School of Computing and Mathematical Sciences

CO7201 Individual Project

Preliminary Report

Exploring HESA Data with Large Language Models for Dynamic Visualisation

Vladimirs Ribakovs

vr112@student.le.ac.uk
239062116

Project Supervisor: Heckel, Reiko (Prof.)
Principal Marker: Goes, Fabricio (Dr.)
Word Count: 1980
(excluding table contents, table labels, headings and references)

26/02/2025

DECLARATION
All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Vladimirs Ribakovs
Date: 26/02/2025

Contents

1. Aims and Objectives

The purpose of the project is to make it easier for University of Leicester to understand its organisational performance, structure, strengths, and weaknesses relative to peer institutions. Currently, this process requires manually researching HESA data about UK universities, which is a tedious, error-prone, and costly process [5].

Although HESA provides the information in downloadable formats (CSV, PDF, and Excel) and in basic tables and charts formats, those visualisations are static and limited [5]. Users must manually navigate through different pages to find the required data, and there is no guarantee that the desired data exists without extensive searching. As mentioned above HESA provides a way to visualize data but it is limited and not customizable. The idea behind the project is to create a dynamic, user-friendly dashboard powered by an LLM (GPT-J) [1] that addresses these issues by providing advanced visualisation and more efficient data comparison.

For instance, if a user asks, *"How has postgraduate enrolment changed between 2021 and 2022 in the University of Leicester?"* the dashboard will interpret this natural language query and convert it into parameters ("postgraduate enrolment," "2021," "2022"), and using these parameters to get the data, and then visualize the percentage change over the specified period.

HESA is the primary open source of data from UK higher education. It provides a massive database that consist of more than millions of records of data. The data it provides covers different aspects like institutional performance, funding, and benchmarking [5]. This project will use GPT-J to significantly improve data accessibility and provide advanced visualisation, enabling users to quickly identify trends. Additionally, users will be able to download the processed data in CSV, PDF, or Excel format for further use or presentation.

The dynamic dashboard will simplify the process of comparing university performance, that can save the time and potentially reducing cost. It will allow users to query data in plain English and receive dynamic charts, diagrams, and summary reports. Moreover, the dashboard will enable benchmarking of the University of Leicester against various Mission Groups (e.g., Russell Group, University Alliance, Million+), and it will support easy updates as new HESA data becomes available.

The motivation for this project is to make comparisons between universities more efficient and accurate that will allow the University of Leicester to quickly identify its weaknesses and address them resulting in ultimately saving time, reducing errors. For example, areas of study that are more popular or more challenging can receive additional funding, leading to improved outcomes for students and, in turn, boosting the university's reputation and popularity.

Key objectives:

- Using natural language query interpretation that will allow users to ask an English like based questions about performance metrics directly to the LLM
- Integrate and process HESA extensive data that will be extracted from CSV files to create a clean and structured dataset
- Creating user-friendly dashboard that will represent data in multiple formats like tables, charts, graph, diagrams and summary reports.

- Support comparative analysis by allowing users to benchmark against Mission Group.
- Allow users to export reports and strategic planning documents generated with their queries in CSV, PDF and Excel formats.

Main challenges:

- **Data Volume:** Despite HESA's strict quality control, it provides of more than millions of records, so it requires sanitisation and standardisation to minimize any possible errors.
- **Natural Language Query Interpretation:** Converting free-form queries into effective Pandas operations is complex and requires precise parameter extraction.
- **Integration of Multiple Data Sources:** HESA data is spread across various files and stored in formats therefore making it challenging to find a common key for benchmarking.
- **Performance Optimization:** Handling large datasets can slow down even by a simple query so using techniques such as caching, summarisation, and truncation are crucial for maintaining responsiveness.
- **Automated Data Updates:** HESA does not provide any API so creating a custom script will be needed to parse the HTML tree for updated timestamps, version tags, or new file links, and then automatically download the latest data files.

