Technical Documentation

for

An Interface to plan and analyze university programmes

Version 1.0 approved

Prepared by the Development Team

UniProgrammes

21 October, 2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The document describes the Architecture and design of UniProgrammes which is an interface to plan and analyze university programmes detailing the solution as per scope.

1.2 Intended Audience and Reading Suggestions

The document is for anyone who would like to understand the design aspects of the UniProgrammes Project. The key audience are:

- *Product and Project Management Teams*
- *Deployment Team*

1.3 Product Scope

The scope of the document covers architecture and design aspects required to build the UniProgrammes Project which is a tool that can be used to plan and analyze university programmes.
The following will be covered in the document:

1. *High-Level Architecture*
2. *Use-Case Diagram*
3. *UI/UX Wireframes*
4. *Database Schema*

2. Overall Description

2.1 Product Perspective

This project focuses on developing a user-friendly interface that collects course and programme data from university systems and creates a visual representation of the program structure. The key purpose of this tool is to simplify and enhance the planning, organization, and analysis of university programs. The interface will help users ensure that course prerequisites, learning outcomes, and degree requirements are met, while also providing the ability to modify the course sequence or content dynamically.

The interface supports academic planning by:

1. *Visualizing Course Progression:*
   The graphical tool will create a tree hierarchy of courses, allowing users to visualize the flow and dependencies of each course within the program. This will help in ensuring a logical and consistent progression across courses. The visualization will be based by period and/or by years.

2. *Ensuring Course Requirements:*
   By automatically updating and displaying course-specific requirements (such as prerequisites, main area, and learning outcomes), the tool will ensure that any modifications to the program maintain the integrity of the requirements for the intended degree.

3. *Learning Outcome Analysis:*
   The tool will allow users to filter and analyze courses based on specific learning outcomes (e.g., group work, report writing), facilitating a more detailed understanding of the program.

4. *Programme Customization:*
   The tool will enable the addition, removal, or replacement of courses, with real-time updates indicating whether course and program requirements remain satisfied.

5. *Educational Alignment:*
   The tool's capability to automatically highlight whether course changes maintain educational goals and degree criteria ensures that any reorganization of courses still aligns with the broader objectives of the program.

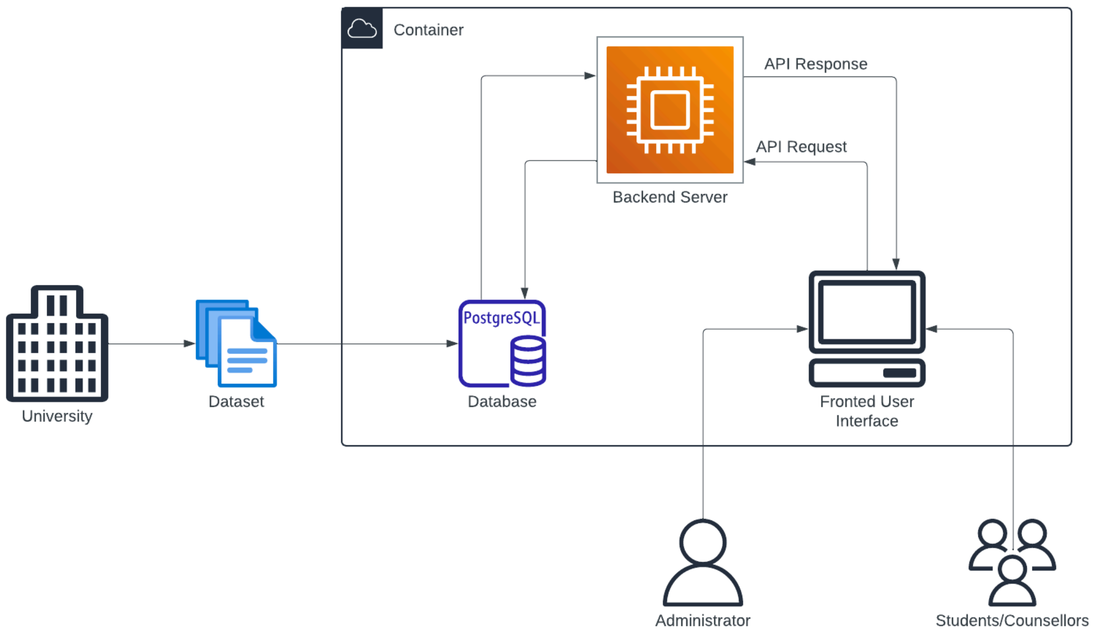
This system, when implemented, will become an essential tool for universities, enabling them to effectively plan, restructure, and ensure that students' learning paths align with the required academic standards.

