine, likely using the Decision Tree/Random Forest classifiers mentioned in Section 1.3.)

C. Technical and Tool References

Flask Documentation. (n.d.). Flask: A Python Microframework. Retrieved from [Insert Official Flask URL Here, e.g., https://flask.palletsprojects.com/] (Referenced for the choice and implementation of the backend web framework.)

PostgreSQL Documentation. (n.d.). The World's Most Advanced Open Source Relational Database. Retrieved from [Insert Official PostgreSQL URL Here, e.g., https://www.postgresql.org/] (Referenced for the database technology supporting data persistence.)

Mermaid Documentation. (n.d.). Diagrams as Code for Developers. Retrieved from [Insert Official Mermaid URL Here, e.g., https://mermaid.live/] (Referenced as the tool/syntax used to generate the UML diagrams (SSDs, Sequence, Activity) included in the report.)