A Brief Report of the Assessment Inquiry Grant

Christopher Ryu, Professor of Computer Science (CS) Department

Project title: Upgrading the CS Assessment System

1. Brief description of the project and motivation

The CS department developed a web-based assessment system (http://assessment.ecs.fullerton.edu) before the ABET accreditation visit for the B.S. in computer science in 2020. The system has greatly helped the instructors measure the student learning performance in their classes and collect the assessment data for the related student learning outcomes. For the new instructors, the system also provides rich information about the purpose of assessment, assessment process, data collection process, data evaluation, and improvement in student learning.

However, the system lacks analytical features. Analyzing collected data requires significant manual effort, especially when the department wants to understand the overall student learning performance in each learning outcome and the specific areas for improvement. The system with powerful analytical features such as visual charts, statistical summaries, and automatically identifying the areas of poor/good learning performance will save the faculty time significantly and encourage them to focus more on improving student learning and curricular improvement.

2. The project goal and methods

The primary goal of this project was to automate the assessment workflow. To achieve this goal, I developed analytical features such as visual charts or graphs, statistical summaries, and automatic identification of the areas that show learning performance, integrated the features with the current system, and fixed some bugs identified so far. This development process required common software development activities such as requirements collection, analysis, specification, design and implementation of the requirements and testing, and deployment.

3. Outcomes of the project

The development was completed by August 31, 2023. The upgraded system is currently available. The flow diagram in Figure 1 shows the assessment workflow created by the department.

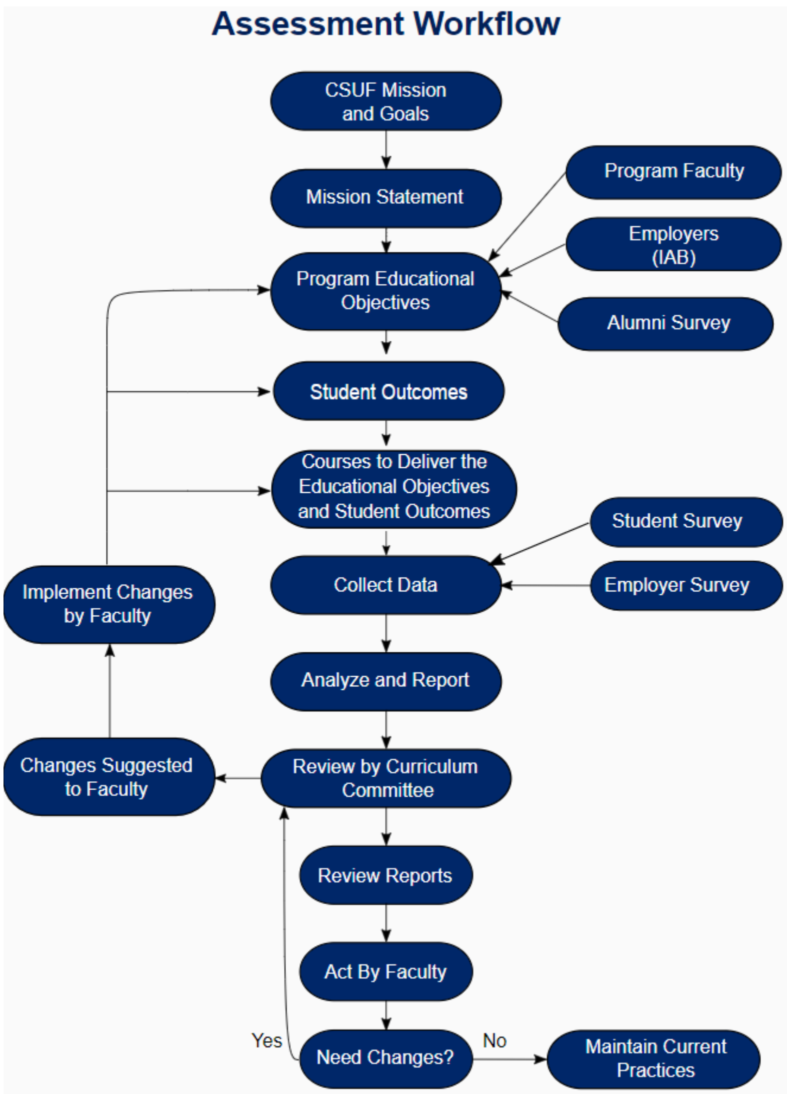


Figure 1: Assessment Workflow

The CS department used to manually collect and analyze the assessment data in the past. The current system was developed to automate the workflow of data collection, data analysis, and other necessary administrative work.