2. Requirements

Essential:

- **Natural Language Query Interpretation:** The system understands users free-from queries and retrieve, and process data based on that.
- **Integrating and Processing Data:** Extracting and preparing clean HESA data, making it usable for comparing metric performance
- **Interactive Dashboard:** It must dynamically display results in different format like tables, charts, diagrams or summary reports.
- **Comparative Analysis**: A feature that must allow to group and benchmark universities by different categories like Mission Groups
- **Data Export Capabilities:** Allow users to download generated reports and data as CSV, PFD or Excel file

Recommended:

- **Automated Data Retrieval:** Make a way to automatically update the dashboard with new data from HESA when it is available.
- **Using AI To Improve Analysis:** LLM should provide insights, trend identification, and recommendations if user asks for that.
- **Query Building Guide**: Provide a guide or pre-build example for users that are unfamiliar with the free-text queries.
- **Advanced Visualisation:** Support interactive graphs, trends projections that offer visual comparison of performance metrics.
- **Historical Data Tracking:** Tracking data changes overtime to see how organisational performance was changing relative to peers.
- **Caching for Frequent Queries:** Implement caching to perform repeated queries more efficient that will result in better performance

- **LLM Interpretation Feedback:** Allow users to see how their query was interpreted by the system (Pandas) to improved data transparency and remove bias factor
- **Queries Logs:** Log out user queries to improve future prompts tuning
- Data Overload protection: Implement a safeguard that prevent outputs of big queries to potentially crush the system
- **CSV File Validation:** Make a mechanism to automatically sanitize new raw HESA data (checking for correct format and size anomalies) to prepare it for the data processing.

Optional:

- **Predictive Modelling:** Using historical data and machine learning (ML) to predict future universities performance metrics
- **Chatbot Integration:** Include an AI chatbot assistant to help user to refine their queries dynamically, in real time.
- **API for External Use:** Create an API so that external and third-party application can use dashboard's functionality
- **Collect feedback:** Implement mechanisms that would collect users' feedback through the application surveys to continuously
- **Data Storage Scalability**: If HESA data volume grows beyond local CSV storage it will be migrated to a more scalable database solution such as (PostgreSQL or SQLite)
- **Multi-User Concurrency:** Allow users to submit multiple queries simultaneously by using asynchronous job queues (such as Celery) and session management.
- **Deployment**: Develop a deployment plan starting with local deployment and extending to containerization (Docker) and orchestration (Kubernetes) for future scalability.

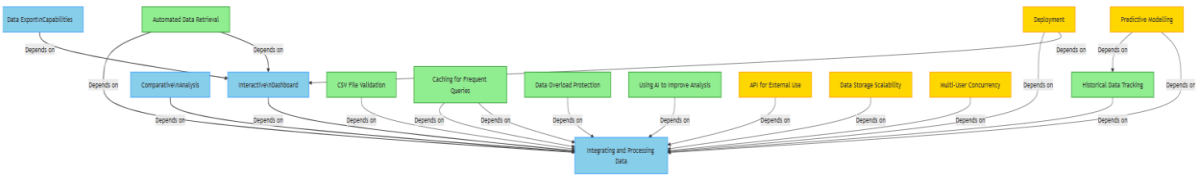

Figure 1: Hierarchy of the Project Requirements (Natural Language Query Interpretation)

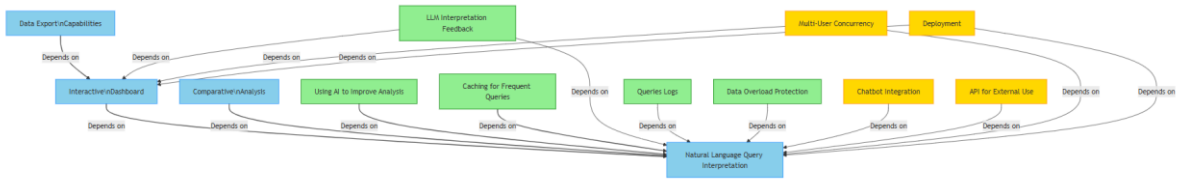

Figure 2: Hierarchy of the Project Requirements (Integrating and Processing Data)

3. Technical Specification

System Architecture:

- **Backend**: Built with Django (Python) that integrates GPT-J via Hugging Face Transformers for LLM functionality [1], [6].
- **Data Processing:** Using Pandas for data cleaning and transformation [7].