2.2 Process Flow

The process flow for the development of the university program planning interface can be broken down into several key stages:

1. *Data Collection:*
    - *The interface receives data from the university, which includes course information (credits, educational level, prerequisites, main area, and learning outcomes).*
    - *This data is parsed and stored in a central database, ensuring that all relevant information is accessible for further operations.*
2. *Program Structure Creation:*
    - *Using the data, the interface generates a graphical program tree that visually represents the sequence of courses.*
    - *Courses are arranged based on their prerequisites and progression, allowing users to see the hierarchical structure and flow of the program.*
    - *The interface also highlights the main area (e.g., specific subjects like ELA or DVA) and categorizes courses based on their learning outcomes (e.g., group work, report writing).*
3. *Course Customization:*
    - *Users can interact with the program structure by modifying courses. The interface allows:*
        - *Adding new courses*
        - *Removing existing courses*
        - *Reordering courses to adjust the flow of the program*
    - *As modifications are made, the system checks the course requirements to ensure that the prerequisites, credits, and learning outcomes are still valid.*
4. *Real-time Validation:*
    - *After any changes in the course structure, the system performs an automatic validation to verify:*
        - *Course prerequisites are satisfied.*
        - *The sequence of courses maintains a logical progression.*
        - *The program still meets the overall requirements for the intended degree (bachelor, master, etc.).*
    - *If any issues arise (e.g., missing prerequisites or unmet learning outcomes), the system highlights those courses for the user to adjust.*
5. *Graphical Representation Update:*
    - *The interface updates the graphical representation of the program dynamically as changes are made.*
    - *Users can toggle between different visualizations, such as:*
        - *Viewing the full program tree hierarchy.*
        - *Highlighting specific courses based on their main area (e.g., ELA or DVA courses).*
        - *Highlighting specific courses based on specific learning outcomes (e.g., all courses that emphasize group work).*
        - *Filtering courses by specific learning outcomes (e.g., all courses that emphasize group work).*

6. *Final Output:*
    - *Once users are satisfied with the program structure, the system produces a final graphical description of the entire program.*
    - *This output can be used for further analysis or documentation purposes, ensuring that the program meets academic standards and provides a*



well-rounded learning experience for students.

2.3 System Architecture

The architecture for the university program planning interface is composed of several core components that work together to provide efficient program visualization and management. The system operates within a containerized environment that houses the backend server and a PostgreSQL database. The university dataset containing course information is loaded into the database, which serves as the primary data storage layer. The backend server handles all API requests and responses, interfacing with the frontend to provide real-time updates and interactions. The frontend user interface allows administrators, students, and counselors to visualize program structures, modify course sequences, and perform real-time validation of course prerequisites and outcomes. This architecture ensures smooth communication between the user interface and the underlying database, allowing seamless program planning and adjustments.

2.4 Use Cases

The use case diagram represents the primary interactions between users and the university program planning interface system. It highlights the core functionalities available to administrators, students, and counselors, as well as the flow of activities involved in managing program structures.

2.4.1 Actors

- Students/Counsellors:
  Students primarily interact with the system to visualize, select, and validate their course options based on program structure and requirements.
- *Administrators:*
  Administrators have broader control, managing the backend data by updating courses, modifying program requirements, and ensuring overall program compliance with learning outcomes and degree requirements.

2.4.2 Use Cases

- *Collect Course and Programme Information (Administrator):*
  The administrator inputs or uploads the course and program information into the system. This data forms the backbone for other actions within the system.

- *Categorize Learning Outcomes (Administrator):*
  The administrator classifies courses based on their corresponding learning outcomes, ensuring that each course contributes to the overall educational goals of the program.

