

Project 1: MaxGPA
CS 422 Software Methodologies
Prof. Anthony Hornof - 10-1-2026

0. Deadlines

Initial submission: Due at on Thursday, October 8, at 8PM. This will be submitted as a single PDF on Canvas. This initial submission will include everything discussed under "Initial Project Plan / SRS / SDS" at https://pages.uoregon.edu/hornof/cs422-f26/Project_Evaluation.html except that you will not submit an SRS unless you substantially modify this project description.

Final submission: Due on Thursday, Oct. 22, at 8PM. It will be submitted as a zip file on Canvas. It will include everything else discussed at:
https://pages.uoregon.edu/hornof/cs422-f26/Project_Evaluation.html

Presentations: In class on Friday, October 23. Eight minutes per group.
Please see: https://pages.uoregon.edu/hornof/cs422-f26/How_to_Present.html

1. Concept of Operations (ConOps)

The goal of the project is to build a system called *MaxGPA* that will help University of Oregon (UO) students plan their majors, and classes in their majors, to maximize their GPAs. The system will generate reports that show

- (a) The distribution of grades awarded across all required classes in a major, as a predictor of GPA upon completion of a major,
and
- (b) The instructors that give the highest grades for all required classes in a major, shown in a sequence that permits students to plan, in advance, instructors to seek or avoid.

The goal of the project is to create a *process* for showing the above information, a process that can be reused across the hundreds (<https://admissions.uoregon.edu/majors>) of majors at UO.

A sample report might start as follows:

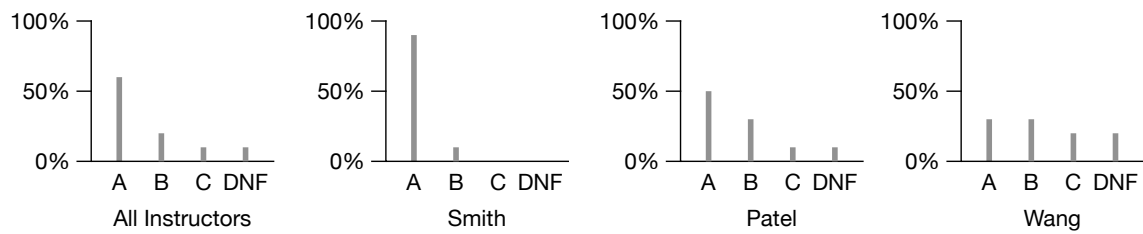
Grades Assigned Across UO Classes for a Major

Report Prepared Oct. 18, 2026, by the *Easy A* system created by
García, Lee, Singh, and Zhang for A.Hornof's Fall 2026 CS 422 Software Methodologies
Data are from 2015-2025. Degree plan is from the 2025 UO Catalog.

Major: Bachelor of Arts in Business Administration

Year 1 - Term 1

BA 101Z - Introduction to Business



MATH 111Z - Precalculus I: Functions

(graphs)

Year 1 - Term 2

EC 201Z - Principles of Microeconomics

(graphs)

MATH 241 - Calculus for Business and Social Science I

(graphs)

WR 121Z - Composition I

(graphs)

DNF includes all non-passing grades. (It might stand for “did not finish” but would more accurately be described as “did not pass”.)

Current System

Degree requirements in the UO Catalog can be viewed side-by-side, as shown in Figure 1. Grade data are not provided. (These two screenshots are from

<https://catalog.uoregon.edu/coll-business/ug-business-admin/#degreeplantext> and <https://catalog.uoregon.edu/arts-sciences/school-computer-data-sciences/computer-science/ug-computer-science/#degreeplantext>.)

Business Administration (BA/BS)		
Four-Year Degree Plan		
The degree plan shown is only a sample of how students may complete their degrees in four years. There are alternative ways. Students should consult their advisor to determine the best path for them.		
Bachelor of Arts in Business Administration		
First Year		
FALL	MILESTONES	CREDITS
BA 101Z	Introduction to Business	4
BA 199	Special Studies: [Topic]	1
Academic Residential Community or FIG seminar		
MATH 111Z	Precalculus I: Functions	4
First term of 1st-year second-language sequence		5
Credits		14
WINTER		
EC 201Z	Principles of Microeconomics	4
Second term of 1st-year second-language sequence		5