- **Visualisation:** Using Matplotlib to generate dynamic charts (graphs, pie charts and diagrams) that can be displayed on the dashboard.
- **Frontend:** Tailwind CSS that will provide modern looking interface for desktop [8].
- **Data Storage:** Initial using local CSV files but consider moving to scalable solution if data grows rapidly (PostgreSQL or SQLite)

Key Components and Their Roles:

- **CSV Data Management:** Data will be separated into raw (/data/raw_files/) and cleaned (/data/cleaned_files/) folders with CSV files. Scripts will transform raw CSVs into cleaned, standardized formats for analysis.
- **LLM Query Interpretation:** The LLM (GPT-J) will process natural language queries and outputs structured instructions that will be used for data retrieval and analysis.
- **Visualization Engine:** Matplotlib generates charts based on the Pandas-processed data to the user.
- **User Interface:** A tailwind-driven dashboard that will allows query input, displays analysis, and supports multiple output formats.

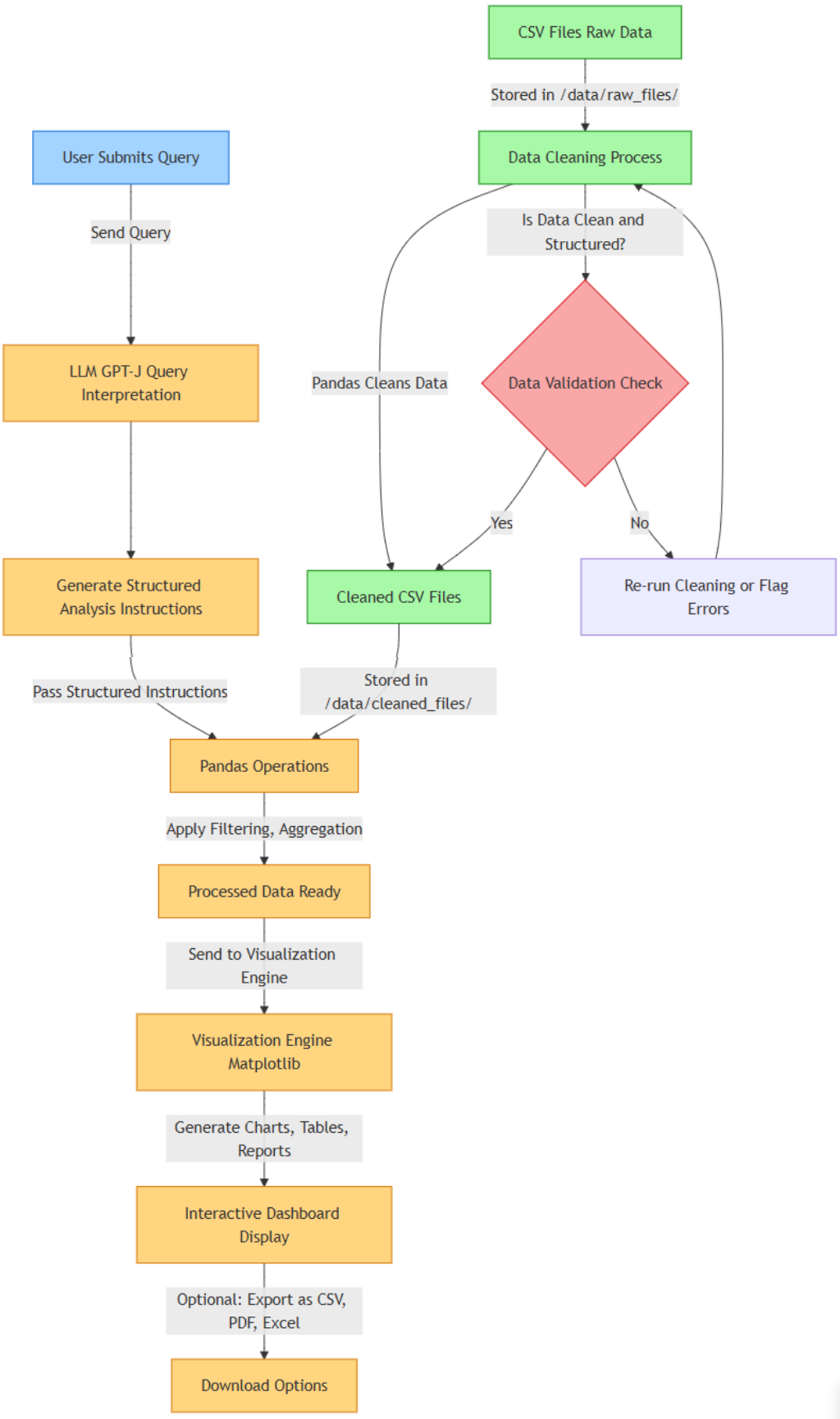

Figure 3: High-level system architecture illustrating the data flow from ingestion to visualisation.

Dependencies:

Primary	Additional	External Tools
Django (5.1.6) - Web framework used for building the backend of the dashboard.	PyTorch (2.6.0) - Required by Transformers for running GPT-J, CPU-only (*choosing CPU-only or GPU version based on available hardware)*	Tailwind CSS - For building the modern, desktop-focused frontend User Interface (UI).
Pandas (2.2.3) - Essential for data extraction, cleaning, integration, and processing of HESA CSV data.	scikit-learn (1.6.1) - Support optional predictive modelling features by providing machine learning tools.	Node.js (*any latest version*) – For local compilation of Tailwind CSS (*in case of custom styles*)
Matplotlib (3.10.0) - Used to generate visualizations (charts, graphs, diagrams) embedded in the dashboard.	Requests (2.32.3) - HTTP requests, such as if you need to scrape or fetch data from web sources.	PostgreSQL (*Optional requirement*) - For scalable data storage if the CSV-based approach becomes insufficient as data volume grows.