- *Update Course Requirements (Administrator):*
  The administrator can modify the course requirements, including prerequisites and corequisites, which are reflected in the system for both students and other users.

- *Sign In/Sign Out (Both Actors):*
  Both students and administrators must authenticate by signing into the system to perform their respective tasks.

- *Analyze Programme (Both Actors):*
  Administrators can analyze the entire program structure for compliance with academic standards, while students can view and analyze their individual progress within the program.
  - *Validate Course Requirements: Ensures all prerequisites and requirements are met for both students' selections and the program as a whole.*
  - *Check Learning Outcome Progression: Tracks whether students and the program are meeting the set learning outcomes.*
  - *Ensure Degree Requirements: Ensures the program satisfies degree-related criteria, such as credit limits and mandatory courses.*

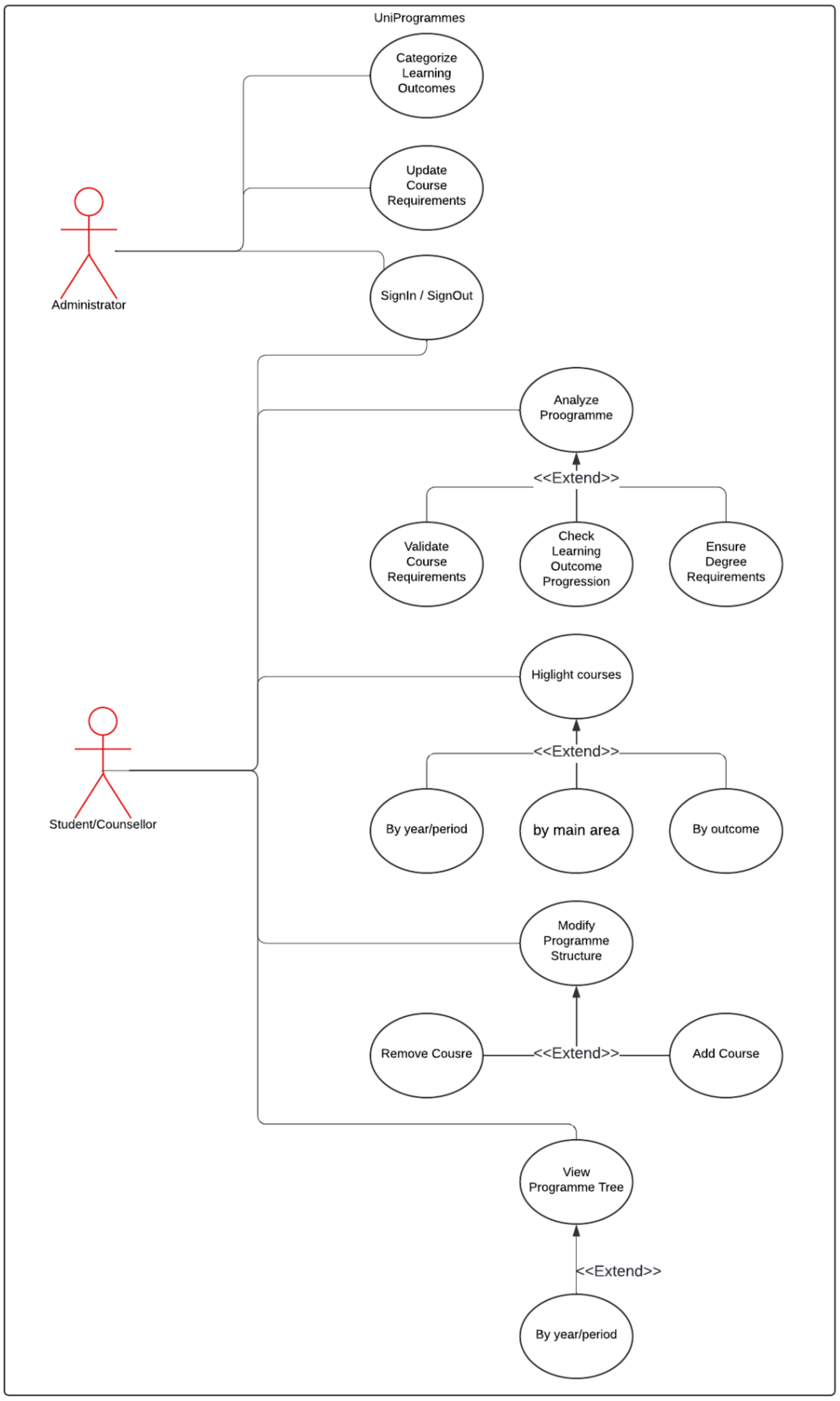
- *Highlight Courses (Students):*
  Students can highlight and organize courses based on:
  - *Year/Period: View courses by their academic year or semester.*
  - *Main Area: Group courses by area of study.*
  - *Outcome: Highlight courses contributing toward specific learning outcomes.*

