Diversity in Computing Functional Specification

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

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

Table of Contents

1. Introduction

The purpose of this functional specification is to outline the requirements and features of a system designed to analyse and improve the gender balance in computing degree program descriptions. This system aims to automatically analyse text in course and module descriptions from third-level institutions in Ireland, identify correlations between description content and gender diversity, and propose modifications to make descriptions more appealing to a diverse audience. This document will provide a clear understanding of the project's scope, objectives, and functionality.

2. Vision

2.1 Project Description

The project involves the development of a software system that addresses the issue of gender imbalance in computing degree programs by analysing program descriptions and identifying correlations with gender diversity. The key components of the system include:

- **Text Analysis:** The system will automatically analyse educational, social, and technical content in course and module descriptions.
- **Diversity Data:** It will collect diversity data for staff and students from Irish Higher Education Institutions (HEIs) offering computing courses.
- **Correlation Analysis:** The system will determine if there is a correlation between the 'social' content in descriptions and the diversity of staff and students.
- **User Interface:** An intuitive and interactive interface will be designed to allow users to modify descriptions and receive suggestions for making them more inclusive.

2.2 Target Users

The system is designed to cater to a diverse set of users with varying objectives and interests in the domain of gender balance and diversity in computing degree programs. Target users include:

1. **Researchers:** Researchers in the field of gender diversity in computing education who want to analyse and study the correlation between program descriptions and diversity data. Researchers will benefit from the system's analytical capabilities and the ability to draw insights from the data to advance their studies and publications.
2. **Academic Institutions:** Irish Higher Education Institutions that offer computing degree programs and want to assess and improve their program descriptions to attract a more diverse student population. These institutions aim to create an inclusive learning environment and use the system to align their program descriptions with best practices.
3. **Students:** Students interested in pursuing computing degree programs who are seeking program descriptions that are more inclusive and resonate with their values. The system empowers prospective students to make informed choices and encourages diversity and gender balance in their academic pursuits.

3. Functional & Non-Functional Requirements

3.1 The Project Application's Core Functions

Data Collection:

- **Automatic Web Scraping:** Develop a web scraping module to automatically collect course and module descriptions from Irish third-level institutions' websites. Ensure it adheres to ethical and legal guidelines regarding web scraping.
- **Diversity Data Retrieval:** Establish connections to reliable sources (e.g., INGENIC) to obtain recent diversity data for staff and students. Implement data retrieval mechanisms to keep this information up to date.

Text Analysis:

- **Natural Language Processing (NLP):** Utilize NLP techniques, including NLTK or similar libraries, to process and analyse the collected descriptions. Implement keyword extraction, sentiment analysis, and content classification.
- **Content Classification:** Develop algorithms to classify terms within descriptions as 'educational,' 'social,' or 'technical' based on predefined criteria.
- **Profile Determination:** Calculate an overall profile for each description, quantifying the relative proportions of educational, social, and technical content.

Correlation Analysis:

- **Statistical Analysis:** Use statistical methods to correlate the description profiles with diversity data, such as gender and ethnicity breakdowns, for staff and students in the respective institutions.
- **Correlation Coefficients:** Calculate correlation coefficients (e.g., Pearson's correlation) to measure the strength and direction of the relationships. Visualize these correlations in reports.

Interactive Interface:

- **Real-time Editing:** Enable users to interactively modify course and module descriptions within the system's interface.
- **Analysis Feedback:** Provide immediate feedback to users, displaying the evolving profile of the description as they make modifications.
- **Modification Suggestions:** Offer intelligent suggestions for modifications that enhance the description's appeal to a diverse audience. Maintain the essential educational content while making it more inclusive.

Machine Learning:

- **Training Data Collection:** Collect a diverse set of module descriptions from institutions with varying degrees of student diversity (high, medium, and low).
- **Machine Learning Model:** Train a machine learning model (e.g., classification or regression) to predict the likely diversity of students based on unseen module descriptions.
- **Feature Importance:** Implement algorithms to automatically analyse module descriptions and determine which features are likely to attract more diverse student populations.

