



Diversity in Computing Design Document



Student Name: Daniel Giedraitis
Student Number: C00260331
Supervisor: Dr Chris Staff
Academic year: 2023/2024

Table of Contents

1. Abstract	2
2. Introduction	3
2.1 Background.....	3
2.2 Problem Statement.....	3
2.3 Project Objectives.....	3
3. Technologies	4
3.1 Programming Languages.....	4
3.2 Natural Language Processing (NLP).....	5
3.3 GUI Development.....	6
4. System Architecture	8
4.1 Data Collection.....	8
4.2 Text Analysis.....	8
4.3 Correlation Analysis.....	8
4.4 Interface Development.....	8
4.5 Machine Learning.....	8
5. Application Flow Chart	9
6. Database	10
6.1 Database Hosting.....	10
6.2 User Registration.....	11
6.3 Data Encryption.....	11
7. Class Diagram	12
8. System Sequence Diagram	13
9. Challenges Faced	14
10. Mock-up GUI Screens	15
11. Bibliography	17

1. Abstract

The project aims to investigate the relationship between university computing degree program descriptions and gender balance, with a focus on increasing diversity in the field. Based on the research conducted by Zanardi et al. (2023), which suggests that the way universities communicate their values in program descriptions may impact gender balance, this project seeks to design and build a system that can automatically analyze text in publicly available course and module descriptions. It will also collect recent gender and ethnicity data from Irish institutions to determine if correlations exist between the textual content and diversity. The system will utilize technologies such as Java, Python, NLTK (Natural Language Toolkit), GUI (graphical user interface) development, and machine learning toolkits. This document outlines the design and architecture of the proposed system, detailing the technologies, system architecture, database structure, class diagrams, and system sequence diagrams. The results will contribute to efforts to improve diversity and inclusion in computing education.

2. Introduction

2.1 Background

The field of computing is experiencing substantial growth, and universities play a vital role in preparing students for careers in this field. However, gender balance and diversity within computing programs remain a significant challenge. [1] Research conducted by Zanardi et al. (2023) suggests that the way universities present their computing degree programs may impact the gender and ethnic balance within their student populations. This project aims to explore this relationship further and develop a system to automatically analyze program descriptions and diversity data to identify potential correlations.

2.2 Problem Statement

The problem at hand is the underrepresentation of women and members of other groups in computing degree programs in the European Union (EU). While some universities have managed to achieve a better gender balance, many still struggle to attract diverse student populations. Understanding the role of program descriptions in shaping this issue is essential for addressing the problem effectively.

2.3 Project Objectives

The main objectives of this project are as follows:

1. Design and build a system that can automatically analyze text in publicly available course and module descriptions from third-level institutions.
2. Collect recent gender and ethnicity data from Irish institutions, [2] including data from the Irish Network for Gender Equality in Computing (INGENIC).
3. Investigate whether there is a correlation between the content of program descriptions and the gender and ethnic diversity of students and staff.
4. Develop an easy-to-use interface that displays the terms in descriptions that convey different senses.
5. Propose modifications to program descriptions to make them more appealing to a diverse audience while maintaining their educational content.
6. Machine learning system to predict diversity based on module descriptions.

3. Technologies

This project will utilize various technologies such as:

3.1 Programming Languages

Java

Java, a robust and versatile programming language, will play a central role in the project. It is chosen for its portability, scalability, and extensive libraries that facilitate backend development. Java will be used to build the core components of the system, manage data, and ensure the stability and reliability of the project. Its platform independence allows the system to run on various operating systems, enhancing its accessibility and usability.



Figure 1. Java screenshot

Python

Python, which is well known for its simplicity and power in data analysis and machine learning, will be leveraged for several critical aspects of the project. It is an ideal choice for natural language processing (NLP) tasks due to its rich ecosystem of NLP libraries and tools. Python will enable text analysis, including tokenization, lemmatization, and the classification of terms into 'educational,' 'social,' and 'technical' categories. Additionally, Python will be used for machine learning model development, enabling the system to predict diversity based on module descriptions and analyze correlations effectively.