- *Modify Programme Structure (Administrator):*
  Administrators can alter the program structure by adding or removing courses.

    - *Add Course: Add new courses to the program, reflected across student views.*
    - *Remove Course: Remove courses from the program, affecting course availability.*

- *View Programme Tree (Students):*
  Students can view the complete program structure in a tree format, which shows the relationships between courses and their prerequisites.
    - *By Year/Period: The program structure can be visualized according to academic timelines, aiding in student planning.*

Use-Case Diagram



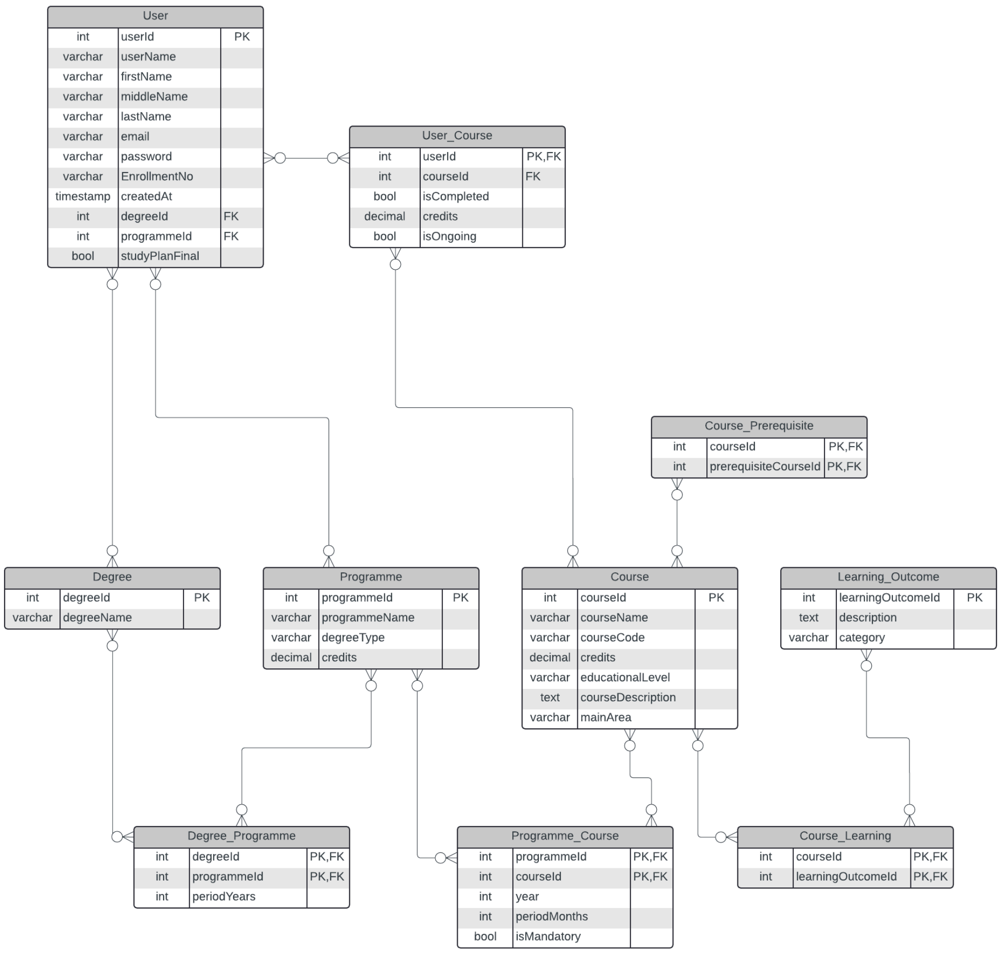
3. Database Description

3.1 Introduction

The database system presented is designed to manage a university's academic programs and courses, providing an efficient way to track user interactions, course prerequisites, degree structures, and learning outcomes. This database is central to ensuring that students and administrators can effectively plan, analyze, and manage academic progress. It includes a range of features, from user authentication to tracking course completion and integrating degree program requirements. The database supports both Students and Administrators in managing course enrollments and validating learning outcomes. The database's relational structure ensures the integrity of the data and seamless access to all necessary academic information.

3.2 Database Schema

Database Entity-Relationship Diagram



User		
int	userId	PK
varchar	userName	
varchar	firstName	
varchar	middleName	
varchar	lastName	
varchar	email	
varchar	password	
varchar	EnrollmentNo	
timestamp	createdAt	
int	degreeId	FK
int	programmeId	FK
bool	studyPlanFinal	

User_Course		
int	userId	PK,FK
int	courseId	FK
bool	isCompleted	
decimal	credits	
bool	isOngoing	

Course_Prerequisite		
int	courseId	PK,FK
int	prerequisiteCourseId	PK,FK

Degree		
int	degreeId	PK
varchar	degreeName	

Programme		
int	programmeId	PK
varchar	programmeName	
varchar	degreeType	
decimal	credits	

Course		
int	courseId	PK
varchar	courseName	
varchar	courseCode	
decimal	credits	
varchar	educationalLevel	
text	courseDescription	
varchar	mainArea	

Learning_Outcome		
int	learningOutcomeId	PK
text	description	
varchar	category	

Degree_Programme		
int	degreeId	PK,FK
int	programmeId	PK,FK
int	periodYears	

Programme_Course		
int	programmeId	PK,FK
int	courseId	PK,FK
int	year	
int	periodMonths	
bool	isMandatory	

Course_Learning		
int	courseId	PK,FK
int	learningOutcomeId	PK,FK

