American University of Ras Al Khaimah

Name and ID:

Muhammed Irtiza - 2022005573
Rashid Ibrahim - 2022005506

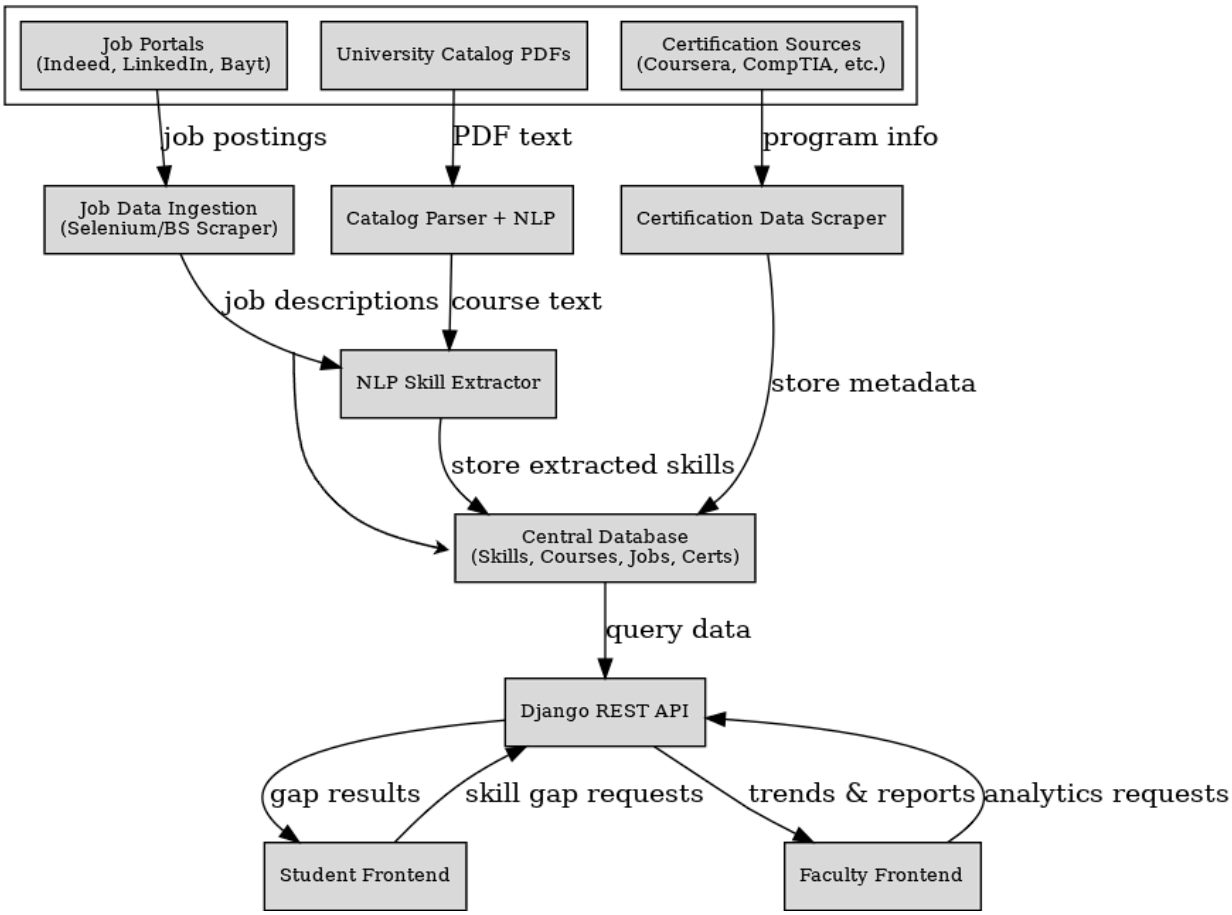
Instructor: Dr. Imad Youssef Hoballah

AT & AI

Summer, 2025

Project: Career Path Analysis Tool for Market Trend and Certification Alignment .

