3-Page Plan

(Project Plan / SRS / SDS)

Project Plan

Project Goal: Design a web application that allows for the creation of a degree plan for a student based off major requirements and a student's current progress.

Group Name: TBD (To-Be-Decided)

Team Members: JT Kashuba, Noah Kruss, Logan Levitre, Zeke Petersen, River Veek

Requirements:

Must have:

- data persistence ~~of some kind (database)~~ (object pickling)
- default CIS track (prerequisite tracking – may be a LL)
- visualization (matrix view, tree view, long/short term plan, etc.)
- web interface
    - drop-downs for adding classes taken, student inputs
    - drop-down for setting current term
    - interface for viewing created schedule
- put together the 4-year plan for you based on core courses

Should have:

- exportability of plan ~~(pickling perhaps?)~~ via pickling
- take UO term availability into account
- Display what requirements have not been accomplished yet by the user

Could have:

- adjust four-year-plan based off of desired graduation data
- universal for any major
- ability to determine which classes shouldn't be taken together (difficulty values)
- Take into account multiple majors/minors

Won't have:

- log in/authentication system
- ability to directly import information for a new major

- ability to copy student data into another student object

Initial assignments (though ~2/19):

JT – Docker initial code

Noah – SRS/SDS initial plan

Logan – User Interface (web page)

River – ~~MongoDB initial code~~ Pickling initial code

Zeke – SRS/SDS initial plan

Later tasks to assign:

- Class for the major, class for storing UO_class_info (goes in the major class), prerequisites defined by a DAG

- Integration testing

- Adding features from the Should or Could have categories

Timeline:

Set up modules individually (Week 2)

Integrate modules (Week 2-3)

Add logic (Week 3-4)

Additional functionality as time allows (Week 4)

List of technologies:

Docker (containerization)

Flask (microframework for web apps)

~~MongoDB (database)~~ Python (pickling)

Python/HTML/JavaScript (primary programming languages)

Adobe XD (UI development)

SRS

The Concept of Operations (ConOps)

Current System or Situation

Once a student begins college, one of the first things they need to do after determining their declared major is to draft a four-year-plan. This four-year-plan is a theoretical guide for the student of the classes that they are going to need to take and when these classes should be taken. As of right now, when a student wishes to plan out their degree path, they have two main options. The first option is to contact their advisor for their major and schedule a meeting to construct their four-year-plan. The second option is to search for the major requirements on their own and use those to construct their plan.

Justification for a New System

While there is nothing inherently wrong with either of the approaches stated in section 2.1, each method comes with some distinct drawbacks. Regarding the first option, there is the issue that both students and advisors can become very busy over the course of a typical college term. Considering the many responsibilities already assigned to the typical college student, finding time to schedule and attend an advising meeting can be difficult and deprioritized. Regarding the second option, there is the problem of lack/loss of information. While all the information a student needs to plan out their time in college is usually available on university web pages, it is quite easy for some crucial information to be missed. To construct a robust degree-plan, the student needs to know what courses are required, what the prerequisites are, and the terms in which the courses are offered (it should be noted that it is also useful to know the difficulty level of the course).

Additionally, both options possess the same major problem; plans change over time. While it is convenient to have an initial four-year-plan to help make short term decisions, these plans are almost certain to change before graduation. This could be due to a variety of reasons, such as having to retake a course, course conflicts with other required or desired courses, or perhaps a desire to adjust the rate of major progress. When any of these possible causes for adjustment occur, they result in the student needing to repeat the degree-planning process.

By creating a system that can store a student's major progress state, along with the knowledge of what is required for the major, we effectively remove the need to regularly repeat the degree-planning process for both the student and their academic advisor when a previously unforeseen change occurs.

