Requirement Specification Document

Course Planner

Devin Zeller & Nic Huang

1. Introduction
   * This product needs to be built because faculty members need help planning their class offerings.
   * This software will solve time conflicts, classroom conflicts, rescheduling, and provides a visualization of a schedule.
   * This product will be used by students and faculty.

1. High Level Description
   * The software will provide a user interface to allow the display of a weekly schedule of selected classes. It will hold all the class information, which is course name, course code, section number, course credit, frequency of lectures, instructor name, preferred timing, and preferred classroom. If there is conflicting information within the user's input, the software will alert the user that there is a conflict. At this time the user will be able to correct the conflicting information before proceeding with the timetable. The software will then take all the information inputted by the user and create a timetable with all the course information. This timetable will be displayed in an easy-to-read calendar format. Once the timetable is created the users will be able to make any necessary changes to courses or courses information. This will include adding a course, removing a course, and editing a course. There will also be features to easily view the list classes to view all the course information more easily.
   * The Architecture Design pattern that was used is the Model View Controller due to its attribute of introducing the concept of a ‘controller’. Since we will have to be using multiple flows of data, separating the user’s action of requesting the data and presenting the data will make the implementation easier. The controller will assist in this manner by making the system centralized. Distinguishing the separation will allow it to be more efficient at processing the data before sending it to be viewed, as well as assuring a seamless experience.

1. Detailed Description
   * Definition
     + The system will employ the use of a database to maintain, verify, and update information that the user enters into the software. To maintain the backend use of our system, an ER (Entity Relationship) diagram was created to help. This ER diagram allows us to see and understand the relationships between our data.

Diagram

Description automatically generated

* + Functional Requirements
    - The software will allow the user to input course specific parameters, such as course name, course code, section number, course credit, etc..
    - Once the software is given the appropriate information, the system will resolve any time conflicts, classroom conflicts, as well as suggest to the user the appropriate solution.
    - The user will also have the ability to add, edit, or delete courses. The system will be able to update accordingly to these actions.
    - The software will also be able to generate reports of the courses, as well as provide a list of said courses.
  + Non-functional Requirements
    - The software’s user interface will be simple and intuitive to allow any inexperienced user a full experience.
    - The system will run efficient and fast, as to not distract them from using the software.
    - Software will be cross-platform, being able to run on any desktop.
    - The software will save the courses to a database, as to provide permanence for the user.
  + Constraints
    - The software will be programmed using the Python language, which allows the use of simple, lightweight, yet intuitive libraries.
    - For the user interface, the software will utilize the CustomTkinter library.
  + Detail Design – UML Class
    - Class diagram was chosen since the concept of a ‘Course’ in our system resembled a class the most. Since the ‘Course’ itself would have attributes like a class, it made sense to use the Class diagram for our detail design.

Diagram

Description automatically generated