The assessment data are collected based on the program's educational objectives and student outcomes shown in Table 1, which the CS department defines. The program objectives are to assess the attainment of longer-term educational objectives in the future after students' graduation. To assess the program objectives, the department identified shorter-term objectives called "student outcomes" that state expected student learning goals upon graduation.

Programs Educational Objectives

The Computer Science programs have established the following educational objectives:

1. Technical Growth – Graduates will be successful in foundational and modern computing practices, integrate into the local and global workforce, promote growth and prosperity of the regional economy in the state and national level, and have passion for the profession and its growth.
2. Professional Skills – Graduates will continue to demonstrate the professional skills and communicative abilities necessary to be competent employees, assume leadership roles, and have career success and satisfaction.
3. Professional Attitude and Citizenship – Graduates will become productive members of society with high ethical and professional standards, who make sound technical or managerial decisions.

B.S Program Student Outcomes

Upon completion of the B.S degree program, graduates of the Computer Science program will have an ability to:

1. Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions.
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
3. Communicate effectively in a variety of professional contexts.
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
6. Apply computer science theory and software development fundamentals to produce computing-based solutions.

M.S Program Student Outcomes

Upon completion of the M.S degree program, graduates of the Computer Science program will have an ability to:

1. Demonstrate knowledge and competence in such fundamental areas of computer science as algorithms, design and analysis, computational theory, computer architecture, and software engineering.
2. Be able to analyze a problem, define the computing requirements appropriate to its solution, and apply design principles in the construction of software systems of varying complexity following systematic processes.
3. Be able to survey an area of interest, identify the key issues and problems of the selected area through review of academic literature, and provide potential solutions to the issues and problems.
4. Be able to function effectively on a team to accomplish a common goal.
5. Be able to communicate effectively with a range of audiences in both written and oral form.
6. Be able to understand and weigh possible social impacts of their work.

Table 1: Program objectives and student outcomes

Although both program objectives and student outcomes describe what the program want our students to learn from the program when they complete courses and degree, they are still general statements without specifying how and what to measure. Many instructors, especially adjunct faculty members, are unsure of what to do with them. Adjunct faculty members teach many CS classes. Therefore, helping them to assess student learning properly is essential. To provide instructors with specific guidelines, the department developed performance indicators, as shown in Table 2, that specify the areas of their assessment for learning performance and rubrics shown in Table 3 that guide them in assessment metrics.

Performance Indicators

The following performance indicators are used to measure student's learning performance:

ACODE: Write syntactically-correct and more advanced source code that make appropriate use of object-oriented concepts such as classes, encapsulation, and templates; and includes pointers, recursion, and memory management. Write source code with clear and informative comments following some coding standards or conventions.

ALG: Design an algorithm to solve a novel computational problem that builds upon classical techniques (e.g. data structures, discrete mathematics tools, divide-and-conquer, dynamic programming) and analyze the algorithm in terms of formalisms such as asymptotic efficiency, lower bounds, or computational complexity.

CODE: Write syntactically-correct source code, making appropriate use of fundamental constructs such as variables, branches, loops, and functions that solves a well-posed computational problem. Understand how computers process data, how to model domain concepts and procedures as data types and code, and how to formulate a human problem as an abstract computation.

COOP: Cooperate effectively on a group project.

DESC: Design software exhibiting design best practices, such as clarity, structured programming, separation of concerns, and/or design principles and patterns, and describe it clearly (using e.g. pseudocode, database schema, flowcharts, etc.)

ETH: Demonstrate an understanding of professional ethics appropriate to the use or development of computer science artifacts, and social impact of computer technology.

FB: Foundational Breadth: Demonstrate knowledge and competence in such fundamental areas of computer science as algorithms, design and analysis, computational theory, computer architecture, and software engineering.

