

Computer Science (CS) Placement Exam

COP 4934 Group 8 Spring 2020

04/21/2020

Team Members:

Alexis Alonso

Mark Fuller (Project Lead)

Lukas Getter

Danny Parsons

Dunquan Zheng

Sponsors:

Dr. Mark Heinrich

Dr. Sumanta Pattanaik

Prof. Arup Guha

Melissa Dagley

Denise Tjon Ket Tjong

1	Executive Summary	1
2	Project Overview	2
2.1	Statements of Motivation	2
2.1.1	Alexis Alonso	2
2.1.2	Mark Fuller	2
2.1.3	Lukas Getter	3
2.1.4	Danny Parsons	3
2.1.5	Dunquan Zheng	4
2.2	Broader Impacts	4
2.3	Legal, Ethical, and Privacy Issues	5
2.4	Project Budget	6
3	Computer Science (CS) Placement Exam	6
4	Exam Content	8
4.1	Initial Goals	8
4.2	Testing Focus Areas	9
4.3	Automatic Item Generation	10
4.3.1	Introduction	10
4.3.2	Development	10
4.3.3	Benefits	11
4.3.4	Considerations	11
4.3.5	Future Developments	12
4.3.6	Alternatives and Consequences for Isomorphs	13
4.4	Question to Exam Variations	14
5	Sample Example Asset	16
5.1	Pseudocode Notation	16
5.1.1	Prior to question 1:	18
5.1.2	Prior to question 11:	18
5.1.3	Prior to question 16:	19
5.1.4	Prior to question 18:	19
5.1.5	Prior to question 19:	19
5.1.6	Prior to question 20:	20
5.2	Detailed Sample Exam Analysis	20
5.2.1	Question 1	20

5.2.2	Question 2.....	21
5.2.3	Question 3.....	23
5.2.4	Question 4.....	25
5.2.5	Question 5.....	27
5.2.6	Question 6.....	28
5.2.7	Question 7.....	30
5.2.8	Question 8.....	31
5.2.9	Question 9.....	32
5.2.10	Question 10	33
5.2.11	Question 11	35
5.2.12	Question 12	36
5.2.13	Question 13	38
5.2.14	Question 14	39
5.2.15	Question 15	41
5.2.16	Question 16	42
5.2.17	Question 17	44
5.2.18	Question 18	45
5.2.19	Question 19	47
5.2.20	Question 20	48
5.3	Testing Concepts On Sample Exam Summary	50
6	Replacement Exam Content	53
6.1	Revised general pseudocode notation instructions:	53
6.1.1	Instruction language	58
6.2	Question Design	60
6.3	Testing Concepts Not on Sample Exam	61
6.4	New questions:.....	64
6.4.1	Function question that uses void and strings	64
6.4.2	Revised question 13	66
6.4.3	Operator precedence challenging.....	67
6.4.4	Basic recursion	68
6.4.5	Basic recursion	69
7	Project Milestones.....	71
7.1	Shift in Goals.....	71

8	Specifications and Requirements	72
8.1	Hardware Summary.....	72
8.2	Software Summary	73
8.3	Website Design Summary	74
8.4	Systems Design Summary.....	75
9	Design Considerations	75
9.1	JavaScript Framework	75
9.2	Single Page vs. Multi Page	76
9.3	Browser Compatibility.....	77
9.4	Linux, Apache, MySQL, PHP (LAMP) Stack.....	77
9.5	Administrator Portal.....	78
9.6	Instruction Presentation.....	79
9.7	Statistical Analysis	80
9.8	Question Resources.....	81
10	Diagrams	82
10.1	Block Diagram.....	82
10.2	Use Case Diagrams	84
10.2.1	Registered Student	85
10.2.2	Administrator.....	86
11	Storyboards and Wireframes.....	87
11.1	User Storyboards.....	87
11.1.1	Student View	87
11.2	User Wireframes	88
11.2.1	Nonregistered Student	89
11.2.2	Registered Student Who Has Taken the Exam	90
11.2.3	Registered Student Who Will Take the Exam	90
11.3	Administrator Portal.....	93
12	Backend Development.....	98
12.1	Single Sign-On.....	98
12.1.1	Apache .htaccess File.....	99
12.2	PHP Development Environment.....	101
12.3	Database Environment.....	102
12.3.1	Schema Design.....	103

12.4	RESTful API	110
12.4.1	RESTful API Organization	110
12.4.2	RESTful API Endpoints.....	111
12.4.3	Database.php.....	114
12.4.4	Read.php (Basic CRUD template)	116
12.4.5	ReadByKey.php (Basic CRUD template).....	117
12.4.6	Delete.php (Basic CRUD template)	119
12.4.7	Insert.php (Basic CRUD template)	120
12.4.8	Update.php (Basic CRUD template).....	122
12.4.9	Getscore.php	123
12.5	RESTful API Testing.....	124
13	Steps Going Forward	125
14	Project Summary	126
15	References	128

1 Executive Summary

The Computer Science (CS) Placement Exam project is sponsored by University of Central Florida (UCF) faculty members and staff. The initial goal of the project was to generate exam content for a computer science placement exam that will assist in deciding what courses a student in the incoming Summer 2020 semester should initially enroll in to pursue a computer science degree. The second goal of our project is to design a website that can administer subsequent exams and also serve as a faculty portal to manipulate exam content and examine exam metrics.

All candidate students for the CS Placement Exam must already be admitted to the University of Central Florida and possess a valid UCF network identification (NID). The CS Placement Exam website will allow students to take the exam using their existing browsers without the need to install additional software. The exam will be a timed exam. After completing the exam, the results of the exam will be immediately displayed to the student with the option to email the results to an email address of their choosing. Upon completion of the exam, the results determine which initial computer science course a student can enroll in. Course override decisions will be determined based on the results of the CS Placement Exam. If a passing score is not achieved, then students must enroll in *Introduction to Computing* (COP-2930). This course does not count towards the required courses for the Computer Science (B.S.) degree. Students that pass the CS Placement Exam can enroll directly in the course *Introduction to Programming with C* (COP-3223C).

The first CS Placement Exam content is derived from our careful analysis of the concepts tested, the language used, and the structure of the content of an example exam asset. The exam content must be able to test a general sense of programming skills in order to gauge students' readiness for the material covered in *Introduction to Programming with C*. The exam content must be able to be formatted for Webcourses in such a way that it is unlikely that two students will take the exact same exam. The exam content must be able to be used for the students enrolling in classes in the Summer 2020 semester.

The CS Placement Exam website will be required to interface with other University of Central Florida computer systems. It must be hosted on the Computer Science department server using existing hardware and technology stacks. An inbound interface in the form of a nightly batch list of student network identifications that are required to take the exam will be provided by the University of Central Florida enrollment system. A faculty-only interface will allow management and statistical information for all exams, questions, students, and results.

2 Project Overview

2.1 Statements of Motivation

2.1.1 Alexis Alonso

I am highly self-motivated when it comes to daily tasks, but in a working world I am powered by value and community. I chose this project for a few different reasons, one of which is the value that it will bring to our computer science department and future student community. I came to the University of Central Florida without any prior coding or computer science experience, and I remember feeling overwhelmed walking into *Introduction to Programming with C*. If this placement test were implemented four years ago, I definitely would have been one of the students placed into the Python course. For that reason, this project feels very relatable in that I know the exact kind of student it will benefit. I was able to learn quickly through my coursework with the help of online videos, professor's office hours, and class teaching assistants. I have made it this far, but my transition into this field or topic could have gone smoother if I took a baby step into our program. As I am planning on graduating this August, the quick project timeline was also an aspect that drew me in. I like efficiency and getting things done quickly and correctly. I cannot wait to graduate in August and be proud of a final senior design project that will give back to the department and school that have gotten me this far.