Transformers (4.48.3) - Provides integration with GPT-J (via Hugging Face) for natural language query interpretation.	SQLparse (0.5.3) - Useful if you need to parse SQL queries in future enhancements.	Celery (*Optional requirement*) - For handling asynchronous tasks, such as supporting multi-user concurrency for processing large queries
Huggingface-Hub (0.28.1) - Supports model management and retrieval needed for GPT-J.		

Table 1: Project Dependencies by Category

Additional dependencies may be introduced as the project evolves.

4. Requirements Evaluation Plan

Evaluation Criteria:

- **Accuracy and Clarity of Query Interpretation:** Measure how accurately the LLM will convert free-form queries into correct data operations. This will be tested by using sample queries with expected outcomes and comparing the system's results to the expected result. Also, user feedback will be used to understand if interpretation is working as expected [1], [9].
- **System Performance:** Access how much time it will take to process a query from query submission to visualisation output. Monitoring data processing efficiency on large datasets and considering time taken for cleaning, merging and running analytical operations to ensure the system respond is within acceptable time frame.

- **Usability of the Interactive Dashboard:** Using user surveys like System Usability Scale (SUS) scores to evaluate the ease of use. Observing how quickly users can located different features like query input, results display and exporting options.
- **Data Export Functionality:** Verify that exported files (CSV, PDF, Excel) are complete, accurate, and properly formatted.
- **Comparative Analysis Capabilities**: Check that system correctly group and benchmark universities against selected categories like Mission Groups. This will be tested by comparing the dashboard's output of expected results against manually computed results or benchmarks to check for accuracy.