FDBK: Demonstrate ability to make improvements after receiving constructive feedback.

HW: Demonstrate understanding of the architecture of computer hardware (i.e. CPU, memory, storage, etc.), low level programming (e.g. assembly), operating system, middleware, and computer communication protocols.

IPSEC: Demonstrate an understanding of intellectual property laws and ethics, software licenses, and commensurate rights. Demonstrate an understanding of security, privacy, and other ethical or legal issues, that arise in the context of computing.

PROC: Demonstrate knowledge of a formalized software engineering process (e.g. Agile, spiral, waterfall).

RESPEC: Translate an informal description of a problem into a precise requirements statement and develop specifications for a software system based on requirements.

SPEAK: Deliver a clear oral presentation which meets the needs of the intended listener(s).

SURV: Survey an area of interest, identify the key issues and problems of the selected area through review of academic literature, and provide potential solutions to the issues and problems.

TEST: Determine whether a program correctly meets its requirements, either through direct observation or the use of testing tools.

WRITE: Write a clear document which meets the needs of the intended reader(s).

Table 2: Learning performance indicators

The rubrics for performance indicators recommend that instructors measure student learning performance in three categories: Unsatisfactory, Developing, and Satisfactory.

Rubrics for Performance Indicators

Guideline:

The rubrics provide instructions on how to assess the performance indicators in classes. The instructors shall design the course work so that the data needed can be collected and evaluated. The following three categories are defined:

- **Unsatisfactory**: unable to achieve the basic skill and/or knowledge required by the performance indicator
- **Developing**: able to achieve the basic skill and/or knowledge required by the performance indicator
- **Satisfactory**: able to achieve advanced skill and/or knowledge required by the performance indicator.

Indicator \ Performance	Unsatiafactory	Developing	Satisfactory
ACODE	Unable to develop programming source code that make appropriate use of object-oriented concepts and follow acceptable coding standards or conventions.	Able to write programming source code that make use of some object-oriented concepts and follow some coding conventions.	Able to demonstrate the ability to write programming source code that make appropriate use of object-oriented concepts such as classes, encapsulation, and templates; includes pointers, recursion, and memory management; and follow industry coding standards or conventions.
ALG	Unable to design a process or algorithm	Able to design an semi- complete and/or semi- correct algorithm.	Able to design a correct algorithm to solve a novel computational problem.
CODE	Unable to write syntactically correct code. Making little to no use of fundamental constructs such as variables, branches, loops, and functions.	Able to write mostly syntactically-correct source code, which exhibits misunderstanding of fundamental constructs such as variables, branches, loops, and functions.	Able to write syntactically-correct source code, making appropriate use of fundamental constructs such as variables, branches, loops, and functions that solves a well-posed computational problem.
COOP	No evidence of collaboration, no plan, and/or no objectives.	Some evidence of collaboration yet not achieving team's objectives through deliberate planning.	Evidence of effective collaboration achieving team's objectives through deliberate planning.
DESC	Unable to describe a design clearly using pseudocode, database schema, flowcharts, etc.	Able to produce a design document but design is unclear and/or incomplete.	Able to describe a design clearly using pseudocode, database schema, flowcharts, etc.
ETH	Limited awareness of professional ethics and their practice.	Developing an understanding of professional ethics appropriate to computer scientists.	Demonstrate an understanding of professional ethics appropriate to computer scientists.
FB	Unable to demonstrate knowledge and competence to apply fundamental methods in algorithms, design and analysis, computational theory, computer architecture, and software engineering to solve a real life problem.	Able to use knowledge learned in such fundamental areas of computer science as algorithms, design and analysis, computational theory, computer architecture, and software engineering to formulate a solution to solve a real life problem, even the solution has flaws.	Able to use knowledge learned in such fundamental areas of computer science as algorithms, design and analysis, computational theory, computer architecture, and software engineering to effectively solve a real life problem.
FDBK	Unable to receive or react to constructive feedback.	Ability to receive constructive feedback and take action, which may not be appropriate.	Ability to receive constructive feedback and take appropriate corrective action.
HW	Unable to demonstrate basic understanding of the architecture of computer hardware (i.e. CPU, memory, storage, etc.), low level programming (Assembly), operating system, middleware, and computer communication protocols.	Able to demonstrate basic understanding of the architecture of computer hardware (i.e. CPU, memory, storage, etc.), low level programming (Assembly), operating system, middleware, and computer communication protocols.	Able to integrate the architecture of computer hardware (i.e. CPU, memory, storage, etc.), low level programming (Assembly), operating system, middleware, and computer communication protocols to achieve clear system level understanding.
IPSEC	Unable to demonstrate an understanding of copyrights, intellectual property, licenses, commensurate rights and responsibilities, issues of security, privacy, and/or trust as they relate to systems.	Able to demonstrate some understanding of copyrights, intellectual property, licenses, commensurate rights and responsibilities, issues of security, privacy, and/or trust as they relate to systems.	Demonstrate an understanding of copyrights, intellectual property, licenses, commensurate rights and responsibilities, issues of security, privacy, and/or trust as they relate to systems.
PROC	Unable to understand the concept of process and lack of knowledge of common software life cycle.	Able to demonstrate the basic knowledge on software process framework and common lifecycle, like waterfall and [illegible]	Have a clear understanding on software process and its relation to product quality, demonstrate the knowledge on [illegible]