3.2 Basic Functions

User Registration and Login:

Account Creation:

- Users can create accounts by providing their name, email address, and a secure password.
- Implement email verification to ensure the authenticity of user accounts.

Authentication:

- Use industry-standard encryption methods to securely store and transmit user passwords.
- Implement multi-factor authentication (MFA) options for enhanced security.

Profile Management:

- Allow users to update and manage their personal details, including contact information and notification preferences.
- Provide the option for users to set and reset their passwords securely.

Data Management:

Secure Data Storage:

- Ensure data is stored securely using encryption and access controls to protect sensitive information.
- Implement regular data backups to prevent data loss.

Search and Retrieval:

- Enable users to search for specific course and module descriptions using filters such as institution, program level, and content classification.
- Provide options for users to sort and filter diversity data for analysis and reporting.

Version Control:

- Maintain a version control system for descriptions, allowing users to track and access previous versions for reference.
- Implement access controls to ensure only authorized users can modify descriptions and diversity data.

Reporting and Visualization:

Generate Reports:

- Develop a reporting module that generates visual reports and graphs displaying correlation results, diversity data trends, and content profiles.
- Provide customization options for users to tailor reports to their specific needs.

Export Data:

- Allow users to export reports in various formats (e.g., PDF, CSV) for further analysis, sharing, or archiving.
- Implement data anonymization or obfuscation for privacy when sharing reports externally.

3.3 Usability

User-Friendly Interface:

Intuitive Design:

- Create a visually appealing and user-friendly interface with a clean and modern design.
- Ensure responsive design to support various devices and screen sizes.

Contextual Help:

- Implement context-sensitive help features that provide information or guidance based on the user's current actions or queries.
- Include tooltips and hints for complex or less intuitive features.

Accessibility:

- Design the interface to be accessible to users with disabilities, adhering to relevant accessibility standards (e.g., WCAG).
- Provide options for adjusting text size and contrast for visually impaired users.