Testing Methods:

- **Unit Testing:** Create unit tests for individual components that are validating CSV files, perform data cleaning and converting queries into Pandas operations. This can be done using Django in-build testing framework [6].
- **Integration Testing:** Simulating end-to-end scenarios where query is submitted and processed through all the steps, from being a raw CSV file to visualizing the data on the dashboard.
- **User Acceptance Testing (UAT):** Engage a small group of people to test the dashboard and collect feedback on usability, clarity of query interpretation and overall performance
- **Regression Testing:** After performing updates or enchantments re-run existing tests to confirm that the previous functionality works as expected.

Performance Metrics:

- **KPIs (Key Performance Indicators):** Define target performance metrics such as a query response time under 3 seconds and at least 90% accuracy in interpreting queries. While testing keep monitoring errors rates, system crashes and note any changes.
- **Documentation of Test Cases:** Keep track of a documents of test where each test case with a sample query, expected outcome, and actual outcome. This will be used to verify that the system meets all requirements.
- **Continuous Monitoring:** Using logging and performance tools to keep track of system performance during tests.

Documentation:

- Tables and Diagrams: Creating a detailed table that would include different system evaluations criteria alongside target values and actual outcome:

Evaluation Criterion:	Target Value	Actual Measured Value	Comments
Query Response Time	Under 3 seconds	2.5 seconds	Additional notes

- Sample Test Cases: Provide detailed examples of test cases both unit tests and integration tests. Including things like test name, input data, expected output, actual output and evidence (screenshots)
- Feedback Reports: Received feedback from users will be analysed and summarized in a table that would explain what type of issue exists, priority of the issue, suggested improvement and status.

Issue	Severity	Suggested Improvement	Status
Slow responses	Medium	Caching out repeated queries	Pending

5. Background Research and Reading list

HESA Data and Benchmarking in Higher Education:
Exploring how HESA data is used for analysing institutional performance, decision-making and benchmarking in UK universities. This will help to understand the nature of data and potential challenges in working with it. [1], [5].

Large Language Models (LLMs) in Data Processing:
Review recent updates on LLM with focus on GPT-J model and applications where it is used for interpreting natural language queries. Dive into discussions of LLMs performance, prompt design, and methods for converting queries into structured data operations [1], [2], [3], [9].

Data Extraction and Cleaning Techniques:
Study methodologies for web scraping, CSV file processing, and data cleaning using Pandas (Python library). This will include handling large datasets, ensuring data quality, and automating data updates [4].

Documentation and Technical Guidelines:
Read through the official documentations of key tools and frameworks that will be used in the project such as Django, Pandas and Tailwind. This ensures that the project follows best practices [6], [7], [8].

Performance Monitoring and Logging:
Guides and best practices on monitoring of Python applications in the real time. For example, Prometheus, Grafana, or Sentry for logging and error tracking [10], [11], [12].

6. Time-plan and Risk Plan

Time Plan:

Phase	Dates	Key Tasks
Phase 1: Project Initiation & Planning	February 11 – February 14	• Analyse the provided GitLab repository (test access via SSH key and perform a test commit). • Create the development environment and virtual environment • Define the project scope, goals, and objectives. • Identify inefficiencies in the current manual data