1 System Architecture



System Architecture

- **Integration of Course Catalog NLP:** The architecture now includes an NLP skill extraction pipeline for university course catalogs. PDF files from university catalogs (course descriptions, program outlines) are fed through a PDF text parser + NLP module, which extracts relevant skill keywords from the text. This works similarly to the job description NLP flow – using natural language processing to identify skill phrases (e.g. by part-of-speech tagging and chunking). We can leverage existing skill taxonomies to improve accuracy – for example, using a dictionary of known skill terms (from sources like the O*NET skill ontology or EMSI's open skills database) to recognize skills in text. The extracted skills from course descriptions are then stored in the central database, linked to the corresponding major or course. This means for each academic major, we can derive a list of skills taught by its curriculum.

- **Existing Job Postings Pipeline:** Job ads from job portals (such as Indeed, LinkedIn, Bayt) are collected via a web scraper (using Selenium for dynamic pages or BeautifulSoup for HTML parsing). The scraper feeds raw job posting text into the NLP Skill Extractor, which parses each job description to pull out required skills (e.g. programming languages, tools, soft skills). Using NLP here allows us to find key skill phrases

in unstructured text (e.g. identifying that a data scientist job requires "Python, R, machine learning, statistics", etc.). The extracted job skills are stored in the same central database. This unified DB now contains skill data from both course catalogs and job listings.

- **Certification Data Ingestion:** In addition, the system gathers data on professional certifications and online courses from platforms like Coursera, LinkedIn Learning, CompTIA, and PMI. A Certification Data Scraper or API integration pulls metadata for each certification – including the certificate name, provider (platform or organization), skills covered (many course descriptions list "Skills you will gain" or exam objectives), cost, and duration. For example, Coursera's catalog metadata includes skill tags and estimated learning time for each course. Similarly, LinkedIn Learning offers partner APIs to query their course catalog (alternatively, scraping can be used if API access is not available). This certification info is stored in the central database and each certification record is mapped to the skills it teaches (e.g. a Coursera "Data Science Specialization" might map to skills like Python, data analysis, SQL, etc.).

- **Central Database:** All extracted data academic skills, job-required skills, and certificate skills converge in a central database. The database schema would have tables for majors, courses, job postings, certifications, and a master list of skills, with many-to-many relations (e.g. a MajorSkills table linking majors to skills, JobPostingSkills linking jobs to skills, etc.). This design allows queries like "What skills are associated with B.Sc. Computer Science" or "Which jobs require skill X" by joining through the skills table. (The ER diagram from earlier design can be updated to include a Catalog_Major and Catalog_Course entity linked to Skill.) All data stored is cleaned and standardized (e.g. consistent skill naming) during extraction.

- **REST API and Frontends:** A Django REST API serves as the interface between the database and the front-end applications. The updated API exposes endpoints (detailed below) for retrieving skills by major, job-required skills, skill gap analyses, and certification recommendations. The Student Frontend (e.g. a student-facing web app) can request a skill gap analysis by providing the student's profile or selected major and a target job – the API responds with gap results and recommended learning opportunities. The Faculty/Advisor Frontend can use API endpoints to get aggregated insights, such as trending skills in industry vs. what the curriculum covers, to inform curriculum improvements. For example, faculty might request an analysis of skill demand trends (from job data) to see if any important skills are missing in their program. The API handles these queries by pulling from the central DB and applying any necessary logic (e.g. comparing lists of skills or computing frequencies). All components are integrated: as new job postings are scraped or new course catalog data is ingested, their skills populate the database. This unified pipeline enables end-to-end skill gap analysis – comparing the skills students acquire (from courses) against those employers seek (from jobs), and then suggesting targeted certificates to fill any gaps.