2.1.2 Mark Fuller

I have been employed over 35 years now and most of that time has been centered around education in some form or another. I have worked as a private tutor, an instructor in a very large multinational computer technology corporation, as a graduate teacher assistant, and even as a college lecturer. I have also been a very involved parent and grandparent regarding educational goals. One of my core values is that education is the cornerstone of the American dream. I became infatuated with computers at a very young age even though the age of computers was in its infancy, and that passion has led me to fulfil my dreams regarding computers. I hope that by completing this senior design project, I can take a small part in helping to fulfil the dreams of those students that also have a passion for computers and computing in general. I want them to succeed to their highest expectations just as I have in my computing career. It is never too late, and you are never too old to embrace

education. I do not feel as though I choose this project, but rather this senior design project chose me.

2.1.3 Lukas Getter

With this project comes the opportunity to learn new technologies and to give back to my school by providing a system that will help future students. By undertaking this project, I have put myself in a situation where I will need to learn many new tools to be successful. I look forward to learning throughout this experience and to then apply those skills towards a practical solution. Here at the University of Central Florida I have made a number of friends that wanted to major in Computer Science but struggled to pass the required entry level courses. To me, it is unfortunate to see so many people give up on their original goals because it seemed too difficult. By being part of the development of this placement test, I hope to ensure that future students are confident in their ability to take on the challenges this major provides.

2.1.4 Danny Parsons

Learning science has been a passion of mine since high school. From tutoring at the high school level to substitute teaching at the elementary school level, I have taken great care to be open to students showing me new ways of engaging with educational materials that lead to success. Different learning styles each provide a unique insight into the way that people think. The exact manifestation of someone's learning style can often be difficult to judge, especially without context, and having worked with students of all walks of life, I have been gifted a greater context. Even within my own academic career, I have had experiences that help me empathize with all sorts of students who might be starting their Computer Science educational career: I have been a transfer student, a graduate student of three different varieties, and, now, a second bachelor-degree seeking student. This project affords me the opportunity to apply what I have learned from my students and my own experiences while still feeling both technically challenging and feasible for my programming level. I now have a year's worth of professional software development experience, and most of what I have learned over the last year is that there is so much more to learn in order to make good products that stand up to the test of time. Additionally, accessibility of educational material can often be overlooked or delayed until later stages of development when it is more difficult to apply. I currently have a decent if basic understanding of accessible design and a deep passion for learning more, but this project will allow me to apply what I know and give me the opportunity to learn more about both accessibility of UI design and to the actual material of the test. If all incoming CS students

are going to take this test from summer to infinity, then I want a hand in making the test as accessible to as many students as possible from its earliest iterations.

2.1.5 Dunquan Zheng

I am highly motivated about doing this project. Since high school, when I first touch computer science, I am very interested in the software, application and game development area of computer science. Last semester, I learned a few things about web development, it is also very interesting to me, and I would like to learn more about it. Other than that, I think that doing a formal project for the University of Central Florida would be very helpful for my career later in life. I also think this project is very meaningful for my college life. Since I have some programming experience in high school, I did not experience the drastic learning curve, but a few of my friends and *Introduction to Programming in C* classmates felt difficulties when they headed to their first programming class. It is the first step that caused trouble. A gentle introduction to the challenging *Introduction to Programming in C* class would have been very helpful for them. A placement test would be a good way for student to go straight into the *Introduction to Programming in C* course. Therefore, I am very happy about creating something useful for future students and the University of Central Florida before I graduate.

2.2 Broader Impacts

There are many students that enroll in *Introduction to Programming with C* (COP-3223C) at the University of Central Florida who have never been exposed to programming or computer science. Prospective students come from very diverse socioeconomic groups, and the reality is that some students grow up in poorer countries, poorer neighborhoods and attend less affluent schools. They may have never owned a personal computer or had the opportunity to take computer programming courses in high school with certified professionals. While many of these students perform exceptionally well in their first programming course at the University of Central Florida, there are students that are not ready for the rigorous academic endeavors of computer programming, let alone programming in a language that requires direct interaction with computer memory. This can lead to students changing their intended majors and to students doubting their abilities to perform well or graduate with a computer science degree. It can lead to shattered dreams and a loss in self-esteem. Poor experiences with early classes in degree programs encourage students to discourage others from taking similar courses or from working hard to succeed when they are sure that they will fail.

The most prominent broader impact of the CS Placement Exam is the identification of students that would benefit from additional academic course offerings which will afford students a slower-paced introduction to the world of programming with a higher level of abstraction. Catching these students before they enroll should result in a greater degree of success, a positive outlook on their choice of a computer science plan of study, and a reduced failure rate of students taking *Introduction to Programming with C* (COP-3223C). Ultimately, it is about assisting everyone who has a dream to excel in computer programming, to enhance their abilities with the hopes that those students graduate, and to give students the chance to give back meaningfully to the system that they took part in.

2.3 Legal, Ethical, and Privacy Issues

Because this senior design project is not working with an external customer, there is no specific non-disclosure agreement or intellectual property rights issue specific to this project. Nevertheless, there are legal, ethical, and privacy issues that must be addressed. In 1974, President Gerald Ford signed into law the Family Educational Rights and Privacy Act (FERPA). This law governs the privacy and access to educational information records by publicly funded educational institutions and other public entities. It gives students the right to limit disclosure of educational records to other parties and ensure privacy. Information collected on students is divided into three categories: directory information, personally identifiable information, and educational records. Directory information includes items such as name, current mailing address, telephone number, date of birth, major, dates of attendance, and enrollment status. This information may be disclosed unless a student should request otherwise. Personally identifiable information includes items such as social security number, the University of Central Florida student ID (UCFID), residency status, gender, religious preference, race or ethnicity, and email address. Educational records include information such as grades, grade point average, class schedule, test scores, academic standing, and academic transcripts. Both personally identifiable information and education records categories may not be released to anyone but the student, and after that student has verified their identity with proper identification. However, University of Central Florida officials with a legitimate education interest may access non-directory information without a student's written consent. Our software design project must comply with FERPA laws. The CS Placement exam system will receive a list of UCF network identification numbers (NID) without any additional personally identifiable information directly from University of Central Florida's internal systems in a nightly batch exchange. Students will enter their NID to verify their eligibility to take the exam. If the students have been flagged as needing to take the exam based on their NID, they will then be redirected to an internal University of Central Florida system to complete single sign-on much like

Webcourses and myUCF web portals. The CS Placement exam system will not store or have access to student passwords. When the CS Placement exam result is computed, the pass-fail indicator will be displayed on the screen. A feature request is to allow the result to be emailed to the student. If this feature is implemented, the results web page can prompt the student to enter the email address to send the result to. Checks can be implemented to ensure that only official University of Central Florida email addresses are valid destinations, and the email can be generated without permanently storing the student's email address.