		analysis process (both time-consuming and error-prone). • Enhance the university's ability to benchmark performance against peer institutions. • Finalize and submit the Project Description
Phase 2: Research, Data Analysis & Preliminary Report	February 15 – February 28	• Gather and read academic papers, reports, and online sources on HESA data analysis, benchmarking in higher education, and LLM usage in data processing. • Identify research gaps that the dashboard should address. • Determine what data HESA provides and what additional data is needed. • Collect necessary HESA CSV files and perform an initial analysis. • Assess data quality, volume, and potential challenges • Finalize and submit the Preliminary Report
Phase 3: Iterative Development & Prototyping	March 1 – April 4	• Building and testing script to extract, clean, structure data and ensure that it is stored properly. • Connect the LLM (GPT-J) so it coverts free-text queries into structured operations • Creating interactive dashboard that supports multiple format outputs • Implementing features such as comparative analysis (grouping by Mission Groups) and data exporting • Prepare and conduct for Principal Marker Interview
Phase 4: Final Integration, Testing, Documentation & Viva	April 5 – May 23, 2025	• Optimizing project performance and UI improvements • Updated documentation, finalizing final report draft

		and preparing for final report. • Conduct unit, integration and user acceptant testing • Finalizing code improvements and making sure the code meets all the requirements • Submtting the final version of final report and system code. • Prepare and attend for Viva presentation and final evaluation.

Table 2: Time Plan with Phases, Dates, and Key Tasks

A Gantt chart based on the current plan has been included in the report which might be revised over time

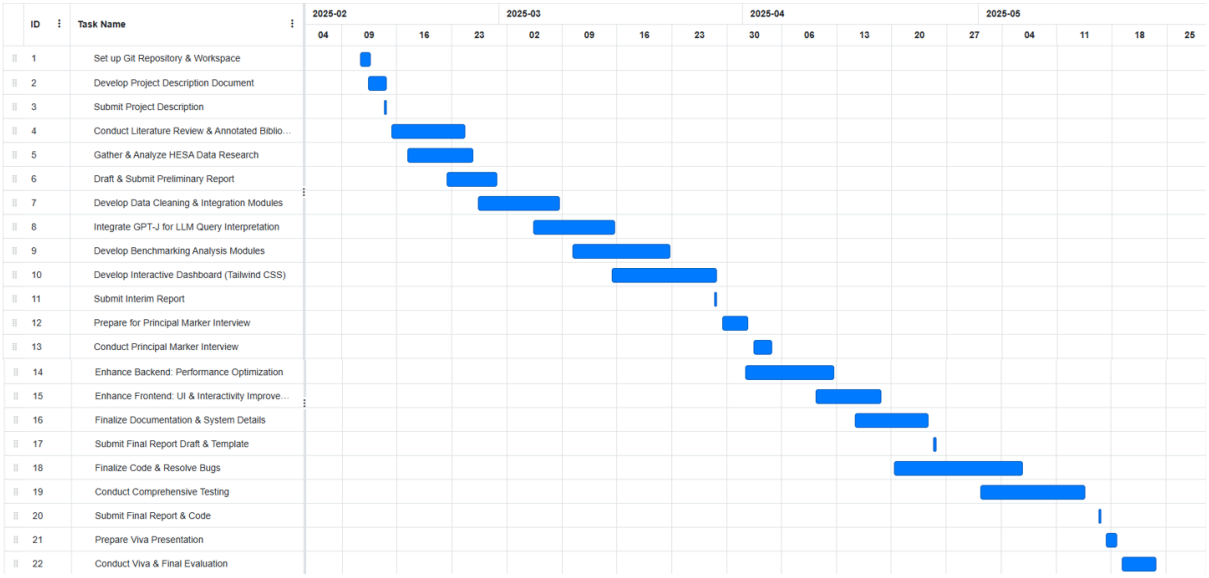

Figure 4: Gantt chart of the project timeline (initial plan).

Risk Plan:

Risk Assessment, Impact, and Probability

Risk Category	Risk Description	Probability	Impact	Mitigation Strategy
Data Integration Complexity	Difficulty in merging and cleaning multiple CSV files due to inconsistent formats or missing data.	Medium	High	Use robust data cleaning methods (Pandas scripts), standardize CSV formats, and perform iterative testing on sample datasets.