Table 3: Rubrics for performance indicators

With more specific information, instructors assess student learning performance from their classes through various methods and activities such as projects, discussions, assignments, exams, surveys, or presentations, collect the data, and upload them to the system based on the schedule as specified in Table 4.

Data Collection Schedule

Please upload your assessed data by 05/29/2023.

- Use the rubric for each Performance Indicator to assess the student outcomes.
- More information about assessing student outcomes and some examples for data collection.

For any questions related to assessment process or data collection, please contact the assessment coordinator, Kevin Wortman.

Course (course coordinator)	Performance Indicator to measure in Spring	Performance Indicator to measure in Fall
CPSC 120 (Kevin Wortman)		CODE
CPSC 121 (Paul Salvador Inventado)	ACODE Enter the data	
CPSC 131 (Anand Panangadan)	ACODE ALG Enter the data	
CPSC 223C (Mikhail Gofman)	CODE TEST Enter the data	
CPSC 223J (Floyd Holliday)		CODE TEST
CPSC 223N (Floyd Holliday)		CODE TEST
CPSC 223P (Christopher Ryu)	CODE TEST Enter the data	
CPSC 223W (Paul Salvador Inventado)	CODE TEST Enter the data	
CPSC 240 (Floyd Holliday)		HW
CPSC 253/353 (Mikhail Gofman)		ETH IPSEC
CPSC 311 (Doina Bein)		FDBK WRITE SPEAK
CPSC 315 (Doina Bein)		ETH IPSEC WRITE SPEAK
CPSC 323 (James Choi)		DESC COOP FDBK
CPSC 332 (Shawn Wang)		RESPEC COOP FDBK
CPSC 335 (Kevin Wortman)	ALG DESC Enter the data	
CPSC 351 (Mikhail Gofman)	HW DESC Enter the data	
CPSC 362 (James Choi)	TEST RESPEC PROC Enter the data	
CPSC 471 (Yun Tian)		HW
CPSC 481 (Wenlin Han)		DESC FB
CPSC 490 (Christopher Ryu)	RESPEC PROC **Writing Requirements** Enter the data	
CPSC 491 (Christopher Ryu)		DESC COOP WRITE

Table 4: Data collection schedule

Administrative Features

In addition to the features of data collection and analysis, the system also provides the department and assessment committee with several convenient administrative features that allow easier maintenance of program objectives, student outcomes, and assigning student outcomes and performance indicators to courses and rubrics, as shown in Figure 2.