Figure 2. Python screenshot

3.2 Natural Language Processing (NLP)

NLTK (Natural Language Toolkit) Library

The Natural Language Toolkit (NLTK) is a fundamental component of the project's technology stack for natural language processing (NLP). NLTK is a widely used and respected Python library that offers a comprehensive set of tools and resources for processing and analyzing human language data. In the context of this project, NLTK will serve several crucial roles:



Figure 3. NLTK Screenshot

Text Tokenization

NLTK provides powerful text tokenization tools that can break down program descriptions into individual words or tokens. This is a fundamental step in the analysis of textual data, as it allows the system to work with individual words, making it easier to categorize and analyze the content.

Lemmatization

NLTK includes lemmatization capabilities, which is the process of reducing words to their base or root form. Lemmatization ensures that different forms of words (e.g., "running" and "ran") are treated as the same word, simplifying the analysis of terms, and reducing complexity in the correlation analysis. NLTK's lemmatization functions will be employed to streamline the identification of educational, social, and technical terms.

Categorization of Terms

One of the primary objectives of the project is to categorize terms within program descriptions into predefined categories: 'educational,' 'social,' and 'technical.' NLTK's extensive language data and text analysis capabilities will be important in achieving this goal. It will assist in determining the content and context of each term, enabling the system to classify terms accurately.

3.3 GUI Development

Flask

For the graphical user interface (GUI) development, the project employs Flask, a Python web framework. Flask was chosen due to its simplicity, flexibility, and ease of integration with Python code. It allows for the creation of web-based interfaces and facilitates seamless communication between the user and the backend system.

Flask's templating system, enables the rendering of HTML templates with Python variables, facilitating dynamic content presentation. The choice of Flask enables the development of a responsive and interactive interface for users to input program descriptions and receive analyzed results.



Figure 4. Flask screenshot

HTML (HyperText Markup Language)

HTML forms the structure and content of the interface. It provides the skeleton upon which the entire GUI is built. Through HTML, elements such as text areas, input fields, buttons, and containers are defined, allowing users to input program descriptions and interact with the application.

For instance, the `<textarea>` element within the HTML form captures the user's input for analysis, while the `<button>` initiates the analysis process. The HTML structure defines the basic layout of the web page and dictates how different components are displayed.



Figure 5. HTML Screenshot

3.3 GUI Development

CSS (Cascading Style Sheets)

CSS complements HTML by defining the visual presentation and styling of the interface elements. It controls aspects such as colors, fonts, spacing, alignment, and responsiveness, enhancing the overall aesthetics and usability of the application.

The style.css file demonstrates the usage of CSS for styling the elements. The body selector defines background gradients, providing an engaging visual experience. Additionally, classes like .container, .result-container, and tags like h1 are styled to ensure readability and aesthetic consistency.

Selectors and rules within CSS enable the customization and layout of various HTML elements, ensuring a cohesive and visually appealing user interface.

By leveraging HTML and CSS in conjunction with the Flask framework, the project achieves a responsive and visually appealing interface, facilitating user interaction and providing a seamless experience for inputting/selecting program descriptions and retrieving analysis results.



Figure 6. CSS Screenshot

4. System Architecture

The system architecture will consist of the following components:

4.1 Data Collection

- Automatic collection of web-based English language program, course, and [\[3\]](#) module descriptions from Irish and non-Irish institutions.
- Methodology for collecting, cleaning, and processing institutional and course diversity data.

4.2 Text Analysis

- Tokenization and lemmatization of program descriptions to extract and categorize words into 'educational,' 'social,' and 'technical' terms.
- Analysis of the text to determine the degree of 'educational,' 'social,' and 'technological' content.

4.3 Correlation Analysis

- Automatic determination of correlation between institutional diversity data and description profiles.
- Interactive display of program descriptions, results, and terms conveying respective profiles.

4.4 Interface Development

- Development of an easy-to-use interface that allows users to interactively modify and analyze program descriptions.

4.5 Machine Learning

- To train a machine learning system using module descriptions for predicting student diversity.

5. Application Flow Chart

The application flow chart provides a structured representation of the user interactions and system functionalities within the software. Each step in the chart corresponds to a specific use case or system function, outlining the logical progression of actions within the application.