In addition to FERPA legal issues, the CS Placement Exam project will utilize existing hardware and software resources acquired by the University of Central Florida. It is assumed that the internal hardware systems and technology stacks utilized by the University of Central Florida are maintained in accordance with any licensing terms. This project will only use open-source developer resources that are already included on the University of Central Florida systems. It is assumed that all supplied software used by existing systems is in legal compliance. The ethical standards of our team members will ensure no violations of software usage, copyright infringement, or plagiarism occur during this project.

2.4 Project Budget

The budget goal for this project is \$0.00. It is expected that the existing server hardware used by the computer science department will be used without the need to increase CPU, memory, or disk resources. It is expected that the open-source technology stack currently deployed on the server will be used without any additional software licensing cost. Team members are expected to use their own computer hardware for developer resources or utilize the computer systems provided to University of Central Florida students for development. All development tools such as compilers and debuggers should be open-source or software tools provided by the University of Central Florida. All project management tools such as tools for graphics, drawing, documentation, and illustrations should be open-source tools or tools supplied by the University of Central Florida.

3 Computer Science (CS) Placement Exam

The exam must be taken online from remote locations using only a browser as the client platform. There are no plans for this exam to be proctored or forced to run in a restrictive sandboxed or locked-down browser environment. This reduces work required by the

faculty during the test-taking process itself. Any issues of cheating due to the unrestricted nature of the interface are determined by the sponsors to be unimportant considerations due to the nature of the outcomes of cheating: a student being placed in a class they may not be ready and thus they might be at a higher risk of failing than other students. There are no plans to allow practice exams and retakes will be granted only for system related issues encountered during the exam. Practice exams and retakes would make it harder to discern what a student is ready for in a classroom setting. Each retake would often result in a higher score, and the exam is not designed to be used as a learning tool. A small amount of reference material on pseudocode will be provided to standardize initial engagement with the materials, but this reference material will strictly be related only to the syntax necessary to approach questions and not any material that is being tested within itself.

The exam will be timed and the exact time limit for taking the test will be determined pending a final decision on the total number of questions used on a specific exam and the calibration of the exam with various student groups: middle and high school students with no programming experience or who have taken an initial course in Python, C, or Java; students who have taken the existing *Introduction to Computing* Python course; and students before and after their *Introduction to Programming with C* course, including those who had taken *Introduction to Computing* as well as those who had not. These different sets of students will serve as a wide variety of baselines to determine if the test that is designed is valid in that it is testing what we want it to test and that the difficulty level is appropriately set. If a student has demonstrated abilities in understanding programming, their scores can be used as potential failure points. For students who take the test before and after their *Introduction to Programming with C* course, then the pre- and post-test results can be compared directly to see if there is a correlation between the kinds of materials that a student comes out of the *Introduction to Computing* course understanding and is ready to apply to the C programming language. The ability to parameterize certain questions such as changing the integers or numbers used in a formula and thereby changing the answer should be considered. All questions are weighted equally for grading purposes. The following types of questions can be commonly found on exams:

- Multiple choice
- Matching
- Short answer
- True / False
- Fill in the blank
- Essay
- Performance or task-based
- Computational

- Adaptive

It was decided in the initial sponsor kickoff meeting that all questions should be of type multiple choice and that true/false type questions should be greatly reduced from the number that exist in the sample bank or eliminated entirely. Therefore, the exam will be multiple choice with a maximum number of answer options that will be the same across all questions. The exam grade should be available immediately at the completion of taking the exam and the results displayed. In addition, the result of pass or failure will be marked on the student's record. The passing score has not been determined at this time but will be also be calibrated with the same groups of students mentioned above. Students should be able to email the results to an email address supplied by the student. This email address is either not stored in the system, or it must follow FERPA laws and regulations if stored.

4 Exam Content

4.1 Initial Goals

Our project began with two different goals. The first goal was to establish a short-term viable solution for the next incoming class of students to be used as the first trial of the exam system. The test would need to be finished by April and be deliverable in some manner that would allow students to take the test and have their scores recorded. When discussing this original goal, the content of the exam questions itself became our first and most important deliverable. We were given an example asset of twenty questions that we were tasked with turning that twenty-question set into a workable first exam for students to take.

We were able to focus on the content of the exam exclusively because it was decided that the best way to allow for full system integration as early as possible would be to use Webcourses, which is already in place for the Computer Science department to use to keep track of students' enrollment and scores. Generating a single exam on Webcourses is fairly standard.

With a simple JavaScript file, with a given set of questions acting as templates, the output can be input into Webcourses with the Prospectus plugin. Prospectus is already included for instructors at University of Central Florida, and the instructor can then manually go through and choose which question is mapped to a quiz or test question on the Webcourses system. So, for instance, Dr. Pattanaik was able to generate a question set that had limited randomness autogenerated and could then pick which question was going to be the second question that appeared in the Webcourses test.

These first demonstrations of this feature reduced our initial work to working solely with the content, as the other aspects of our first deliverable were mostly taken care of for us. This left us exclusively with the aforementioned twenty-question example set of questions to analyze and rework the existing set into something usable for April.

4.2 Testing Focus Areas

The initial sponsor kickoff meeting indicated that the exam should cover four specific areas of concentration: general or sequential program flow, conditionals, loops, and functions. These kinds of questions can be broken down further into subcategories, but to derive the intended interpretations of what these categories mean, we looked to the sample exam.

The twenty-question sample exam provided to this team is mapped to each of the four testing categories in Table 1.

Table 1: Sample exam questions mapped to the categories identified as the desirable broad subjects areas to cover in the exam

Focus Area	Sample Exam Questions
General and sequential	Q1, Q2, Q3, Q4, Q5
Conditionals	Q6, Q7, Q10
Loops	Q11, Q12, Q13, Q14, Q15, Q16, Q17, Q18
Functions	Q19, Q20

Question 8 and question 9 are not listed in Table 1 since they appear to be testing only mathematical concepts and not programming specific concepts. It should be noted that the University of Central Florida *Introduction to Programming with C* course has no pre-requisites of any type. Students that do not score high enough on the Math Placement Test (MPT) must enroll in a bridge mathematics course such as *MAT 1033 Intermediate Algebra*. This course does not count toward degree requirements and does not satisfy the General Education Program (GEP) or Gordon rule. However, students enrolled in this bridge course are eligible to enroll in the *COP 3223C Introduction to Programming with C* course since there are no pre-requisites. Unless pre-requisites change, there must be a balance between programming skills and math skills that students possess at the introductory level.

Additionally, if there is a strong mathematics portion of the exam, if a student has a strong math background and is able to pass the test on math alone, then the exam is not valid as

an instrument that evaluates readiness for computer science and programming and would need reevaluation and editing.

This led us into an immediate design consideration that had to be addressed: when generating a set of questions to be used to input into the exam, would it be better to use automatic item generation to continually generate new versions of the same question or would it be better to focus on building a pool of questions to select from?

4.3 Automatic Item Generation

4.3.1 Introduction

Automatic Item Generation (AIG) is a test development strategy that involves computers handling test questions with question stems or models and a series of rules or algorithms. Another name for Automatic Item Generation is rule-based item construction. A well-developed Automatic Item Generation algorithm should be able to generate items in such a way that human involvement is minimal in determining the acceptability of questions and in the creation of new questions that “share the same psychometric properties, like measuring the same construct and having the same specific values of item parameters in an appropriate modeling framework” [1, 2].