2 API Endpoint Documentation

Below is a RESTful API specification (OpenAPI-style) for the key endpoints. Each endpoint is designed to fetch or compute specific information from the system:

GET /majors/majorId/skills

- **Description:** Retrieve the list of skills associated with a given academic major. This is derived from the major's curriculum (skills extracted from course descriptions in that major). Useful for seeing what competencies a student in this major will have.
- **Path Params:** majorId (string or int) – The unique ID of the major (or major name).
- **Response:** JSON object with the major name and an array of skill names (and optionally importance or frequency).
- **Example:**

```
{
  "jobTitle": "Data Analyst",
  "company": "TechCorp Inc.",
  "skills": [
    "SQL",
    "Data Analysis",
    "Python",
    "Tableau",
    "Communication",
    "... more skills ..."
  ]
}
```

GET /jobs/jobId/skills

- **Description:** Fetch the required skills for a specific job posting. The job posting can be identified by an ID in our database (after scraping). This returns the key skills that were extracted from the job's description.
- **Path Params:** jobId (string or int) – The ID of the job posting (or an external job reference).
- **Response:** JSON with job title and the list of skills required for that job.
- **Example:**

```
{
  "major": "B.Sc. Computer Science",
  "skills": [
    "programming",
```

```
    "data structures",
    "algorithms",
    "database design",
    "machine learning",
    "... more skills ..."
  ]
}
```

GET /students/{studentId}/skill-gap?jobId{jobId}

- **Description:** Perform a skill gap analysis between a particular student (or student profile) and a target job. The student's skills can be inferred from their major and courses (or a stored student skill profile), and the job's required skills come from the job posting data. The API compares these to find which required skills the student does not yet have.
- **Path Params:** studentId – The student's ID (to look up their major or personal skill list).
- **Query Params:** jobId – The target job ID to compare against.
- **Response:** JSON listing the student's current skills, the job's required skills, and the missing (gap) skills the student would need to acquire. Optionally, it could include skills the student has that match the job (for completeness).
- **Example:**

```
{
  "studentId": "12345",
  "studentMajor": "Information Systems",
  "studentSkills": ["Excel","SQL","Data Modeling","Communication"],
  "jobId": "987",
  "jobTitle": "Data Analyst",
  "jobSkills": ["SQL","Python","Tableau","Data Modeling","Communication"],
  "missingSkills": ["Python","Tableau"]
}
```

In this example, the student has SQL, data modeling, etc., but lacks Python and Tableau which the job requires, so those are identified as the gap.

GET /students/{studentId}/recommendations?jobId{jobId}

- **Description:** Provide certificate or course recommendations to bridge the skill gap for a student targeting a specific job. This endpoint builds on the gap analysis: it looks at the student's missing skills (as identified by the skill gap endpoint) and then finds relevant certifications or online courses that teach those skills. The result is a list of recommended learning resources to acquire the missing skills. (This could also be

designed as a separate endpoint like **GET /recommendations?skills={skillList}** to directly input skills.)

- **Path Params:** studentId – ID of the student (to derive their gap from the specified job).
- **Query Params:** jobId – ID of the target job.
- **Response:** JSON array of recommended certification programs or courses. Each recommendation includes the certificate/course name, provider, and which missing skill(s) it covers. The list may be sorted by relevance (e.g. how many gap skills it addresses, or overall popularity).
- **Example:**


```
{
  "studentId": "12345",
  "jobId": "987",
  "recommendedCertifications": [
    {
      "name": "Google Data Analytics Professional Certificate",
      "provider": "Coursera",
      "coversSkills": ["Python","Tableau"],
      "duration": "6 months",
      "cost": "$39/month"
    },
    {
      "name": "Data Visualization with Tableau",
      "provider": "LinkedIn Learning",
      "coversSkills": ["Tableau"],
      "duration": "4 weeks",
      "cost": "Subscription"
    },
    {
      "name": "Python for Everybody Specialization",
      "provider": "Coursera",
      "coversSkills": ["Python"],
      "duration": "4 months",
      "cost": "Free to audit; \$49/month"
    }
  ]
}
```


In this example, the system found that the student is missing Python and Tableau, so it suggests a comprehensive certificate covering both, and also specific courses for each skill as alternatives. The metadata includes provider, estimated duration, and cost.

Note: All endpoints return data in a readable JSON format, making it easy for frontends to consume. The API would include error handling (e.g. if an invalid ID is provided) and could be secured (since student profiles might be sensitive). These endpoints enable the core functionalities: checking a major's skill outcomes, checking job requirements, analyzing gaps, and finding learning resources to fill those gaps.