This flow chart serves as a visual guide, mapping out the user journey and system responses while considering potential alternative scenarios and error-handling processes. It aims to illustrate the cohesive and systematic flow of interactions, ensuring a streamlined and intuitive user experience throughout the application.

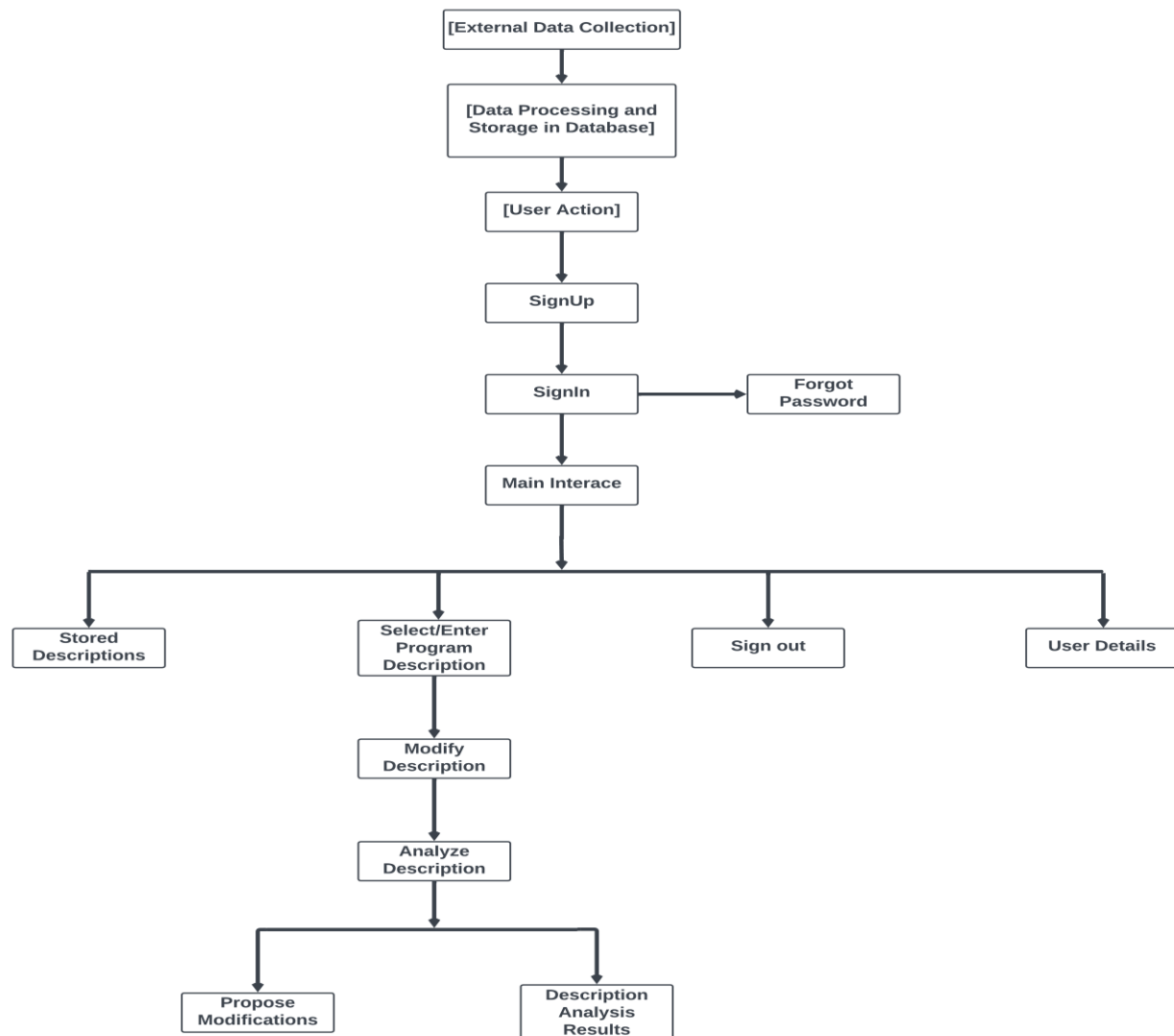


Figure 7. Application Flow Chart

6. Database

The database serves as the foundational repository for storing and managing crucial data within the application. It acts as a central hub, securely storing user information, program descriptions, and analytical outputs. This section outlines the structure, architecture, and functionalities of the database system employed in the application.