4.3.2 Development

In *The Foundations of Item Generation for Mass Testing*, Irvine broke the abstract idea of item models into two parts: radicals and incidentals [3, 4]. Radicals are the fundamental properties of the question which determine concept and difficulty and are relatively synonymous with question stem. An example for a computer science question would be determining if a loop is finite or infinite. The incidentals are the individual parameters that can be manipulated to produce many “question-clones” or isomorphs. In that example, some incidentals could be whether counters are utilized correctly, counters are manipulated correctly, and the type of loop. A question that uses the same radical but has a counter incrementing to a certain value is isomorphic to a question that has a counter decrementing to a value.

In our case, the initial twenty questions given to the team would serve as a first draft of the radicals for the placement exam. Because the exam has no documentation explaining a cognitive model that helped guide writing the exam, however, the questions need to be broken down further until our team can generate a cognitive model of our own to extract

concepts and write radicals. See below for our discussion of the individual questions. The development of generating programs or methods/functions inside a single program can only be begun when there is a fundamental understanding of what the test should be testing at its core. After the radicals have been identified and written, the logic behind generating the incidentals becomes trivial.

4.3.3 Benefits

The greatest benefit of creating models for questions and writing programs that implement Automatic Item Generation is the ability to generate variations of questions to increase the total question pool to pull from and, if it is a concern, reduce security risks of cheating. The more advanced the model, the greater the difference between each generated item, which increases the impact of these benefits. Additionally, because of the structured approach authors of the question models must take, psychometric analysis is able to be conducted more finely [5].

Besides being an easy way to refresh the item pool if it should ever be determined to be stale, the Automatic Item Generation would make psychometrics per question much easier. After a student completes an exam, each question stem would have metrics for the individual student's time to complete and the pass/failure status on that question, on its own and in comparison to the overall pass/failure status of the test. When aggregated for entire semesters, question isomorphs can be evaluated against each other to determine if there are any issues in difficulty that persist across different variations despite the logic of the Automatic Item Generation algorithm, which can be calculated both by the time to complete and the pass/failure rate. Later student success can be mapped onto individual question pass/failure rates as well.

4.3.4 Considerations

The cost of development of Automatic Item Generation to reap these benefits vs. the cost of hiring experts in the domain area to manually generate item questions is discussed by Kosh et al., and their analysis indicates that in order to make the financial cost worth the development of the cognitive model necessary for "strong-theory AIG," for each very small concept being tested, it would take roughly 200 isomorphs for just a small concept to be the tipping point to make the Automatic Item Generation be worth the upfront cost. Until that question item count threshold is met, it is cheaper to manually develop each question [6].

One current model of our design is such that our Automatic Item Generator could be run just once and then never run again. This could be useful in testing the future viability of additions to the question pool, but with the limited scope of a short test it is especially more costly to develop Automatic Item Generation than to manually write questions. The time cost on the development of scripts would be a trade off against developing better, more robust question stems or finding groups to calibrate the existing pool of items, or the development of ways of making crowdsourcing additional radicals easier.

With a test that is very unlikely to have more than a total of 50 questions, at least initially, that span across many fundamental concepts of different aspects of computer science and programming, the financial cost of developing Automatic Item Generation for this placement exam would far exceed the cost of hiring domain experts to manually write the handful of questions. The financial cost does not necessarily apply to our particular endeavor given that we are tasked with creating the most useful system and are neither in a position to pay others nor being paid ourselves, but it is a consideration for future faculty or Senior Design teams after our initial development.

4.3.5 Future Developments

A paper directed to the team specifically was *On the Use of Semantic-Based AIG to Automatically Generate Programming Exercises* by Zavalo et al. [7]. In the paper, the authors used two sections of an introductory Python programming course and, for one section, gave their Automatic Item Generated homework questions with semantic groupings as the majority of their incidentals as an intervention while the other received standard homework assignments. The authors found that in a post-test of the students' understanding of the materials, they saw more even distribution across all tested area of working with code: code reading, code manipulation, and code writing. Without the intervention, the researchers saw that those students tended to have uneven skills: weakest growth of skill in code writing, stronger in code manipulation, and stronger still in code reading.

The most novel inclusion of the Zavalo et al. paper is the introduction of Linked Open Data in order to make contextual programming assignments. By gathering data that are linked to each other, such as gathering movies in the 2010s and their main casts, the programming questions could have more cohesive applicability to students and allow them to grapple with programming concepts with relatable and understood content. In combination with Automatic Item Generation, this allows for a different set of incidentals to be incorporated into the radicals and increase the variations in questions.

The cost of such an inclusion would be further development time, as well as potentially backfiring and making certain questions harder because of the context for students without the cultural context the Linked Open Data provides. Alphabetically sorting common words such as “table” and “key” might be less distracting than big names like “Madonna” and “Cher.” The paper, because of its small and constrained sample size, does not have room to address these concerns.

Additionally, the paper discusses programming exercises that are more advanced concepts than what is being tested by the placement test. Any insight into translating the advanced semantic-based Automatic Item Generation into something that can be tested quickly with multiple choice potentially before a student has had any exposure to computer science concepts remains undiscussed in their paper. Still, the use of Linked Open Data or other methods of generating incidentals outside of simple number variation is worth investigating, but the problem remains first and foremost with the development of the question stems.

4.3.6 Alternatives and Consequences for Isomorphs

Because of the aforementioned issues with the lack of repetition, a program that does Automatic Item Generation would be inefficient. The benefit of generating many more questions with each iteration is eliminated without iteration.

If a crowdsourcing model was used to add more questions stems, using an Automatic Item Generation model would make the design of a single question more difficult than if questions were manually written. A contributor would have to not only write the question but decide the variables and algorithms used to create the isomorphs. At the cost of being able to say with greater certainty that each isomorph is similar to each other, manually writing the questions would not necessitate such specific, strict rules and can instead be written based purely on the experience by a domain area expert. This is greatly important for the ability for crowdsourcing additional questions. If faculty wants to add questions, having a high number of required isomorphs per question stem discourages this crowdsourcing due to the increased amount of effort.

The number of isomorphs for a hand-written radical can vary. With Automatic Item Generation, it is trivial to add more, but the expense of adding even a single clone more can be a deterrent from adding an entire question stem to a manual test writer.

Writing three isomorphs of a radical in an exam with twenty questions yields a total of 60 question items in the pool of possible questions. If 1,000 students take this exam per semester, then there would be over 300 data points per isomorph to analyze and draw conclusions from. With ten isomorphs, then the same population of students taking the exam would only generate 100 data points per isomorph, making it slightly harder to draw conclusions about the difficulty of any given isomorph. The upside of having more isomorphs of each question stem would be having a larger pool of question items such that each student is unlikely to encounter the exact same subset of question items in their exam, which could reduce the possibility of cheating.

If students can easily access the questions and there is no proctor to validate, students have a high degree of freedom to distribute any combination of questions online. With a greater number of variations, then it becomes less likely that all of the possible isomorphs of a question are publicly distributed.

4.4 Question to Exam Variations

Entangled in the discussion about utilizing automatic item generation and item pools, we had to consider how many questions would need to be included. With either a very large pool or a reliable method for generating many versions of the same question, whether two students would encounter the same exam becomes an increasingly unlikely occurrence.