3 Data Collection Plan

To power this system, we need a robust data collection strategy for both job postings and certifications. Below is the plan with target sources and methods:

1. **Job Postings Data Sources:**

   - **Indeed.com:** A major job portal with many listings. We will scrape Indeed for job postings across relevant industries. Approach: Use the HTTP client (e.g. Python requests or httpx) along with BeautifulSoup to fetch search result pages and job detail pages. If needed (to handle dynamic content or loading), use Selenium to simulate a browser. We'll specify job search criteria (e.g. by keyword and location) and iterate through result pages. Key data to collect per posting: job title, company name, location, posted date, and full job description. Indeed's HTML can be parsed to extract these fields. (Note: Indeed's official APIs for job search were deprecated, so web scraping is the viable approach. We will respect their robots.txt and rate limits to avoid blocks.) The benefit of scraping Indeed is that it gives real-time insights into job market demands – we can aggregate the postings to identify in-demand skills and job requirements.

   - **LinkedIn Jobs:** LinkedIn's job board often contains higher-end or specialized roles. However, LinkedIn aggressively moderates scraping. Approach: Utilize Selenium or a headless browser to log in (if necessary) and navigate LinkedIn's job search. Alternatively, use an open-source scraper library like linkedin-jobs-scraper which automates a headless Chrome browser and can retrieve job fields such as title, company, location, and description. For each job posting, capture the job description text (where skills are mentioned) and other details (title, company, etc.). We may need to incorporate delays and possibly proxy IPs to avoid detection. (If available, LinkedIn's official Jobs API could be used, but generally it's restricted, so automation is the fallback.)

   - **Bayt.com:** A popular job portal in the Middle East. Bayt listings are valuable especially for local market skills. Approach: Use requests/BeautifulSoup if Bayt pages are straightforward HTML, or Selenium if dynamic. We'll scrape job listings by category or search query. Data to collect: job title, company, location, description, requirements – Bayt provides comprehensive job details. Bayt's site can be navigated page by page; we will parse the job detail page for the description and requirements section where skills are often mentioned. External services

like Apify have Bayt scrapers, indicating it's feasible to get structured data (titles, companies, descriptions, etc.). We can either use those as references or implement a custom scraper.

2. **Additional Job Data Considerations:** We will ensure to store each job posting with an ID, and perhaps the industry or category, to allow filtered analyses (e.g. skills in Finance jobs vs. IT jobs). We'll also monitor new postings periodically to keep data fresh (could schedule scraping weekly). To enrich our dataset, we might also incorporate public labor market datasets or employer surveys if available, though the primary focus is on scraping live job ads (which reflect current demand). This continuous collection will let us observe trends over time and update the skill requirements in the database accordingly.

3. **Certification & Course Data Sources:**

   - **Coursera:** We will utilize Coursera's catalog to gather information on relevant certificates and courses. Coursera offers many Professional Certificates and specializations (e.g. Google's IT Support Certificate, IBM Data Science Specialization, etc.). Approach: Use Coursera's catalog API or open data if available – Coursera for Business has an API that provides course metadata including skill tags and duration. If direct API access is not an option, we can scrape course pages. For each targeted certificate/course, collect the name, provider (e.g. Coursera platform and the institution or company offering it), the listed skills it teaches ("Skills you will gain" on the page), the estimated duration (weeks or months of study), and cost (subscription or one-time fee). This data lets us map each certification to the skills it can impart.

   - **LinkedIn Learning:** LinkedIn's learning platform has many courses on technical and soft skills. Approach: If our organization has access, the LinkedIn Learning Catalog API can be used to search courses by skill and get metadata. Otherwise, we can manually compile popular certificates (e.g. Learning Paths or course series) relevant to common skill gaps. We'll record the course title, skills covered (often mentioned in descriptions), length, and access cost (LinkedIn Learning is typically subscription-based, so we might just note "Subscription" as cost).