It details the database schema, specifying the tables, relationships, and attributes necessary to support the application's functionalities. Additionally, this section highlights the database management system (DBMS) utilized, emphasizing its compatibility, scalability, and security measures implemented to safeguard sensitive data.

The database design ensures efficient data retrieval, storage, and manipulation, allowing for smooth interaction with the application's core functionalities. It serves as the backbone of the application, facilitating data-driven processes and maintaining data integrity to support a robust and reliable software environment.

6.1 Database Hosting

I chose to host the database on AWS RDS (Relational Database Service) makes a big difference to how strong and adaptable the application is. With AWS RDS, particularly the managed PostgreSQL service, the app gets a reliable setup that grows as needed, automatically saves backups, and updates smoothly. There's a bunch of different ways to set up the database so it fits perfectly with what the app needs, whether it's for user data, program details, or analysis results. Picking AWS RDS boosts how well the app works, how safe it is, and how easily it can grow, making sure the database behind the scenes stays tough and ready for whatever's next.



Figure 8. PushFYI (2018). [\[4\]](#)

6.2 User Registration

User registration is an essential step ensuring secure and personalized access to the system's functionalities. It allows individuals to create unique accounts, granting them access to customized features within the application. By registering, users establish their identity within the system, enabling secure login, data management, and personalized interactions. Additionally, user registration serves as a foundational security measure, ensuring authorized access and protecting sensitive data by assigning permissions based on user roles.

6.3 Data Encryption

The security of sensitive data is important. To ensure data confidentiality and integrity, all information stored within the database is encrypted using industry-standard encryption algorithms. This encryption process converts data into an unreadable format, making it inaccessible to unauthorized users or malicious entities. Through robust encryption methodologies, sensitive information such as user credentials, program descriptions, and analytical outputs are safeguarded, mitigating the risk of unauthorized access or data breaches.

SQL for users table:

```
CREATE TABLE IF NOT EXISTS public.users
(
  id integer NOT NULL DEFAULT nextval('users_id_seq'::regclass),
  username character varying(50) COLLATE pg_catalog."default" NOT NULL,
  email character varying(100) COLLATE pg_catalog."default" NOT NULL,
  password_hash character varying(255) COLLATE pg_catalog."default" NOT NULL,
  created_at timestamp without time zone DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT users_pkey PRIMARY KEY (id),
  CONSTRAINT users_email_key UNIQUE (email),
  CONSTRAINT users_username_key UNIQUE (username)
)
```

Figure 9. SQL Screenshot

7. Class Diagram

The Class Diagram serves as a visual representation of the system's structural blueprint, illustrating the key entities and their relationships within the software architecture. It provides a high-level overview of the classes, their attributes, methods, and associations, offering insights into the system's design and organization.

Class Overview:

User Class: Represents the entity encapsulating user-related information and functionalities, including actions such as login, logout, and modifying program descriptions.

System Class: Signifies the core system functionalities responsible for analyzing descriptions, proposing modifications, and displaying analyzed data.

Database Class: The Database class not only serves as the interface for managing user data and program descriptions but also incorporates a logging mechanism. This addition allows the system to retain earlier revisions of program descriptions, enabling a historical view of changes made over time. The logging functionality captures essential information, such as when changes were applied, who made the changes (user info), and details of modifications.