Operational Features of the Proposed System

- Store major course list, course requirements for the UO CIS major

- Store student course information in a database
- Use the above information to construct a degree-plan for the selected student based on the major requirements and the courses they have already taken
- Web interface for user
  - Interface for selecting/adding a new student
  - Once student is selected
    - Interface for adding classes taken (along with term taken)
    - Interface for adding current term
    - Interface displaying a visual representation of the degree-plan for the selected student

Commented [RV1]: Change to object pickling?

User Classes

Academic advisor:

- Use system to store multiple student degree paths and check progress

Student:

- Use system to store personal degree paths and check progress
- Use the display functionality to see a possible degree-plan based off current course progress

Modes of Operation

- Edit degree progress
- Save student information
- Display degree plan

Operational Scenarios (Also Known as "Use Cases")

Use Case 1: Add new courses taken to student data.

Brief description:

This use case describes how a user (either academic advisor or student) would go about updating a selected student's data to note that the student had completed some course.

Actors: Academic advisor / student user

Preconditions:

1. User selects which student to update (either through selecting pre-created students or adding new student into the system)

Steps to Complete Task:

Postconditions:

Commented [RV2]: Leave even if blank?

Use Case 2: Save student data state.

Brief description:

This use case describes how a user (either academic advisor or student) would go about saving a selected student's data after changes have been applied.

Actors: Academic advisor / student user

Preconditions:

1. User selects which student to update (either through selecting pre-created students or adding new student into the system)

Steps to Complete Task:

Postconditions:

Use Case 3: Display possible degree plan for student.

Brief description:

This use case describes how a user (either academic advisor or student) would go about displaying a selected student's possible degree plan based on the major requirements and the student data.

Actors: Academic advisor / student user

Preconditions:

1. User selects which student to update (either through selecting pre-created students or adding new student into the system)

Steps to Complete Task:

Postconditions:

SDS

System Overview

The Four-Year Plan Tracker is software that allows users to construct and visualize a long-term plan for scheduling college courses. It permits data to persist in a database while still being able to deviate and recalculate as needed depending on class completion.

The system is divided into three primary components. The first consists of the web interface, consisting of fields to enter recently completed courses and the term of completion, along with a generated matrix of suggestions for future course slates to complete the major on time. It also contains an interface for choosing which user information to load and the option for adding a new user. The second component consists of the logic for determining which future courses to suggest and when. It contains all course, student, and major objects and relates them based on user input from the web interface component (which major was chosen, supplied completed courses, etc.). The final primary component consists of the logic for saving information ~~in a MongoDB database~~ by means of object pickling such that user information may be saved and later restored as students make progress in their curriculum.

The first two components are further subdivided depending on the specific task they are being asked to do. For example, the web interface factors out the matrix visualization from the main webpage that contains user input fields. This is because the visualization is much more specialized than simple HTML forms. Additionally, the logic component breaks up class definitions to properly distinguish major information, course relationships (prerequisites primarily), and student data.

Software Architecture

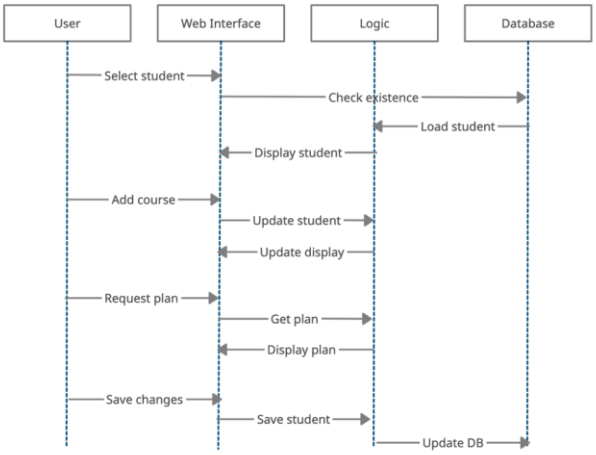

Figure 3.1 *Sequence diagram. Describes the sequence of events that take place among the major software components in the main use cases/ interactions initiated by the user.*