4. Use Case Diagram

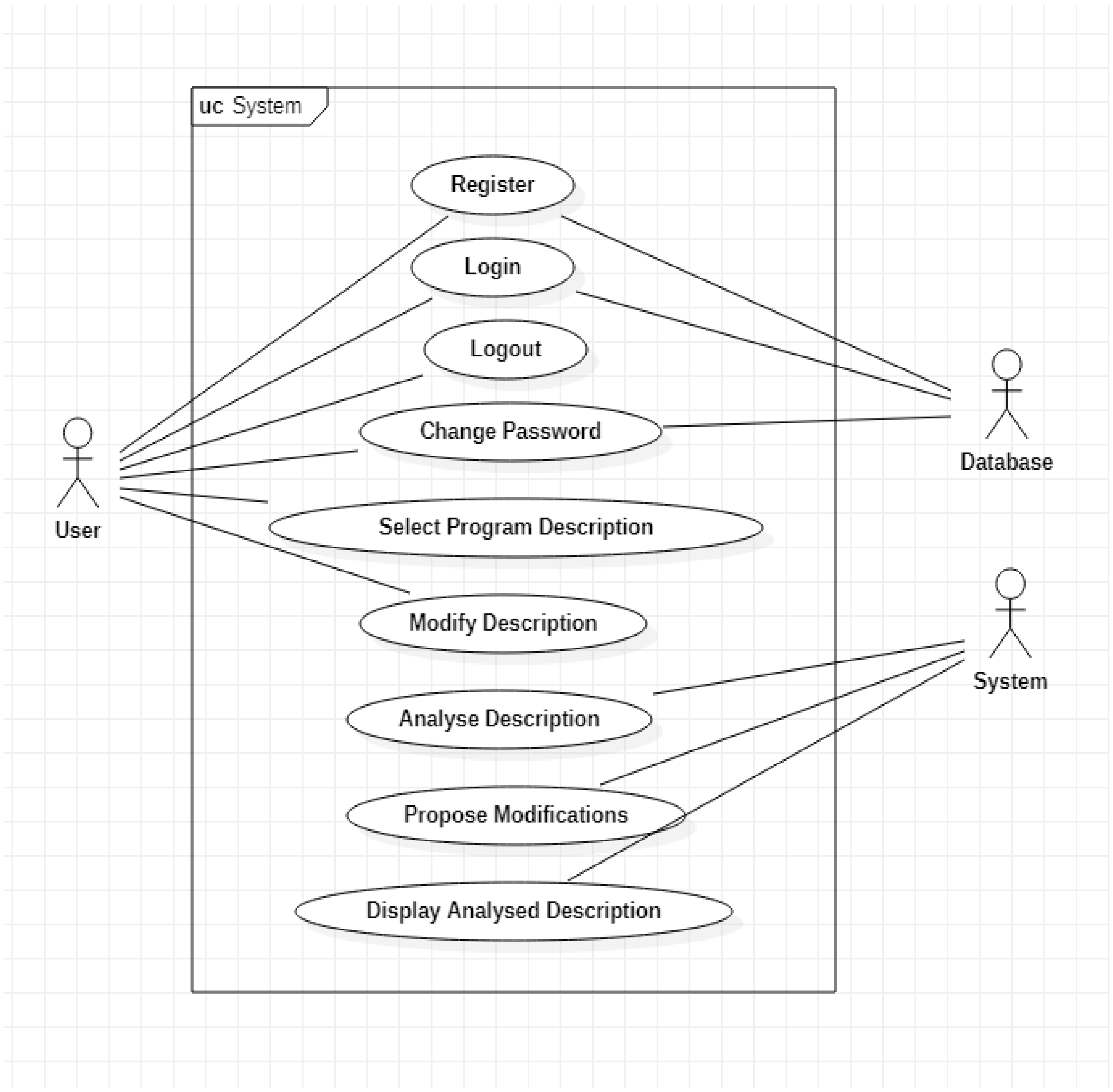

Figure 1. Use Case Diagram Screenshot

5. Brief Use Cases

The brief use cases outline the key functionalities within the system, focusing on the interactions between users and the system itself. Each use case represents a specific action or task that a user or the system can perform within the software environment. These brief descriptions provide an overview of key functionalities such as registration, login, managing user sessions, modifying account details, entering, and altering program descriptions, system analysis, proposal generation, and displaying analysed data. Each use case is associated with actors (User, Database, System) involved and their corresponding functionalities within the system architecture.

Use Case 1

Name: Register

Actor(s): User, Database

Description: Allows a user to create an account which is stored in the database.

Use Case 2

Name: Login

Actor(s): User, Database

Description: Lets a registered user to access the system by validating credentials stored in the database.

Use Case 3

Name: Logout

Actor(s): User

Description: Allows a logged-in user to terminate the current session.

Use Case 4

Name: Change password

Actor(s): User, Database

Description: Permits a user to modify their account password, which is updated in the database.

Use Case 5

Name: Select/enter program description

Actor(s): User

Description: Allows a user to input or select a program description.

Use Case 6

Name: Modify description

Actor(s): User

Description: Permits a user to make changes to a program description.

Use Case 7

Name: Analyse description

Actor(s): System

Description: Involves the system analysing the entered program description.

Use Case 8

Name: Propose modifications

Actor(s): System

Description: Changes to a program description are suggested to increase its appeal to a diverse audience while maintaining essential educational information.

Use Case 9

Name: Display Analysed descriptions

Actor(s): System

Description: Displays the analysed program descriptions to the user.