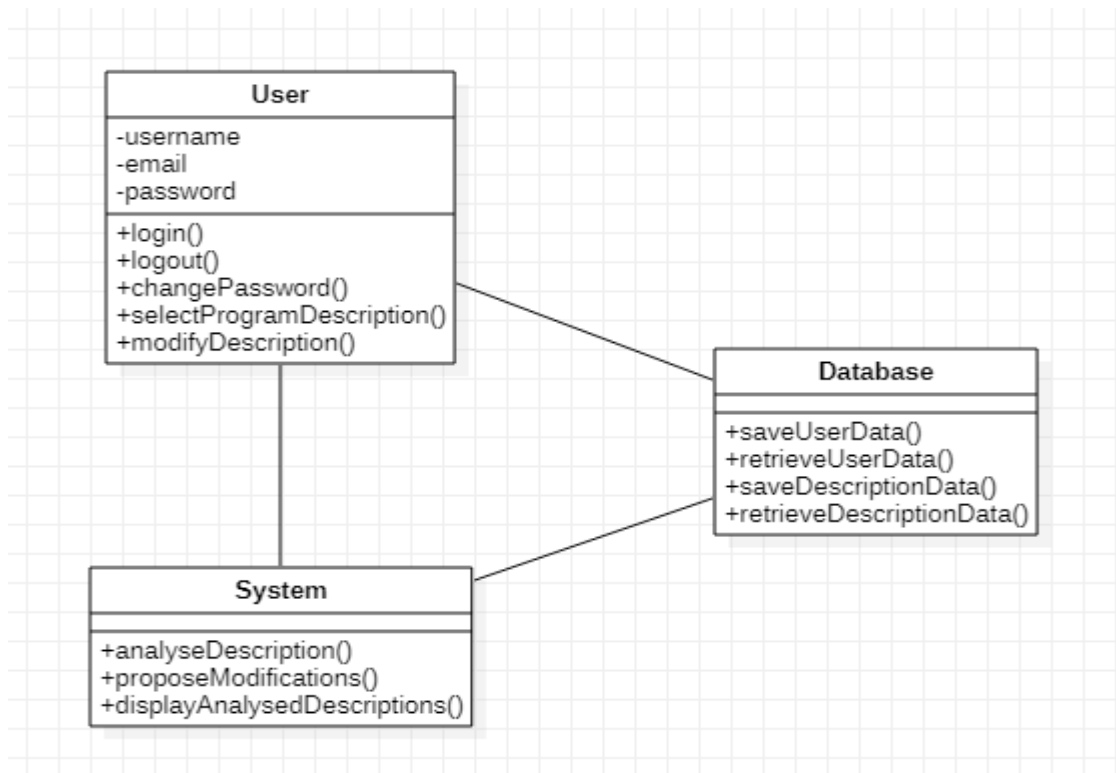


Figure 10. Class Diagram Screenshot

8. System Sequence Diagram

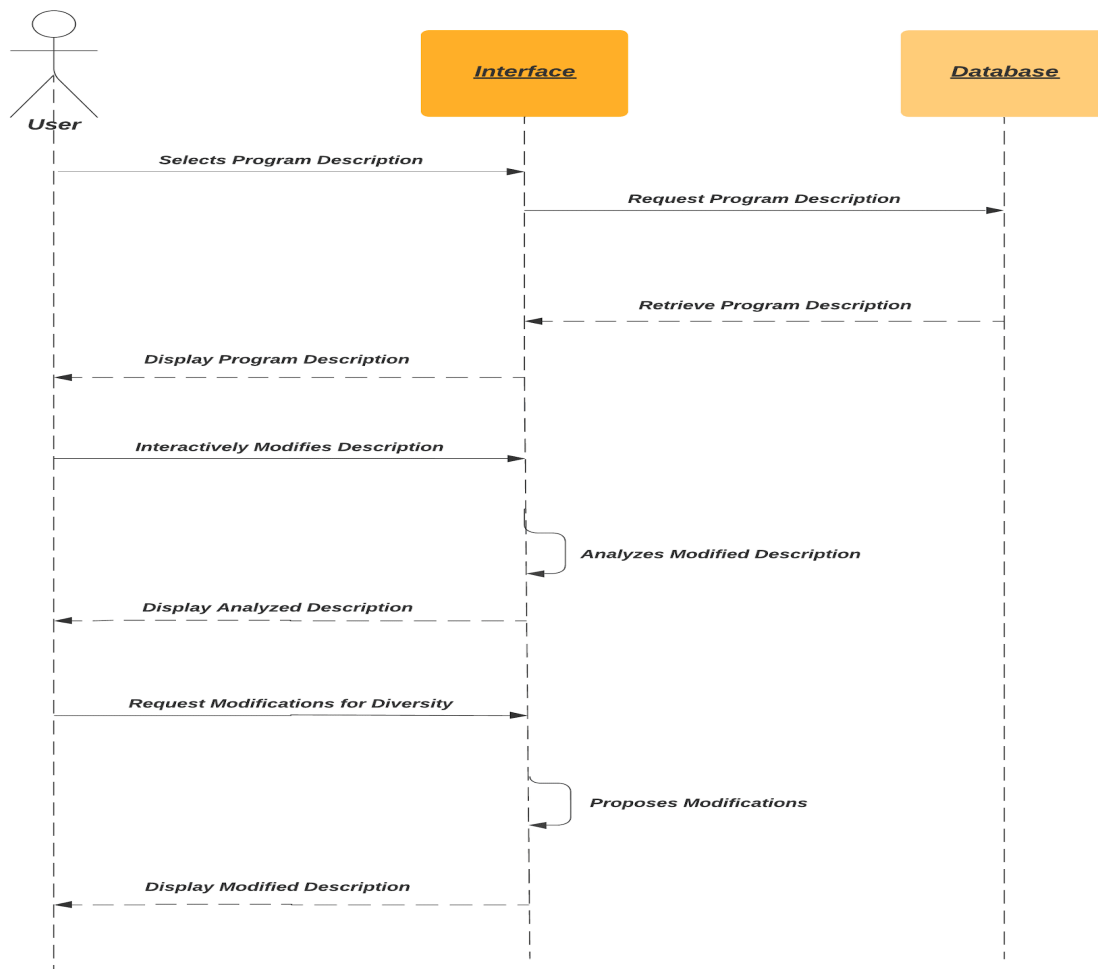


Figure 11. System Sequence Diagram Screenshot

In this sequence diagram:

1. The user selects a program description.
2. The interface sends a request to the database to retrieve the selected description.
3. The database retrieves the description and sends it back to the interface.
4. The interface displays the description to the user.
5. The user can then interactively modify the description.
6. The interface analyzes the modified description for educational, social, and technical content.
7. The analyzed description is displayed to the user.
8. The user requests modifications to make the description more appealing to a diverse audience.
9. The interface proposes modifications based on the analysis.
10. The modified description is displayed to the user.

9. Challenges Faced

Several problems occurred during the development process, ranging from technological complexities to strict regulatory standards. Addressing these challenges was critical to developing a robust and compliant system that adheres to industry standards and regulatory frameworks.

Technical Difficulties:

Handling technical issues caused substantial obstacles. Integration issues, scalability constraints, and data processing complexities need rigorous planning and implementation. These obstacles were overcome using innovative approaches, resulting in an organized design and efficient operations.

Security and Privacy Compliance:

Maintaining strict security and privacy standards, particularly in compliance with regulations such as the General Data Protection Regulation (GDPR), remained a top priority. Efforts to secure user data and implement privacy measures were prioritized. This included strong encryption technologies, and strict access limits.