Computer Science (BA/BS)		
Four-Year Degree Plan		
The degree plan shown is only a sample of how students may complete their degrees in four years. There are alternative ways. Students should consult their advisor to determine the best path for them. Additional information may be found at the department website .		
Bachelor of Arts in Computer Science		
First Year		
FALL	MILESTONES	CREDITS
CS 122	Introduction to Programming and Problem Solving (recommended)	4
MATH 112Z	Precalculus II: Trigonometry	4
WR 121Z	Composition I	4
Core-education course in arts and letters		4
Credits		16
WINTER		
CS 210	Computer Science I	4
MATH 251Z or MATH 246	Differential Calculus or Calculus for the Biological Sciences I	4

Figure 1. A comparison of the UO Business Administration degree plan and the Computer Science degree plan.

The **UO Catalog** is an important document for this project, and for students in general. It is the official statement of what students must complete to get a degree. Students can be held to the requirements stated in the catalog when a student starts the program, or completes a program. This is one of the reasons that UO maintains an archive (at <https://registrar.uoregon.edu/registration/planning-tools/academic-catalog>). (Your new system only needs to work with the most recent catalog.)

Note that the Business Admin degree plan above is from <https://catalog.uoregon.edu/coll-business/ug-business-admin/#degreeplantext>

Justification of a New System

The current system does not show data that would help students maximize their GPAs. Figure 1 shows a side-by-side comparisons of degree plans, but not grades assigned within those plans.

User Classes

The two main users classes here include:

- (1) A student using the system to select a degree plan or instructor.
- (2) A system administrator installing the system, or updating it with new data.

Operational Scenarios (Use Cases)

There are two main use-cases, described here very briefly:

- (1) A student uses the system as described above.

- (2) A system administrator acquires new data for the system, and updates the system with the new data.

2. Functional Requirements

(Some non-functional requirements are also included in this section.)

2.1 Student User

2.1.1 *Select Major and Degree*

The system should make it clear to the student user which majors and degree programs are in the system, and permit the user any one of those.

2.1.2 *Select Years*

The system should make it clear to the student user which years of grade-history data are in the system, and permit the user to then select which span of school years to include in the report. Any span of years of data in the grade-history should be permitted. For example, if the system includes data from AY16 through AY 23 (AY16 is the academic year from Fall 2016 through Summer 2017), the user should be able to select just AY16, AY16-AY23, or AY22-AY23.

2.2 Administrative User

2.2.1 *Read-in the grade-history data.*

The system should provide a means for the administrative user to read-in the grade-history data from the .CSV files provided, as follows.

The headings shown in the first line of the grade-history .CSV file should be used to define how data are organized in the remaining lines, and thus how the data get read-in.

Before replacing the grade history data currently in the system, the system should provide a facility to permit the administrator to confirm that the data will be read-in correctly. This could be accomplished by reading-in the file until encountering a line of non-null (non-asterisk) data, and then displaying that data so that the administrator can confirm that the fields will be read in correctly. This could be as simple as:

```
TERM: 201501
TERM_DESC: Fall 2015
SUBJ: AA
NUMB: 508
...
```

And then the administrator can proceed to update the data in the system, or leave it unchanged.

To assist with testing this feature, a second grade-history data file has been added to the project directory in Canvas, which you should use to test this feature. It has an additional unneeded field. The file is at https://canvas.uoregon.edu/files/25340410/download?download_frd=1 .

Rationale: UO provides additional columns of data when requested. However, fortunately, UO appears to always use the same heading names.

2.2.2 Read-In Degree-Plan Data

The system should read-in degree plan data that is acquired from the UO Catalog. There should be one file per degree plan. Please follow this pattern for naming the files, shown here for the two degree plans in Figure 1:

Business_Administration_BA.CSV
Computer_Science_BA.CSV

The degree plan information should be acquired from the UO catalog and be formatted into a .CSV file formatted as follows.

YEAR, TERM, NUMB, SUBJ, TITLE
1, 1, BA, 101Z, Introduction to Business

Note that this example corresponds to the first class shown in Figure 1. YEAR refers to the year in the degree program, and TERM is 1 for Fall, 2 for Winter, and 3 for Spring.

The creation of these .CSV files may involve some human steps, which should be clearly described in the documentation for the system administrator. Note that Microsoft Excel does a good job of parsing web page data if you just copy the contents of a web page and paste it into an Excel spreadsheet. (Students should have access to Excel at <https://software.uoregon.edu/> . You can assume the person grading your project has Excel loaded on their machine.)