Inaccurate Query Interpretation	LLM (GPT-J) may misinterpret free-text queries, leading to incorrect data analysis.	Medium	High	Calibrate the LLM with test queries, implement a feedback loop (e.g. display corresponding Pandas code), and refine prompts based on user feedback.
Performance Bottlenecks with Large Data	Processing millions of records may slow down query response times.	High	High	Implement caching for frequent queries, optimize data summarization techniques, and consider chunked processing for large datasets.
Automated Data Update Failures	The custom script for scraping new data from the HESA website might fail if the HTML structure changes.	Medium	Medium	Create an error handling into the script, schedule regular checks, and develop contingency plans for manual data updates if necessary.
User Interface/UX Challenges	Dashboard interface may not be intuitive or responsive, affecting user satisfaction.	Medium	Medium	Conduct iterative user testing, incorporate design feedback, and refine UI elements for clarity and ease-of-use.
Time Constraints and Scope Management	Tight project timeline may risk incomplete implementation of advanced features.	High	High	Prioritize core functionalities early, set realistic milestones, and continuously monitor progress to adjust scope if necessary.
Data Security and Confidentiality	Ensuring secure handling of sensitive HESA data.	Low	Medium	Follow institutional data policies, use secure storage practices, and implement proper access controls.

Table 3: Risk Assessment, Probability, Impact, and Mitigation

Risk Monitoring and Control

- **Regular Risk Reviews:** Do weekly reviews to understand the status of current risks and detecting new risk as project progress
- **Monitoring Metrics:** Tracking key performance indicators such as query response time, error rates, system throughput to identify areas for improvement
- **Contingency Planning:** Create a backup plan such as manual data updates to address critical issue when automatic update stops working.
- **Documentation:** Keep updating risk logs to keep track of each risk, its impact, action and migration status.

- **Feedback Integration:** Using user feedback to continuously update risk mitigation plan.

7. References

[1].EleutherAI, 'GPT-J 6B', Hugging Face, 2021. [Online]. Available: https://huggingface.co/EleutherAI/gpt-j-6B (Accessed: 22 February 2025).
[2]. Brown, T. B. et al., 'Language Models are Few-Shot Learners', in Advances in Neural Information Processing Systems, 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (Accessed: 22 February 2025).
[3]. Vaswani, A. et al., 'Attention is All You Need', in Advances in Neural Information Processing Systems, 2017, pp. 5998–6008. [Online]. Available: https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf (Accessed: 22 February 2025).
[4]. Mitchell, R., *Web Scraping with Python: Collecting Data from the Modern Web*, 2nd ed., O'Reilly Media, 2018.
[5]. Higher Education Statistics Agency, 'HESA Data and Analysis', Higher Education Statistics Agency, 2023. [Online]. Available: https://www.hesa.ac.uk/data-and-analysis (Accessed: 22 February 2025).
[6]. Django Software Foundation, *Django Documentation*, 2023. [Online]. Available: https://docs.djangoproject.com/en/4.2/ (Accessed: 22 February 2025).
[7]. McKinney, W., *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, 2nd ed., O'Reilly Media, 2017.
[8]. Tailwind Labs, *Tailwind CSS Documentation*, 2023. [Online]. Available: https://tailwindcss.com/docs (Accessed: 22 February 2025).
[9]. Wolf, T. et al., 'Transformers: State-of-the-Art Natural Language Processing', in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, 2020.
[10]. Prometheus Authors, *Prometheus Documentation*, 2023. [Online]. Available: https://prometheus.io/docs/introduction/overview/ (Accessed: 23 February 2025).
[11].Grafana Labs, *Grafana Documentation*, 2023. [Online]. Available: https://grafana.com/docs/grafana/latest/ (Accessed: 23 February 2025).
[12].Sentry, *Sentry for Python*, 2023. [Online]. Available: https://docs.sentry.io/platforms/python/ (Accessed: 24 February 2025).