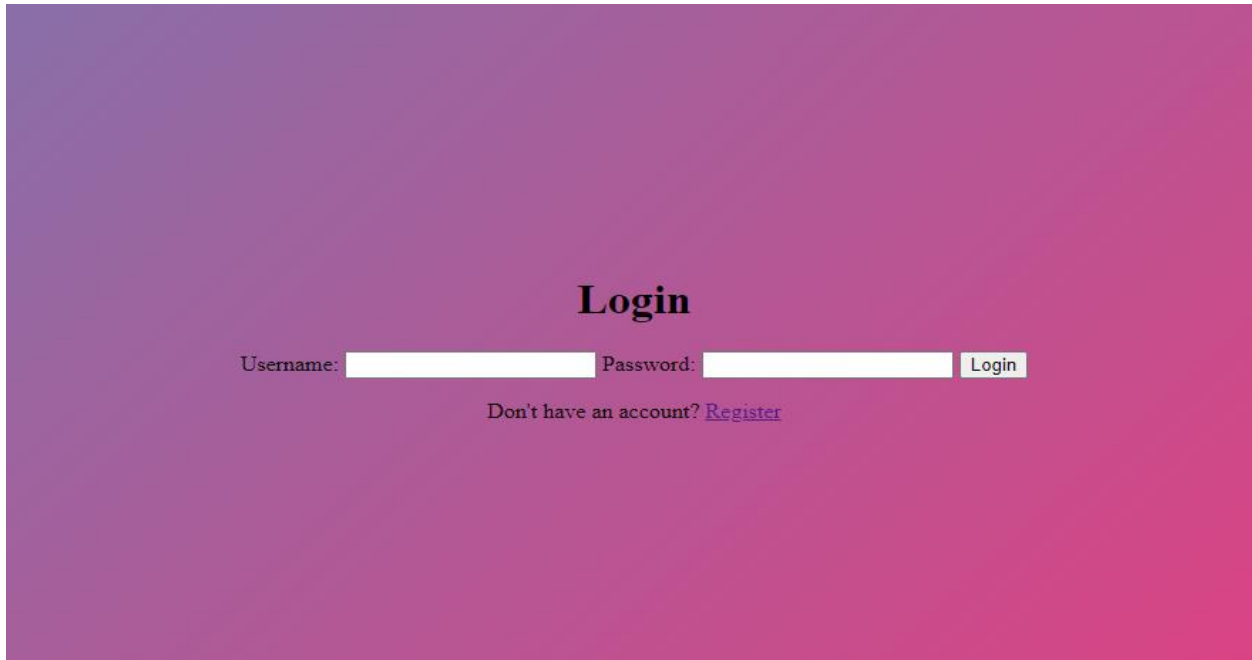
Ethical and Legal Considerations:

A crucial component of system development was navigating ethical and legal considerations. I extensively monitored system operations to ensure that ethical standards were met and that legal obligations were met. Efforts were directed toward ethical data usage and alignment with established legal frameworks governing data protection and user rights.

I took a step-by-step approach to deal with all the challenges, ensuring compliance with regulations like GDPR. Throughout the system's development, it tackled technical challenges while prioritizing security, user privacy, and ethical considerations.

10. Mock-up GUI Screens

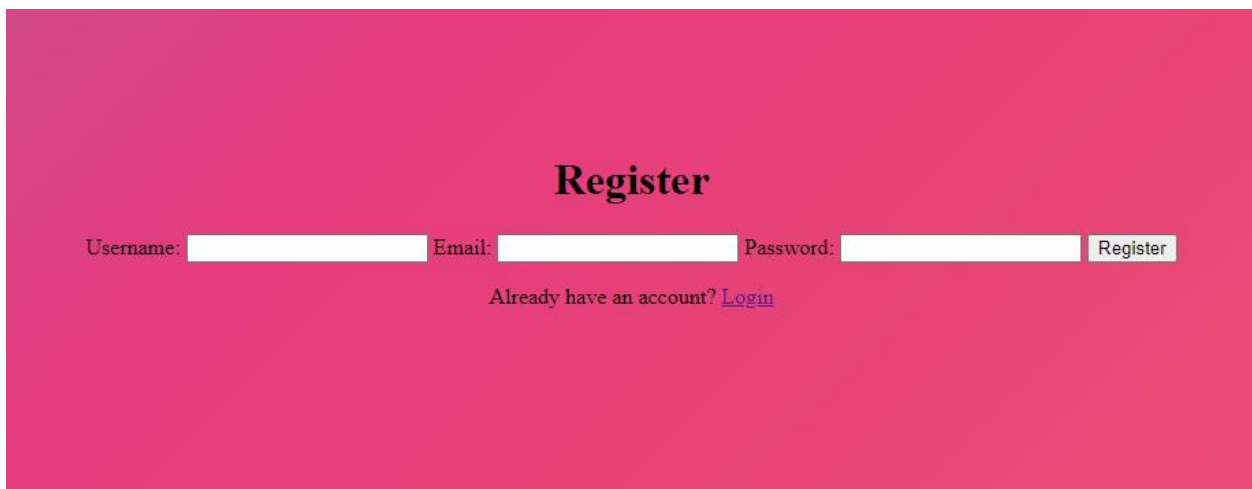
Login Screen:



A login screen with a purple-to-pink gradient background. The word "Login" is centered in a bold, black, serif font. Below it, there are two input fields: "Username:" followed by a white text box, and "Password:" followed by a white text box. To the right of the password box is a "Login" button. Below the input fields, there is a link: "Don't have an account? [Register](#)".