The system should be delivered with the three correctly-formatted .CSV files necessary for the three degrees that your system works with such that the administrator (or person grading the project, and thus installing the system following your written instructions) is only required to complete the computer steps.

To further clarify: The system does not need to “scrape” the catalog web pages.

Similar to the requirements for the grade-history data, the system should provide a facility for the administrator to confirm that data were read-in correctly from the .CSV file before updating the data in the system.

2.2.3 Read-In Reconciliation File

The system should provide a reusable means of reconciling (or connecting) data across the two data sets in the form of a file

Some reconciliation of data will be required, such as to associate “Introduction to Business” in the UO Catalog to “Intro to Business” in the grade-history data, and to collapse “BA101Z” and “BA101” both data sets into “BA101” given that they seem to be the same class.

These reconciliations should be recorded in a .CSV file entitled “entitled “Reconciliation.CSV” so that they can be reused in by all subsequent data loads, and should be applied to the grade-history data rather than the degree-requirements data (given that this is the less internally-consistent data set).

The .CSV file should list changes to the grade-history data and be formatted as follows:

CHANGE_FROM, TO
Intro to Business, Introduction to Business

It is expected that the same reconciliation file would continue to work when new grade-history files are acquired from the university. (Clearly this cannot be guaranteed, but this is the goal.)

The system only needs to deliver reconciliation files for the three degree programs (from the three different majors) delivered with the system.

2.2.4 Independent updating of degree-requirement and grade-history data.

The system should permit both sets of data (grade-history and degree-requirements) to be updated independently. For example, it should be possible to update the grade-history datas without updating the degree plan data, and vice versa.

Rationale: This permits the system to be updated as little as possible whenever new data become available.

2.3 Both Users

2.3.1 No Logins

No logins should be required for either the student or administrator to use the software, other than logging on to the computer being used, such as logging on to the laptop.

3. Non-Functional Requirements

3.1 Degree Plan Data Inclusion/Exclusion

The degree plan data files that are created based on the UO Catalog entry should:

- *include* specific required classes in other departments, such as the Math classes shown in Figure 1.
- *exclude* electives and special studies classes. The second class shown in Figure 1, for example, is “BA 199 Special Studies: [Topic]” and should be excluded.

3.2 Asterisk (*) Data

The grade-history file provided by UO includes a lot of asterisks (*s). These lines should be ignored *unless* asterisks appear in all of the grade data for a specific class that is required for a major; in this case, the report generated by the system should include a statement along the lines of “Class ART 101 is required for the major but no grade data are available.”

Not critical for the design or implementation of the system, but just for some background, the following information was provided along with the grade data and explains the asterisks:

04/02/2026

Dear Mr. Hornof:

Records responsive to your 4/2/26 request can be accessed via this link. Please note that access will expire in five business days on 4/9/26.

Please note that per the Registrar’s policy, grade distributions are subject to the following restrictions:

1. Classes must have at least 10 enrolled students
2. The distribution must be such that someone must find out more than 5 individual student’s grades before they can deduce someone else’s grade. An obvious example is when everyone gets the same grade, then if one knows a student in the class, then the student’s grade is apparent.

If the above are not met, or if the grade distribution causes a student’s grade to become “easily traceable”, then the distribution is redacted (all asterisks).

The office considers this document to be fully responsive to your request and will now close your matter.

Thank you for contacting our office with this request.

Sincerely,
Office of Public Records
6207 University of Oregon | Eugene, OR 97403-6207
(541) 346-6823 | pubrec@uoregon.edu
publicrecords.uoregon.edu

Again, not critical for building the system, but the public records request can be seen here:
<https://publicrecords.uoregon.edu/grade-distribution-4>

3.3 Same Data Reported for All Student Users

The system should report the same data for all student users. The target user is a student starting at UO who wants to pick a major to maximize their GPA throughout their degree program, or a currently-enrolled student who wants to pick instructors that give the highest grades. There is no need to customize the output for any particular student.

3.4 DNF Defined

DNF should include all non-passing grades, including D-, D, D+, F, and N. (DNF probably stands for “did not finish” but would more accurately be described as “did not pass”.)

3.5 Three Top Majors

The system should work for Computer Science plus any two of the other top majors listed at <https://www.uoregon.edu/academics> . (For maximum coverage across all groups, it would be preferred if groups randomly chose from the other fourteen majors.)