A detailed analysis of each question from the provided twenty-question sample exam is provided below. In that analysis we examine how Automatic Item Generation (AIG) could be done for each question. It has not been determined how many variations of each question should be used on the actual exam, should there be any. The initial proposal was for 10 variations per each question, something that is easily doable if there is a sufficiently reliable algorithm to generate different question items. However, this exact number can be lower. Table 2 shows the number of possible question-to-exam mapping possibilities based on the number variations for each question.

Table 2: Count of question-to-exam mappings that exist assuming twenty questions with differing number of variations of each question

Number of Variations Per Question	Question-to-Exam Mapping Combinations Assuming Twenty Questions
3	$3^{20} = 3,486,784,401$

4	$4^{20} = 1,099,511,627,776$
5	$5^{20} = 95,367,431,640,625$
10	$10^{20} = 100,000,000,000,000,000,000$

Automatic Item Generation (AIG) can be achieved if the total random variations that can be calculated for each question exceeds the desired number of variations per question. The number of variations could reasonably be between three and ten. The suggested AIG approaches have the ability to generate a wider range of variations, including more than ten variations. A larger question-to-exam mapping combination results in increased exam content security. Since the exam is not proctored in any fashion, nothing prevents the student from taking screenshots of the exam and posting it to social media. Having billions of potential exam combinations reduces the risk of leaking the exam content in its exact format to social media. In deciding the number of variations for each question, it is shown that having just three variations of each base question provides enough variations for security purposes when considering the entirety of an exam, but does not address the variation on a question-to-question basis, rather than exam-to-exam.

These numbers reflect only one aspect of the security potentialities of the variations of question. While only having three variations is enough to generate a large number of exam possibilities, it does not reflect the issue of students posting individual questions, which is a more likely occurrence if our design displays a single question at a time on a given page while the student takes the exam. If there are only three variations of the same question stem, and there are one-thousand students taking that iteration of the exam, then over 300 students will have the exact same question and variation appear on their test. With ten variations, then only 100 students will have the exact same question and variation appear on their test. Thus, considering the variations in terms of repetitions per instance of a single exam, it may be more prudent to have a higher number of variations in order to reduce the chance that two students sitting next to each other in the library are not getting the exact same questions.

To contrast, having a smaller number of variations for each question also has the additional benefit of easily adding hand-written questions to the exam without the need to design a program to generate many Automatic Item Generation content variations. This manual effort required by the author of the question to determine what other numbers would result in values that are of equivalent difficulty and still maintain the integrity of the question that is being asked is significantly reduced if the number of possible values is decently small.

If the questions are made into templates for Automatic Item Generation, then the effort of the creation is much steeper at the beginning.

This tradeoff is discussed earlier in the greater discussion of Automatic Item Generation. In the context specifically of security, however, this tradeoff must be considered against the consideration specifically of security of this particular exam. For the Computer Science Placement Exam, cheating has not been identified as a major cause for concern or risk because the ultimate result of a student cheating could only harm the student in question and not the integrity of the test itself. Thus, this tradeoff leans in favor toward a smaller number of variations, but the number itself is still debatable. It is possible with certain designs of the test that variable number of variations per question stem would be most appropriate, but the design of the backend requires the flexibility to allow for this.

5 Sample Example Asset

The pool of twenty questions that was a potential candidate for the CS Placement Exam was written by Dr. Alvarez Gonzalez and provided to this senior design group. Dr. Alvarez Gonzalez no longer teaches at the University of Central Florida but authored this sample exam before leaving.

An answer key was not provided with the twenty-question sample exam and the answers were determined by the team members, whenever possible. It should be observed that the nature of this hand-written exam starts with simple questions, and then each following question builds in level of difficulty. This ordering of starting with easy questions and extending to more difficult ones may be lost if the questions are presented to the student in a random order as was suggested in our sponsor kickoff meeting.

5.1 Pseudocode Notation

The questions on the exam are written in a generic pseudocode format. It should be noted that there is no standard definition of what pseudocode looks like. It is an artificial informal language and not a programming language. The choice of notation and semantics of syntax is only dependent on the author of the pseudocode.

A suggestion that has been made is that the exam questions could be ported to a small list of programming languages such as Python, C, JavaScript, and Java, in addition to the pseudocode. These are the primary languages used in the computer science curriculum at

the University of Central Florida and popular first programming languages elsewhere. The suggestion included the idea that someone wanting to take the CS Placement Exam could indicate their language of preference among the choices prior to the start of the exam, and only the questions written in their indicated choice would be presented on their individual exam. Permitting questions to be language specific would allow for a greater variety of questions and allow for syntax specific related concepts. It would also allow the critical skill of debugging to be tested. It would remove the need to understand the artificial nature of pseudocode if desired. However, it would also result in the possibility of some questions not able to be ported precisely to all languages. For example, there are three main loop constructs used in general programming: `FOR` loops; `DO-WHILE` loops, and `WHILE` loops. The Python programming language does not contain the `DO-WHILE` syntax for loops, and therefore questions on this type of a loop construct could not be reproduced in all languages. Anything though that can be done with a `DO-WHILE` loop can be rewritten into a `WHILE` loop. Pseudocode eliminates this specific concern, but since pseudocode is an undefined standard, it can introduce its own syntax concerns.

Our initial sponsor kickoff meeting resulted in the decision that the exam should only be written in pseudocode and no other languages. The reasons for this decision are plentiful. The first and foremost is that students who are exposed to a particular programming language may not be the students who are most ready to take *Introduction to Programming with C*, given that being able to read the syntax of one language over another does not guarantee that the concepts for the class are thoroughly understood. The second is that while pseudocode has its limitations, asking students to identify which language they know comes at the risk of alienating students who have never had exposure to any programming language. Students who have no understanding of what the question is even asking might have to either select a language that they do not understand and thus have to deal with that language's syntax and quirks without having any exposure to them, and an admission to not knowing what language to pick or declining to pick a language should not automatically disqualify a student from being able to enroll in *Introduction to Programming with C*. Alienating students from taking the placement exam that is testing their readiness to begin to learn about programming is a way to cull students immediately from trying to take classes that they might otherwise succeed in. The third reason follows from the second: if there should be an option for pseudocode to be one possible language for students to take, the effort required to port questions to many languages is completely unnecessary.

For the question bank presented to the team, there were multiple instances of pseudocode explained over the course of the test, introducing new concepts as the test questions increased in difficulty. Explanations of this pseudocode form of syntax had to be provided before question 1, question 11, question 16, question 18, question 19, and question 20 for

students because no standard definition of the pseudocode in its entirety was presented to students. If the order of the questions on the exam are to be randomly shuffled for each student, it is not practical to require a pseudocode notation explanation prior to encountering the question. Indeed, for the design of our system, discussed further on, we decided that it would be prudent for the pseudocode information be made available prior to the exam, in the form of a Portable Document Format (PDF) file or a pop-up window available before the exam begins or on any given page or question of the exam.

The following pseudocode explanations are provided in the twenty-question sample exam.

5.1.1 Prior to question 1:

*“The exam is language neutral; that is, the test is designed to evaluate common programming concepts rather than knowledge about a specific language. The exam uses the symbol = to indicate the assignment operator. It uses the word “eq” to indicate the equality operator, which will return a Boolean value of True or False. Likewise, neq indicates not equal to. The word “print” is a generic term to indicate displaying the output of the value returned by an evaluatable [sic] expression that follows the word print. All arithmetic symbols are used (+, -, * and /). Variables are assumed to need no prior definition/declaration before being used.*

Only one answer is correct.”