Figure 12. Login Screenshot

Register Screen:



A register screen with a pink-to-red gradient background. The word "Register" is centered in a bold, black, serif font. Below it, there are three input fields: "Username:" followed by a white text box, "Email:" followed by a white text box, and "Password:" followed by a white text box. To the right of the password box is a "Register" button. Below the input fields, there is a link: "Already have an account? [Login](#)".

Figure 13. Register Screenshot

10. Mock-up GUI Screens

Enter/select Description Screen:



Figure 14. Enter/select Description Screenshot

Analysis Results Screen:

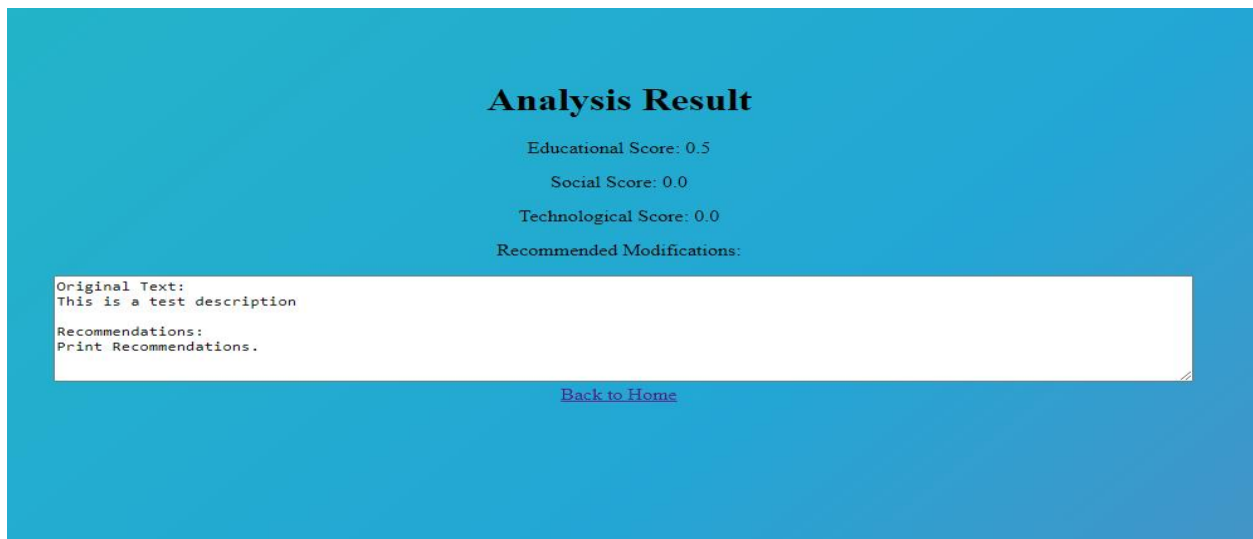


Figure 15. Analysis Results Screenshot

11. Bibliography

[1] Zanardi, I., Versino, S., Szlavi, A. and Landoni, M. (n.d.). Looking for a Major in Computing? Technical Knowledge versus Broader Social Values in Computing Majors. [online] Available at: <https://womencourage.acm.org/2023/wp-content/uploads/2023/06/womencourage2023-posters-paper84.pdf> [Accessed 17 Oct. 2023].

[2] Ingenic.ie. (2018). INGENIC Reps – INGENIC. [online] Available at: https://ingenic.ie/?page_id=331 [Accessed 12 Nov. 2023].

[3] Ingenic.ie. (2018). Courses – INGENIC. [online] Available at: https://ingenic.ie/?page_id=384 [Accessed 18 Nov. 2023].

[4] PushFYI (2018). Amazon Relational Database Service(RDS) - PushFYI - Medium. [online] Medium. Available at: <https://medium.com/@pushfyi/amazon-relational-database-service-rds-ba6ce0874c75> [Accessed 06 Dec. 2023].