   - **CompTIA Certifications:** CompTIA provides industry-recognized IT certifications (A+, Network+, Security+, etc.). These have well-defined skill objectives. Approach: Scrape the official CompTIA certification pages (each cert like A+ has a page detailing what domains and skills it covers). For example, CompTIA A+ covers skills in hardware, networking, and troubleshooting. We capture the cert name, and from the description or syllabus, extract key skill areas (CompTIA often lists exam domains which correspond to skills), and note the exam cost and typical preparation time or recommended experience. These certifications will be mapped to skills like "IT support", "networking basics", "cybersecurity fundamentals", etc.

- **PMI (Project Management Institute):** PMI's certifications like PMP (Project Management Professional) and CAPM are gold standards in project management. We will gather data from PMI's site for each relevant cert: the name, what skills or knowledge areas it encompasses (e.g. risk management, agile principles), the exam cost, and maintenance requirements if any. These certifications map to skills such as "project planning", "stakeholder communication", "schedule management" – which we'll include as their skill tags.

- **Other Providers:** We can also include other popular platforms or organizations as needed (e.g. Udacity Nanodegrees, Google Cloud certificates, AWS training). The metadata captured is similar: name, provider, skills, duration, cost. This ensures our recommendations have a broad coverage of learning options.

4. **Tools & Implementation for Scraping:** For scraping both jobs and certifications, we will use Python-based tooling. Selenium is ideal for sites requiring login or heavy JavaScript (e.g. LinkedIn). For simpler sites like Indeed or static pages (like certification info pages), requests + BeautifulSoup works efficiently to fetch and parse HTML content. We'll build scrapers that systematically navigate the content (with search queries or predefined URLs) and parse out the needed fields (using HTML element selectors). We must include error handling (to skip or retry if a page fails) and politeness (avoiding too frequent requests). Using libraries or APIs when available can speed up development – e.g. the linkedin-jobs-scraper library can save time for LinkedIn jobs, and Coursera's API can directly give structured data. We will also maintain mapping logic: as we extract raw text for skills, we'll normalize them (e.g. unify different spellings, remove duplicates) before saving to the database.

5. **Data Verification & Updates:** After initial data collection, we'll verify samples to ensure quality (e.g. check that the skill extractor is correctly identifying skills from job descriptions – possibly refining the NLP rules or adding to the skill dictionary if we see missing terms). The collection process is ongoing: job data should be updated regularly (jobs are posted daily) to keep the skills database current with market trends. Certification data changes less frequently but should be reviewed periodically for new programs or changes (e.g. new Coursera courses or updated exam content from CompTIA/PMI). By following this plan, we will build a rich, up-to-date dataset that powers the skill gap analysis effectively.

References

Apify. (n.d.-a). *Indeed Scraper*. Retrieved May 2025, from https://apify.com/actors/indeed-scraper

Apify. (n.d.-b). *Bayt Jobs Scraper*. Retrieved May 2025, from https://apify.com/actors/bayt-jobs-scraper

Coursera for Developers. (n.d.). *Coursera Business API – Production Dev Portal*. Retrieved May 2025, from https://dev.coursera.com/

Microsoft Learn. (n.d.). *LinkedIn Learning API – First Steps*. Retrieved May 2025, from https://learn.microsoft.com/linkedin/learning/overview

Syahrial, R. (2025). *Decoding job descriptions: How NLP unlocks the search for skills*. *Medium*. Retrieved May 2025, from https://medium.com/@rifqisyahrial/decoding-job-descriptions-how-nlp-unlocks-the-search-

Method Matters Blog. (n.d.). *Text Analysis of Job Descriptions for Data Scientists, Data Engineers, Machine Learning Engineers and Data Analysts*. Retrieved May 2025, from https://methodmatters.github.io/blog/2023/10/job-description-analysis/

Python Package Index. (n.d.). *linkedin-jobs-scraper*. Retrieved May 2025, from https://pypi.org/project/linkedin-jobs-scraper/

Scrapfly. (2025). *How to Scrape Indeed.com (2025 Update)*. Retrieved May 2025, from https://scrapfly.io/blog/scrape-indeed/