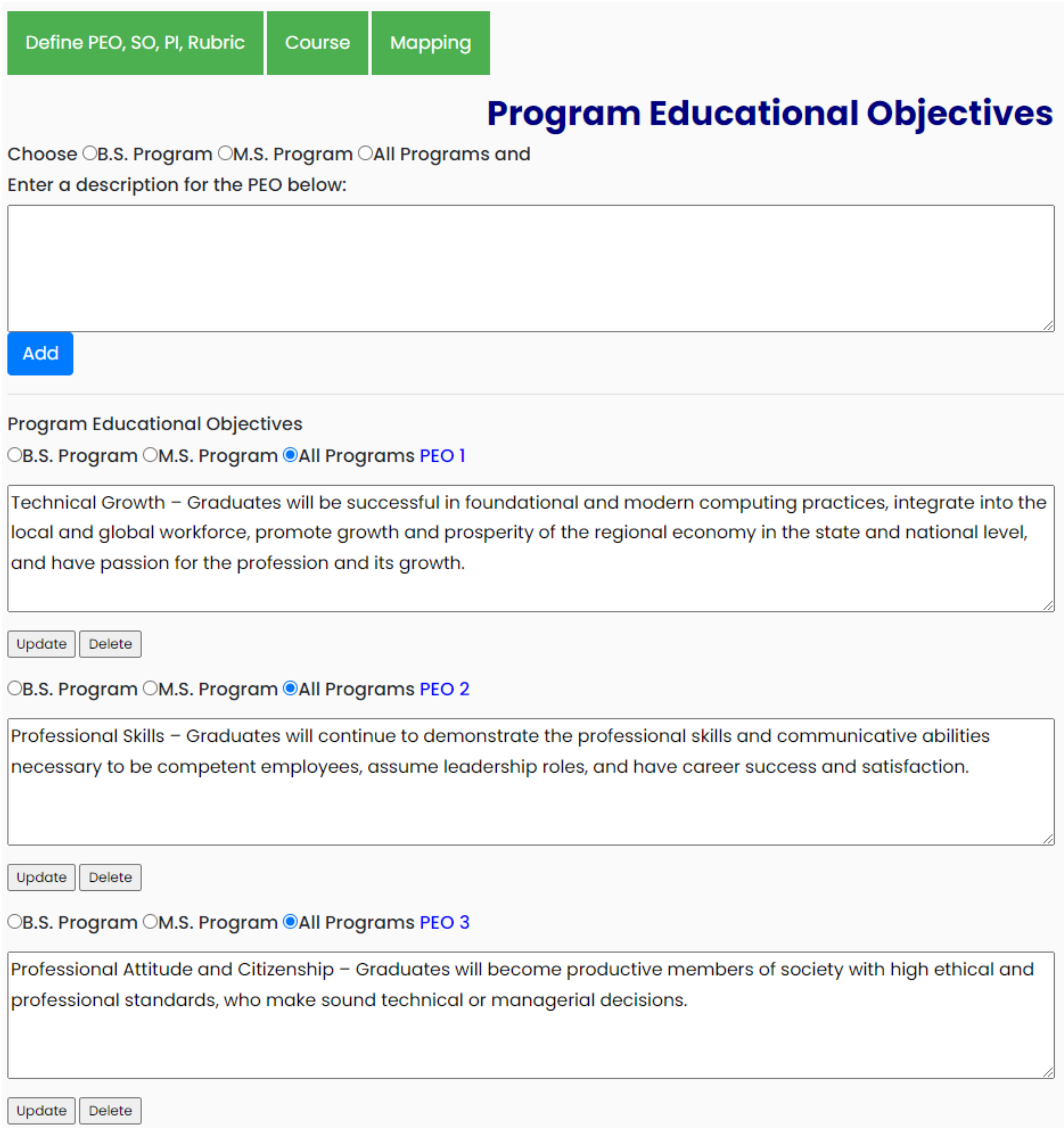


Figure 2: User interface for maintaining project objectives, student outcomes, and course mapping

Table 5 illustrates the mapping table for student outcomes, courses, and performance indicators. This table is useful for the assessment committee to plan for assessment and think about strategies to improve the student learning performance.

Student Outcome, Course, and Performance Indicator

Mapping Table for Courses, Student Outcomes, and Performance Indicators

Mouseover Course#, SO, or PI to view the description.