5.1.2 Prior to question 11:

“For the next several problems, a loop construct called do repeats a block of indented code found below it. The block ends when the indentation stops. This repetition structure is entry controlled – the block of code will be executed if the control statement is valid. The iteration control statement will be found within parenthesis right after the keyword do. The keyword until permits the block of code to execute as long as the condition is NOT satisfied. That is, the block of code can execute until the conditional expression that follows it is satisfied – it does not allow the block to execute when the condition is satisfied. Conversely, the keyword while allows the block to be executed as long as the condition is satisfied. That is, it does not allow the block to execute when the condition is NOT satisfied.

For example:

do (until $x \text{ eq } 10$) will execute the block of code as long as x is not equal to 10. It will not execute the code when $x = 10$.

do (while $x < 10$) will execute the block of code as long as x is less than 10. It will not execute the code when $x \geq 10$ ”

5.1.3 Prior to question 16:

“For the next several problems, assume that an array is defined as

`<arrayName>(<index>)`

Where `arrayName` is a label like any other variable name and `index` is a positive integer that indicates the cell number in the array. The first cell in the array is at $n = 0$.”

5.1.4 Prior to question 18:

“Now assume that a two-dimensional array has two indices within the parentheses, as follows:

`<arrayName>(<row-index>, <column-index>)`

Where the first index indicates the row number and the second index indicates the column number.”

5.1.5 Prior to question 19:

“The next problem will test your knowledge of auxiliary functions – their definition and their use. Assume that the keyword `define` begins a block of code that defines an auxiliary function. A defined function has a parenthetical expression after the name that defines the values of the variables passed to it by the calling function. The format is as follows:

`define<functionName>(<var1>, <var2>, ...)`

Where `functionName` is the name given to the function being defined and `var1`, `var2` etc. are the names of the variables in the function defined that will hold the values passed to the function by the calling function. The keyword `return` as a statement in the block of code that defines the function indicates the value to be returned by the function being defined.”

5.1.6 Prior to question 20:

“For the next problem, assume that the keyword `enter` passes control to the keyboard to allow a user to enter a value. Once the processor reaches `enter`, it will await an entry and a return, and assign the value entered to the variable name listed inside curly brackets after the keyword `enter` as follows:

`enter{<var>}`)

Where the value entered by the user will be assigned to the variable `var`.”

5.2 Detailed Sample Exam Analysis

5.2.1 Question 1

For the pseudo-program below, assume that variables A, B, C and D hold integer values.

```
A = 120
B = A + 100
C = B + 30
D = C * 10
print D
```

The output of the `print` statement will be:

- a. 100
- b. 0
- c. 120
- d. 250
- e. 2500

The answer for question 1 is option e. Question 1 maps to the general or sequential focus area. Table 3 identifies the knowledge concepts that are being tested for in question 1.

Table 3: A matrix of programming and math skills tested in Question 1

Variables or identifier usage	Addition Operator: +
Assignment operator: = assigning integer	Multiplication Operator: *
Assignment operator: = assigning expression	Screen output / display
Associativity: Left to right	Math Skills: addition of 3-digit and 2-digit unsigned integers

Associativity: Right to left	Math Skills: multiplication of 3-digit and 2-digit unsigned integers
ASCII symbol for multiplication: *	

Question 1 on the sample exam was preceded by pseudocode reference material. Using automatic item generation (AIG) for question 1 could be accomplished by randomizing a value for variable A in the range of 50 to 150 in multiples of 10 inclusive resulting in 11 distinct values. Multiples of 10 are chosen to make the math easier. The goal of the exam should not require the use of a calculator. Random values in the range 50 to 150 in multiples of 10 can be used in lines two and three of the code above for the assignments to variables B and C. The multiplication in line four should be left at the number 10 to avoid the need for a calculator. With the suggested AIG substitutions, the question could generate (11)(11)(11) or 1,331 variations of the same question. The question can be improved by making the distractors closer to the actual answer. The current answer is simply the largest answer and could be guessed. Answer B as a distractor is too easy of a distractor.

Distractors using Automatic Item Generation could be made by taking the different operators that are being used in the question and creating distractors by flipping the order that the operators are applied onto the variables. This would be affected by the operators' properties, such as whether the operators were commutative, but it is one viable option. Currently, the distractors are generated in the following ways: A is one of the values that appears in the question and good other distractors would be applying different values in for different orders of the operators.

Concepts that are not covered by question 1 or as a result of using pseudocode are valid identifier naming rules that every programming language includes. It does not cover the distinct contrast between variable declaration and variable initialization. There is no coverage of data types, but the question does indicate only integer values are being used. The pseudocode notation being used contains no type declarations like the programming languages Python and JavaScript. This contrasts with programming languages such as C and Java.

5.2.2 Question 2

For the pseudo-program below, assume that variables w, x, y and z hold integer values.

w = -2

```

x = -7
y = -8
z = x - y
z = z * w
z = z / w
print z

```

The output of the print statement will be:

- a. 1
- b. 0
- c. -1
- d. -5
- e. 6

The answer for question 2 is option a. Question 2 maps to the general or sequential focus area. Table 4 lists the knowledge concepts that are being tested for in question 2.

Table 4: A matrix of programming and math skills tested in Question 2

Variables or identifier usage	Negation Operator: -
Assignment operator: = assigning integer	Addition Operator: +
Assignment operator: = assigning expression	Subtraction Operator: -
Associativity: Left to right	Multiplication Operator: *
Associativity: Right to left	Division Operator: /
ASCII symbol for multiplication: *	Screen output / display
Math Skills: subtraction of 1-digit signed integers	Math Skills: multiplication of 1-digit signed integers
Math Skills: division of 1-digit signed integers	

Using automatic item generation (AIG) for question 2 could be accomplished by randomizing the values of variables W, X, and Y to the range of -2 to -9 inclusive, resulting in 8 combinations for each variable. With the suggested AIG substitutions, the question could generate (8)(8)(8) or 512 variations of the same question. This has the risk though of increasing the problem difficulty. All the math operations in the sample problem result in single digit calculations as designed. Depending on the random values selected, the resulting question may require 2-digit numbers to be calculated for the value of variable Z and needed in the resulting multiplication and division sections.

Distractors can be generated by again changing the order of operations performed on the variables. This question also introduces the issues of accounting for signs when doing these basic arithmetic skills, such as by generating answers that neglect the signs of variables (e.g. adding -1 and -3 and getting 2 or 4 instead of -4). The assignment of the resulting values could also be assigned to variables other than the one intended. In the case of this question, for instance, if $z = z * x$, a distractor could be generated by doing $z = z * y$ instead. In the case of the question, B is a random integer that could be generated through integer division but most likely not by floating point division, C is the correct answer with the wrong sign.

Concepts that are not covered by question 2 as a result of the pseudocode is the data type behaviors of the binary division operator. In the Python programming language, an integer divided by an integer can result in a float. In the C programming language, an integer divided by an integer will always result in an integer, with remaining decimals being discarded. For this problem, the answer and all the distractors are represented by integers only, so it is acceptable for the pseudocode to skip over this very important programming concept. However, question 5 also has the binary division operator with the answer and its distractors expressed in a mix of integer data types and float data types. There is no information in the provided pseudocode notation that indicates the desired behavior for the data type result of the binary division operator using integer data types in the numerator and denominator. Depending on the random AIG values, the corresponding answer will be different among the various programming languages an examinee might be familiar with. This specific programming concept has resulting in bugs during development and is a common concern that is language dependent and cannot be expressed in the pseudocode.