3.6 Graphs Generated by the System

The “All Instructors” graph should average all students who took a class, not averages of class averages.

Letter grades will include plusses and minuses of that grade (For example, A includes A+, A, and A-).

3.7 Grade Data

The grade data from 2015 through 2025 are available at:

https://canvas.uoregon.edu/files/25340409/download?download_frd=1

and from 2016 through 2025

https://canvas.uoregon.edu/files/25340410/download?download_frd=1

Your system should be able to load and work with both data files.

3.8 Do not Submit Grade Data When Submitting Your Projects on Canvas

When you submit your systems on Canvas for evaluation, do not resubmit grade data in any form, such as the .CSV files or a database file created from the .CSV files. Any database that your system uses to store the grade data should be created as part of the administrator initially starting up the system, reading from the save .CSV file(s) provided to students. Your system should thus provide instructions on how to load the .CSV file for use in the system, including to populate any database as needed.

4. Technical Requirements

This section presents a minimum set of technical requirements. Your projects should generate many more than just these.

1. The system should be standalone with all data on a local hard drive. Do not submit a URL in place of all of the code and instructions necessary to install the system. The system may not store or retrieve data from a database system that you created and is running on an external server, either your own server or a commercial service. (Rationale: Instructors need to execute the complete installation of your systems, including any server software and database system, and not just see that you have created an external website or database server that appears to be working.)
2. The system should not require the use of commercial server services. (Rationale: This would require instructors to also use such commercial services to evaluate projects, and this could violate (a) student privacy, especially if students are commenting your code correctly, or (b) user agreements, if students provide their login info for those services).
3. Installation of the system may use the internet to install docker.
4. No Login. There should be no login required for the student or administrator to use the software.
5. The delivered system should be complete and not require installation of software beyond a Python virtual environment (venv) or docker objects. Please do not expect the person installing the system (such as the instructor) to download any sort of programming environments such as XCode, or gaming engine such as Unity. Flask may be used.
6. The person installing the system should not have to create any accounts on any online services, and should not be asked to log on to any such accounts using your login info.
7. All code should be installed using a Python virtual environment (venv) or docker. A current version of Python and Docker Desktop will be pre-installed on the machine that will be used to evaluate your systems (a Mac). No software should need to be installed on the "native" machine used to test the code (such as in the "Applications" or "Library" folders of a Macintosh).
8. Installation instructions: If you are using Python venv, installation instructions should include the creation of the venv, and approved Python libraries should be installed in the venv. If you are using docker, installation instructions should probably include a "docker-compose.yml" file (ideally with comments explaining what it is doing).
9. All code and documents should be submitted on Canvas. (It is fine if your documents identify a github repository, but please submit all necessary code on Canvas.)
10. Systems should use standard libraries to the extent possible. For example, if written in Python, the systems should primarily use the Python Standard Library. Installation of dependencies should be fully documented and tested on Linux and MacOS.
11. Submit a report on libraries used. Justify why you needed to use each. Do not use additional libraries for functionality that can be achieved using a language's standard library. Explain why you used each library rather than the standard library or other libraries that provide

- similar functionality. Discuss the impact on project development if you were to not use the library. Provide a risk assessment of each library (long-term support, maintenance etc.).
12. The only database servers that may be used must be running locally. The program should not otherwise access a database or software running on a server or cloud-service that you have set up for this project.
 13. Instructions must be provided in the README.md to compile, run, and install all of the code necessary to use the system.
 14. Deviations from these requirements should be requested in writing. Cite the approvals in the final submission, such as "Approved in email from Anthony Hornof to Chi Zhu on 4-27-2026."

Build-Related Constraints

Target Platform. The software should run on a laptop or desktop machine.

System Document File Formats

All system-related and system-development-related documents that are intended for human reading must be in either plain text or PDF. For example, Microsoft Word, Microsoft Excel, or markdown language documents must be converted into plain text or PDF.

Programming Constraints

- Instructions must be provided for how to compile the code.
- No gaming engines such as Unity may be required to run your projects.

Installation

- There can be at most 20 user actions to compile the code, install the system, and run the program.
- An experienced computer programmer should not require more than 30 minutes working alone with the submitted materials to compile and run the code.

5. Acknowledgements

Hritvik Jekki Venkateshwarulu, the teaching assistant for CS422 in Spring 2026, assisted in the preparation of this project.