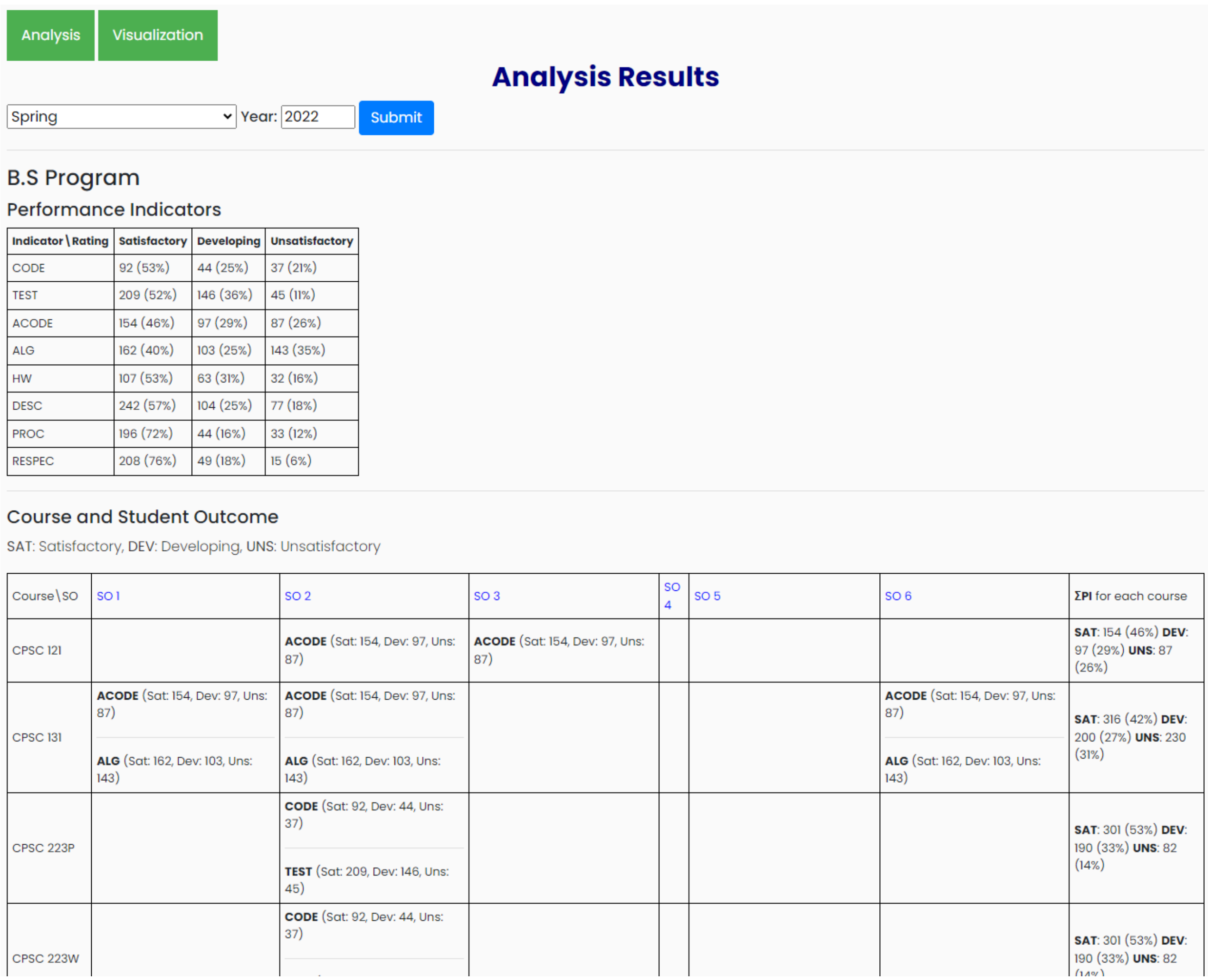
B.S Program

Course	SO.1	SO.2	SO.3	SO.4	SO.5	SO.6
CPSC 120		CODE				
CPSC 121		ACODE	ACODE			
CPSC 131	ACODE ALG	ACODE ALG				ACODE ALG
CPSC 223C		CODE TEST				
CPSC 223J		CODE TEST				
CPSC 223N		CODE TEST				
CPSC 223P		CODE TEST				
CPSC 223W		CODE TEST				
CPSC 240	HW		HW			HW
CPSC 253/353				ETH IPSEC		
CPSC 311			FDBK WRITE SPEAK		FDBK WRITE SPEAK	
CPSC 315			ETH IPSEC WRITE SPEAK	ETH IPSEC WRITE SPEAK		
CPSC 323		COOP FDBK DESC			COOP FDBK DESC	
CPSC 332	COOP FDBK RESPEC				COOP FDBK RESPEC	
CPSC 335		ALG DESC				ALG DESC
CPSC 351	HW DESC	HW DESC				
CPSC 362	PROC TEST RESPEC	PROC TEST RESPEC	PROC TEST RESPEC		PROC TEST RESPEC	
CPSC 471	HW					HW
CPSC 481						FB DESC
CPSC 490			PROC RESPEC		PROC RESPEC	
CPSC 491			COOP WRITE DESC		COOP WRITE DESC	

Table 5: Mapping table

Analysis Features

The collected data will be automatically analyzed by computing the basic statistics showing the student learning performance by student outcome and performance indicator.



Analysis | Visualization

Analysis Results

Spring | Year: 2022 | Submit

B.S Program

Performance Indicators

Indicator \ Rating	Satisfactory	Developing	Unsatisfactory
CODE	92 (53%)	44 (25%)	37 (21%)
TEST	209 (52%)	146 (36%)	45 (11%)