6. Detailed Use Cases

The detailed use cases provide a comprehensive breakdown of the sequential actions undertaken by users and the system to accomplish specific tasks within the software environment. Each use case elaborates on the step-by-step actions performed by users and the system, outlining the precise interactions involved in functionalities such as registration, login, session management, modifying account details, handling program descriptions, performing analysis, proposing modifications, and displaying analysed data. These detailed descriptions offer a thorough guide, highlighting the specific actions taken by each actor, whether user or system, to execute a single task efficiently. Moreover, alternatives are outlined to address potential deviations or invalid inputs during the execution of each use case.

Use Case 1

Name: Register

Actor(s): User, Database

Main Task:

1. **User Actions:**
    - User provides required registration information (username, email, password).
    - User submits the registration form.

2. **System Actions:**
    - System validates the entered information (valid email format, unique username).
    - If validation succeeds, the system stores user data in the database.
    - If validation fails, the system prompts the user to correct the information.

Alternative(s):

- If the entered information is invalid, the system displays error messages indicating the required corrections to the user.

Use Case 2

Name: Login

Actor(s): User, Database

Main Task:

1. **User Actions:**
    - User enters username and password.
    - User submits the login form.

2. **System Actions:**
    - System validates the entered credentials against stored data in the database.
    - If credentials are valid, the system grants access to the user.
    - If credentials are invalid, the system denies access and prompts the user to re-enter details.

Alternative(s):

- If the user fails to provide correct credentials, the system displays an error message and allows for re-entry.

Use Case 3

Name: Logout

Actor(s): User

Main Task:

1. **User Actions:**
    - User initiates the logout action (via a button top right of the screen).

2. **System Actions:**
    - System terminates the user session and clears the active user data.

Alternative(s):

- None.

Use Case 4

Name: Change password

Actor(s): User, Database

Main Task:

1. **User Actions:**
    - User enters the current password and a new password.
    - User submits the password change request.

2. **System Actions:**
    - The system validates the current password.
    - If the current password is valid, the system updates the password in the database.
    - If the current password is invalid, the system prompts the user to re-enter.

Alternative(s):

- If the user fails to provide the correct current password, the system displays an error message and prompts for re-entry.

Use Case 5

Name: Select/enter program description

Actor(s): User

Main Task:

1. **User Actions:**
    - User navigates to a section dedicated to entering/selecting program descriptions.
    - User inputs a program description via text entry or selection from available options.

Alternative(s):

- If the entered program description exceeds the character limit, the system prompts the user to shorten the description or provides guidance on acceptable length.
- In case of network or system issues during description entry, the system saves the entered data as a draft or prompts the user to retry submission later.

Use Case 6

Name: Modify description

Actor(s): User

Main Task:

1. **User Actions:**
   - User accesses the existing program description to be modified.
   - User edits the program description, adding, removing, or altering content as needed.

2. **System Actions:**
   - System updates the stored program description based on user modifications.

Alternative(s):

- If the user encounters an error while editing the description, the system offers an undo/redo feature or autosave functionality to revert changes.

Use Case 7

Name: Analyse description

Actor(s): System

Main Task:

1. **System Actions:**
   - System utilizes natural language processing (NLP) techniques (NLTK library) to analyse the entered program description.
   - System identifies and categorizes the content into educational, social, and technological aspects based on predefined criteria.

Alternative(s):

- If the natural language processing encounters ambiguous terms or unclear content, the system alerts the user and provides suggestions for refining the description.

Use Case 8

Name: Propose modifications

Actor(s): System

Main Task:

1. **System Actions:**
    - System generates proposed modifications to the program description aimed at enhancing appeal to a diverse audience while preserving educational content.
    - System provides suggestions or highlights specific changes that can improve inclusivity without compromising essential messaging.

Alternative(s):

- If the system-generated modifications are rejected by the user, the system allows the user to manually override or fine-tune the suggested changes.

Use Case 9

Name: Display Analysed descriptions

Actor(s): System

Main Task:

1. **System Actions:**
    - System presents the analysed program descriptions along with their categorized profiles (educational, social, technological).
    - System showcases the breakdown of content to users in an interactive and intuitive interface, displaying the identified characteristics of each description.

Alternative(s):

- None.