Mälardalen University

POLITECNICO DI MILANO

UniProgrammes

High-Level Overview of the Requirements

Sprint #0 – 13/10/2024

Summary

Project Overview 2

Core Functionality 2

Data Collection 2

Data Analysis 2

Graphical Representation 2

Requirement Validation 2

User Interface Requirements 3

Programme Tree View 3

Course Management 3

Analysis Tools 3

Technical Requirements 4

Data Integration 4

Data Processing 4

Data validation 4

Visualization 4

User Experience 4

Non-Functional Requirements 5

Performance 5

Scalability 5

Maintainability 5

Compatibility 5

Usability 5

Reliability 5

Future Considerations 6

Project Overview

Customer: Fredrik Ekstrand, Jonatan Tidare, MDU

Goal: Develop an interface for automatic collection and graphical representation of course and programme information to assist in planning and reorganizing university programmes.

Core Functionality

Data Collection

1. Automatically collect course and programme information from university websites
2. Extract and set the following attributes for each course:
    - Course-specific information (credits, educational level, course description, etc.)
    - Specific requirements
    - Main area
    - Learning outcomes
    - Class Schedule

Data Analysis

1. Categorize learning outcomes for all courses
2. Enable filtering of courses by categories such as "group work" or "report writing"
3. Implement text interpretation of learning outcomes
4. Implement a check of the lesson schedule, to avoid overlapping

Graphical Representation

1. Generate graphical descriptions of each programme
2. Display detailed information about courses and requirements
3. Enable visual planning and reorganization of courses within a programme

Requirement Validation

1. Ensure course requirements are valid when reorganizing
2. Verify the progression of learning outcomes
3. Check if the programmes meet all requirements for intended degrees (bachelor, magister, master)

User Interface Requirements

Programme Tree View

1. Display tree hierarchy of courses in a programme
2. Allow highlighting of courses by main area (e.g., ELA or DVA)
3. Enable highlighting of courses with specific learning outcomes (e.g., electronics, group work, report writing)
4. Display separation courses by period or year
5. 5 Display the links between courses and the progression of requirements, showing which courses precede and follow others.
6. Provide visual indicators (e.g., icons or colour) for critical prerequisites versus optional courses, helping users easily distinguish their importance within the programme structure.
7. Display the class schedule for each period.
8. Highlighting the courses requiring a specific course.

Course Management

1. Provide functionality to easily remove, replace, or add courses in the programme tree
2. Automatically update course requirements when changes are made
3. Highlight courses that do not meet requirements after changes
4. Include "drag-and-drop" functionality for rearranging courses within the programme tree, so users can recognize courses with immediate feedback on any requirement violations visually.
5. Add "undo" and "redo" functions for easy management of multiple changes made during programme reorganization.

Analysis Tools

1. Implement features to answer questions such as:
    a. Does this programme have a consistent progression in group work
    b. If students switch places and replace these courses, will students still acquire the specific requirements before each course starts?
2. Programme consistency in the progression of group work
3. Validity of course requirements after reorganization

Technical Requirements

Data Integration

1. Develop capabilities to interface with university websites and databases
2. Implement data extraction and parsing mechanisms

Data Processing

1. Develop algorithms for categorizing and analyzing learning outcomes
2. Implement logic for validating course and programme requirements

Data validation

1. The programme contains a consistent progression in group work
2. Detect if requirements are valid for each course
1. 3 On plan changing, detect if requirements are still valid
3. Automatically check the total credit allocation of a programme in real-time, ensuring that users do not exceed or fall short of required credits for a degree when making changes.

Visualization

1. Develop a graphical engine to render programme trees and course hierarchies
2. Implement interactive elements for course manipulation and highlighting

User Experience

1. Design an intuitive and user-friendly interface
2. Ensure responsiveness and performance when handling large datasets

Non-Functional Requirements

Performance

1. Ensure quick response times for data retrieval and analysis
2. Optimize rendering of complex program trees

Scalability

1. Design the system to handle multiple programmes and a large number of courses

Maintainability

1. Implement modular design for easy updates and expansions
2. Provide clear documentation for future maintenance

Compatibility

1. Ensure compatibility with various university data formats and structures

Usability

1. Intuitive, reactive, and easy interface
2. Not needed specific in-depth training to use the platform

Reliability

1. Consistency in programme creation
2. Consistency in choosing courses
3. Accuracy of displayed information
4. Implement user access control and version tracking so that different users (such as professors, and administrators) can make changes, and all modifications are logged for transparency and rollback if needed

Future Considerations

1. Potential for integration with university management systems
2. Expansion to include more advanced analytics and reporting features
3. Possibility of adding collaborative features for programme planning
4. Devices Compatibility
5. Incorporate predictive analytics to suggest optimal course arrangements based on historical data on student success rate, and course difficulty