ACODE	154 (46%)	97 (29%)	87 (26%)
ALG	162 (40%)	103 (25%)	143 (35%)
HW	107 (53%)	63 (31%)	32 (16%)
DESC	242 (57%)	104 (25%)	77 (18%)
PROC	196 (72%)	44 (16%)	33 (12%)
RESPEC	208 (76%)	49 (18%)	15 (6%)

Course and Student Outcome

SAT: Satisfactory, DEV: Developing, UNS: Unsatisfactory

Course \ SO	SO 1	SO 2	SO 3	SO 4	SO 5	SO 6	ΣPI for each course
CPSC 121		**ACODE** (Sat: 154, Dev: 97, Uns: 87)	**ACODE** (Sat: 154, Dev: 97, Uns: 87)				**SAT**: 154 (46%) **DEV**: 97 (29%) **UNS**: 87 (26%)
CPSC 131	**ACODE** (Sat: 154, Dev: 97, Uns: 87) **ALG** (Sat: 162, Dev: 103, Uns: 143)	**ACODE** (Sat: 154, Dev: 97, Uns: 87) **ALG** (Sat: 162, Dev: 103, Uns: 143)				**ACODE** (Sat: 154, Dev: 97, Uns: 87) **ALG** (Sat: 162, Dev: 103, Uns: 143)	**SAT**: 316 (42%) **DEV**: 200 (27%) **UNS**: 230 (31%)
CPSC 223P		**CODE** (Sat: 92, Dev: 44, Uns: 37) **TEST** (Sat: 209, Dev: 146, Uns: 45)					**SAT**: 301 (53%) **DEV**: 190 (33%) **UNS**: 82 (14%)
CPSC 223W		**CODE** (Sat: 92, Dev: 44, Uns: 37)					**SAT**: 301 (53%) **DEV**: 190 (33%) **UNS**: 82 (14%)

Table 6: Analysis results in terms of statistics

The analysis results are also visualized for the committee to quickly understand the student learning and find out the areas of improvement.