5.2.3 Question 3

For the pseudo-program below, assume that variables w , x , y and z hold integer values.

```
w = 21
x = 17
y = 10
z = w + x * y
print z
```

The output of the print statement will be:

- a. 16
- b. 191
- c. 48
- d. 380

e. None of the above

The answer for question 3 is option b. Question 3 maps to the general or sequential focus area. Table 5 lists the knowledge concepts that are being tested for in question 3.

Table 5: A matrix of programming and math skills tested in Question 3

Variables or identifier usage	Addition Operator: +
Assignment operator: = assigning integer	Multiplication Operator: *
Assignment operator: = assigning expression	Operator precedence: + and *
Associativity: Left to right	Math Skills: multiplication of 2-digit unsigned integers
Associativity: Right to left	Math Skills: addition of 2-digit unsigned integers
ASCII symbol for multiplication: *	Screen output / display

Using automatic item generation (AIG) for question 3 could be accomplished by randomizing the values of variables w and x to the range of 10 to 25 inclusive, resulting in 16 distinct values. The value of y should stay at 10 to simplify the multiplication and avoid the need for a calculator. With the suggested AIG substitutions, the question could generate (16)(16) or 256 variations of the same question.

Automatic Item Generation could be used here to generate distractors that use incorrect order of operations as well as the incorrect order of variables to apply the operations to. Generating random variables from the total number of them and then applying them haphazardly would generate distractors that are all possible applications of the concepts being tested but would require a student to follow along carefully not only inline but between variable assignments as well. In the current question, the answer choice e is suboptimal. Having the final answer choice be “none of the above” is a poor writing decision [8]. Answer choice a is also a poor distractor because it is less than the starting value and there are no operators that would decrease the initial value. B is the correct answer. D is an incorrect application of the testing concept and is thus a good distractor.

Question 3 primarily adds the knowledge area or skill of operator precedence in programming languages, with only the addition and multiplication operators involved. The question can be made more difficult by adding another operator that has the same level of precedence. Many students were taught the memory mnemonic “Please Excuse My Dear Aunt Sally (PEMDAS)” as an aid to remember the rules of precedence in general mathematics studies. However, many students have forgotten that multiplication and

division have the same level of precedence and are performed left to right whether multiplication or division appear first. The corresponding memory mnemonic could therefore be PEDMSA if the orders at the same level of precedence are reversed. A common math question often presented as a challenge on social media sites is to solve $1 + 12 / 6 * 2$. With the proper precedence order as done by computer programming languages, the resulting answer is 5. However, if a strict interpretation of the memory mnemonic PEMDAS is used where the letter for multiplication comes before the letter for division, then the resulting answer is 2. Therefore, this question fails to fully test the students understanding for the order of precedence for the basic math binary operators as used in programming languages. The order of precedence should be restricted to the basic operators discussed in the memory mnemonic PEMDAS. Programming languages like C define 15 levels of operator precedence around 49 different operators [9]. The number of levels and operators is programming language dependent and therefore not able to be easily expressed in pseudocode.

5.2.4 Question 4

For the pseudo-program below, assume that variables `var1`, `var2`, `var3`, `var4`, `var5` and `var6` are able to hold Boolean values and can evaluate to either `True` or `False`. Furthermore, the `print` function will print the value of such variables.

```
var1 = True
var2 = False
var3 = False
var4 = True
var5 = True
var6 = (var1 OR var2) AND (NOT(var3) OR var4) AND var5
print var6
```

The output of the `print` statement will be:

- a. False
- b. True

The answer for question 4 is option a. Question 4 maps to the general or sequential focus area. Table 6 lists the knowledge concepts that are being tested for in question 4.

Table 6: A matrix of programming and math skills tested in Question 4

Variables or identifier usage	Boolean operator: AND (logical conjunction)
-------------------------------	---

Assignment operator: = assigning integer	Boolean operator: OR (logical disjunction)
Assignment operator: = assigning expression	Boolean operator: NOT (logical negation)
Associativity: Left to right	Parenthesis operator
Associativity: Right to left	Screen output / display

Using automatic item generation (AIG) for question 4 could be accomplished by randomizing the values of variables `var1`, `var2`, `var3`, `var4`, and `var5` to the values `True` or `False`. With the suggested AIG substitutions, the question could generate (2)(2)(2)(2)(2) or 32 variations of the same question. To increase the number of variations, the Boolean conjunctive and disjunctive operators could also be randomized resulting in (32)(2)(2)(2)(2) or 512 variations.

Automatic Item Generation distractors here could also rely on confusing different orders and groupings of operators as well as by incorrectly applying Boolean operators correctly, such as evaluating `True OR False` to be `False` rather than `True`. Distractors could also be generated by assuming all variables to be of one value, `True` or `False`, and applying correct logic onto them.

Question 4 primarily adds the testing concept of Boolean operators and the usage of the parenthesis operator to change the order of precedence. The usage of parenthesis is needed in this question. In the C programming language with 15 levels of precedence among around 49 different operators, the logical NOT operator has a precedence level of 2, the logical AND operator has a precedence of 11 and the logical OR operator has a precedence of 12 [9]. Memorization of this level of detail is not common among beginning programmers and leads to a high degree of ambiguity if the parenthesis operator is not used. The usage of the parenthesis operator in this question removes the need to memorize concepts like conjunction over disjunction. The initial sponsor kickoff meeting indicated that `True/False` exam questions should be eliminated, as there is a 50 percent chance of guessing the right answer. This is the only sample exam question to use the `True/False` type of question. Additional distractors could possibly be “NULL” or “The answer cannot be determined” to eliminate the ability to guess correctly 50 percent of the time.

An alternative phrasing of this question such that it might be more challenging, but would more easily fit the format of the answers as they exist now, is to have the question shift away from “what is the answer of the logical operations” and toward “which values, when these logical operations are applied onto them, yield this answer.” This would allow distractors to simply be any of the possible values and the need for distractor generation is only a list of the variables used or a subsection of them.

5.2.5 Question 5

For the pseudo-program below, assume that variables $n1$, $n2$, $n3$ and $n4$ hold integer values. Furthermore, assume that the operator `mod` computes and returns the remainder of the integer division (i.e., the *modulo*).

```
n1 = 12
n2 = 5
n3 = n1/n2
n4 = n1 mod n2
print n3, n4
```

The output of the print statement will be: (don't worry about the formatting)

- a. 2.40 0.40
- b. 17 2
- c. 60 12
- d. 2 2
- e. None of the above

The answer for question 5 is option d. Question 5 maps to the general or sequential focus area. Table 7 lists the knowledge concepts that are being tested for in question 5.

Table 7: A matrix of programming and math skills tested in Question 5

Variables or identifier usage	Division Operator: /
Assignment operator: = assigning integer	Modulo operator: % (remainder)
Assignment operator: = assigning expression	Math Skills: division of 2-digit unsigned integers with 1-digit unsigned integers
Associativity: Left to right	Math Skills: Remainder calculation
Associativity: Right to left	Screen output / display

Using automatic item generation (AIG) for question 4 could be accomplished by randomizing the values of variable $n1$ to the range 10 to 19 inclusive, resulting in 10 distinct values. The variable $n2$ can be randomized to the range 2 to 9 inclusive, resulting in 8 distinct values. With the suggested AIG substitutions, the question could generate (10)(8) or 80 variations of the same question.