The database schema is composed of multiple interconnected tables that manage users, courses, programs, degree structures, and learning outcomes. The main entities in this *schema include Users, Courses, Programmes, and Learning Outcomes. Each table is connected through foreign key relationships, ensuring referential integrity and allowing for the smooth retrieval of data across various entities.*

- *User: Stores user-specific data, such as personal details and enrollment in degree programs.*
- *Course: Contains information related to each course, including credits, course codes, and educational levels.*
- *Programme: Defines the academic programs available, specifies degree types, and associates them with courses and outcomes.*
- *Learning_Outcome: Tracks various learning outcomes categorized into different academic or skill-based areas.*
- *User_Course: Manages student progress in each course, tracking completion status and credit acquisition.*

The schema also integrates Prerequisites and Course Learning, which are crucial to ensuring students follow the proper progression and meet the intended learning outcomes before moving forward.

3.3 Database Tables Description

- *User*
  Stores basic user information, including usernames, email addresses, and enrollment details. This table differentiates between students and administrators within the system.
- *User_Course*
  Tracks the relationship between users and courses. It captures whether a user has completed a course, how many credits were earned, and whether the course is ongoing.
- *Degree*
  Defines the various academic degrees offered by the university, such as Bachelor's or Master's, along with the associated degree types.
- *Programme*
  Represents each academic program, including its name, type, and credits required for completion. This table organizes programs by degree type and links to the courses available.
- *Course*
  Contains core information about each course, such as course code, name, credits, and the main area of study. It includes a description and details about the educational level of the course (e.g., undergraduate or postgraduate).
- *Course_Prerequisite*
  Manages the prerequisite requirements for each course, ensuring students complete necessary coursework before enrolling in advanced subjects.
- *Learning_Outcome*
  Stores the specific learning objectives or outcomes for each course, categorized for easy tracking.

- *Programme_Course*
  Defines the structure of academic programs by linking them to courses. This table specifies mandatory courses and their scheduling within the program.
- *Course_Learning*
  Maps courses to their associated learning outcomes, ensuring that each course aligns with the program's educational goals.