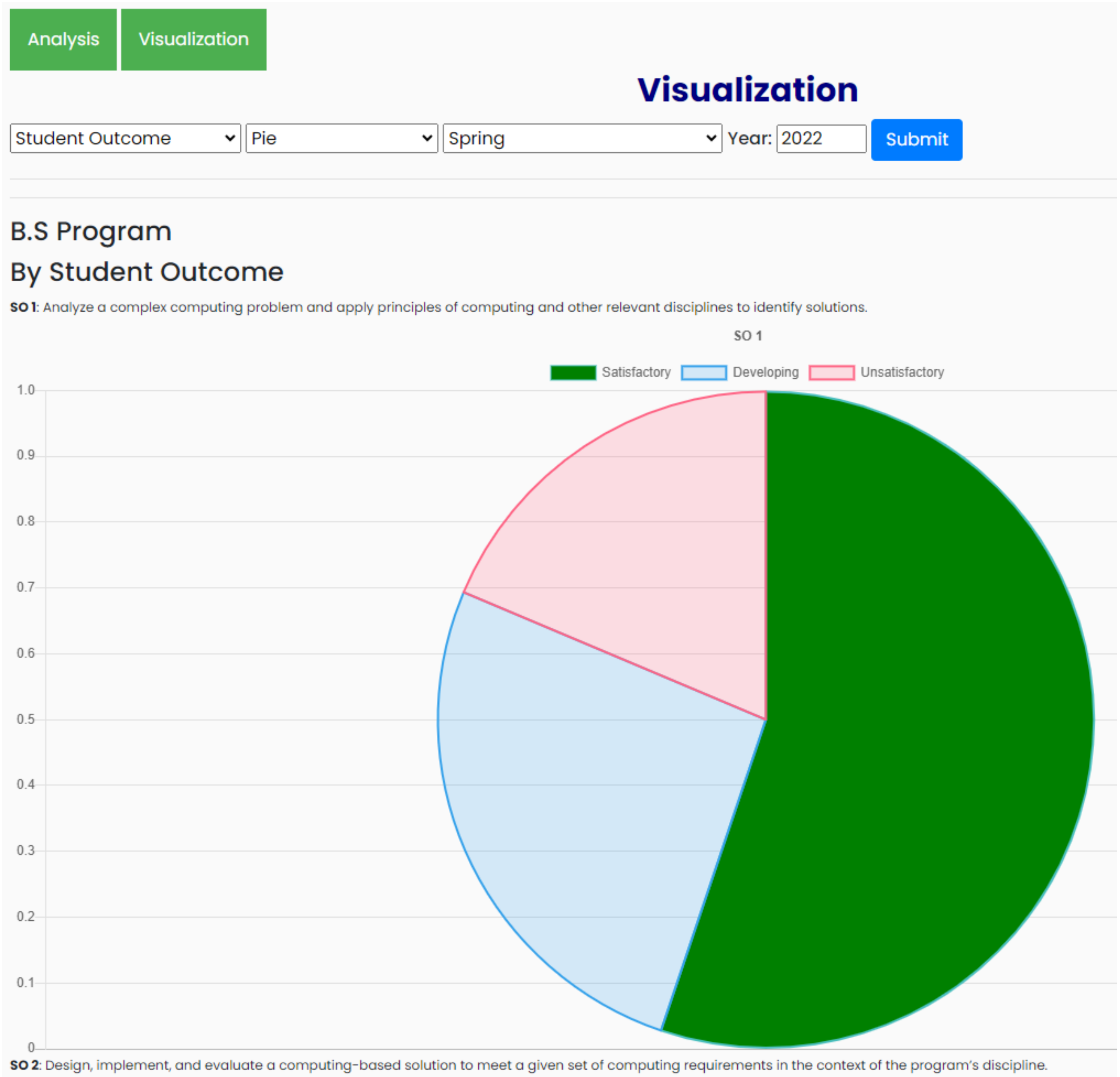


Table 7: Visual analysis results

4. Conclusion

With the upgraded system with analytical features, the CS department will understand student learning performance better, identify the areas for improvement, develop strategies to improve student learning, provide the students with the necessary support, and change the curriculum if necessary. The system can ultimately enable continuous improvement with the least amount of valuable faculty time by eliminating unnecessary manual work. Moreover, the system removes the burden of the significant workload from faculty and administrative staff, especially related to labor-intensive work, so that they can focus more on improving teaching effectiveness and student learning.

The department will benefit from the system by helping faculty focus more on their instruction and student support than unnecessary labor-intensive work. The college will benefit from the system as a similar system can be provided to other departments if necessary, as they regularly go through the same ABET accreditation process. The university will benefit from this project by sharing the lessons learned from this automated assessment system and the possibility of developing similar systems for other colleges to automate their assessment process.

Future work

Currently, a few faculty members are developing more sophisticated strategies, such as machine learning algorithms, to discover potential hidden areas of improvement in student learning, curriculum, and teaching effectiveness. The database created and maintained by the current system will be handy for such advanced analysis.