The distractors for this question as could be generated by Automatic Item Generation could relate to an improper understanding of the modulo operator, incorrect order of precedence,

and incorrect variable assignments after operations have been completed. As mentioned below, the issue of integer division vs. floating point division is significant enough that the distractors themselves would need to have more strict rules governing them than previous questions. For one, the question stem could be arranged in such a way that integer division or not, there is no remainder during division, reducing the contention between the two types of divisions, leaving the remaining areas of distraction to be different number combinations.

A is dependent on student understanding of integer division, which may not be ideal. The other distractors are poor in that there is no instance where $n3$ is printed after $n4$, which would reinforce that there is an important difference between the two but that the concepts are related.

Question 5 is very similar to question 2, but with the inclusion of the modulo operator. Having to demonstrate the precedence level of the modulo operator when combined with other operators is avoided here since the modulo operator is in isolation of other operators. The feedback for question 2 discussed the issues of the return data type when doing integer division, and that it can behave in a language dependent manner. This issue still appears to be the case for problem 5 and is further compounded by the answer choices containing a mixture of floating point and integer results. However, this has been mitigated by the wording of the question that states $n3$ and $n4$ hold integer values. If it was not stated that $n3$ holds integer values, then $n3 = 2.40$ would be the value in the Python programming language. In the C programming language, the result would be $n3 = 2$. It is imperative that the question retains this explanation of the behavior as it pertains to the pseudocode, otherwise the behavior is programming language specific. The knowledge or skills of the modulo function may be a challenge to students in the bridge mathematics courses at the University of Central Florida.

5.2.6 Question 6

For the pseudo-program below, assume that variables x , y and z hold integers.

```
x = 10
y = 15
z = x + y
if (x>y) then print z
if (y>x) then print (z + x + y)
```

The output of the print statement will be:

- a. 15
- b. 0
- c. 25 50
- d. 25
- e. 50

The answer for question 6 is option e. Question 6 maps to the conditionals focus area. Table 8 lists the knowledge concepts that are being tested for in question 6.

Table 8: A matrix of programming and math skills tested in Question 6

Variables or identifier usage	Addition Operator: +
Assignment operator: = assigning integer	Relational operator: > (greater than)
Assignment operator: = assigning expression	Conditional statement: IF/THEN
Associativity: Left to right	Math Skills: addition of 2-digit unsigned integers
Associativity: Right to left	Screen output / display
Operator precedence: ()	

Using automatic item generation (AIG) for question 6 could be accomplished by randomizing the values of variable x and y to the range 10 to 19 inclusive, resulting in 10 distinct values each. The case where x equals y must be excluded from the randomization, as this would result in no output based on the design of the question. With the suggested AIG substitutions, the question could generate $(10)(9)$ or 90 variations of the same question.

Distractors using Automatic Item Generation could be made by outputs of all IF/Then statement. Answer option c and d are pretty good distractor. But answer option both a and b are not a good practice of distractor, those distractor options never would be print. It can be replaced by other distractor, for example: the case where x equals y is been excluded from the randomization, but it can be a good distractor for this question, either one of answer options a or b can be replaced as “Nothing is printed”. Currently, question 6 have the answer choice calculate generated by the following way: choice a is the value of variable y . Answer choice b is a constant value of zero. Answer choice c is the value of variable z and $x + y + z$. Answer choice d is the value of variable z . E is the value of variable $x + y + z$, which is the correct answer.

Question 6 primarily adds the testing concept of IF/THEN conditional statement and the greater than operator. Pseudocode has no standard format and the author choose to express the IF statement with the THEN keyword following the conditional expression. The four primary programming languages used at the University of Central Florida are Python, C, Java, and JavaScript. Those four programming languages do not have a THEN keyword to go with the associated IF statement. Since pseudocode is designed to resemble the English language, it is acceptable to leave the THEN keyword in the code fragment, as the THEN keyword is used in other programming languages. It expresses a cause and effect situation, which complements the English pseudocode reading.

5.2.7 Question 7

For the pseudo-program below, assume that variables x and y hold integers.

```
x = 10
y = 10
if (x>y) then print x
if (y>x) then print y
```

The output of the print statement will be:

- a. 15
- b. 0
- c. 10 10
- d. 10
- e.

The answer for question 7 is option e. Question 7 maps to the conditionals focus area. Table 9 lists the knowledge concepts that are being tested for in question 7.

Table 9: A matrix of programming and math skills tested in Question 7

Variables or identifier usage	Conditional statement: IF/THEN
Assignment operator: = assigning integer	Relational operator: > (greater than)
Assignment operator: = assigning expression	Screen output / display (No Output)
Associativity: Left to right	Associativity: Right to left

Using automatic item generation (AIG) for question 7 could be accomplished by randomizing the values of variable x to the range 0 to 20 inclusive, resulting in 21 distinct

values. In order to achieve the missing situation that x equal to y , the choice of y should always equal to x , resulting in 1 distinct value. With the suggested AIG substitutions, the question could generate (21)(1) or 21 variations of the same question. Although, to increase the number of variations is very easy for this question. The variable x can be increased to whatever number range is needed without impacting the difficulty of this question.

Distractors using Automatic Item Generation could be made by outputs of all IF/Then statement, both answer option c and d are good choice for distractors. One will print the output of both $y > x$ and $x > y$ conditions. The last distractor is yielded when a student assumes the program will print out only one of the statements that has been executed. Other than that, because few numbers are used in the question, there are not many good distractors that can be generated. Currently, question 7 have the answer choice calculate generated by the following way: Answer choice a is a constant value of 15, come from nowhere, may be question 6. Answer choice b is a constant value of zero. Answer choice c is the value of variable x and y . Answer choice d is the value of variable x . Answer choice e is “a choice of empty block”, which represents that nothing is printed, which is the correct answer.

Question 7 is very similar to question 6 with the exception being the variables are equal and the two relational operators used do not include the test option for equality. This exception will make the two conditional statements return a value of false and resulting in an empty output. This is the first question in this exam with an empty answer. It is a very poor answer design as it appears there was an exam creation error for option e. Option e should at least contain some phrase such as “Nothing is printed”. Concepts that are not covered in question 7 are the other relational operators: \geq and \leq . Those additional two relational operators do not appear for any sample exam question.

5.2.8 Question 8

The binary number 0011010011001100 is equal to what decimal value:

- a. 15,256
- b. 8,600
- c. 990
- d. 13,516
- e. 2,048

The answer for question 8 is option d. Question 8 does not map to one of the four focus areas of the exam. Question 8 tests only a math skill: base 2 to base 10 conversion.

Using automatic item generation (AIG) for question 8 could be accomplished by randomizing the binary number. Since, the goal of the exam should not require the use of a calculator, 16-bit conversion are very large to deal without using the calculator. This question can be improved by decreasing the random binary value range, and the making the distractors closer to the actual answer. To achieve that, a random binary value ranging from 0010 to 1111 inclusive should be used, resulting 14 distinct values. The binary values 0000 and 0001 are excluded as being trivial. With the suggested AIG substitutions, the question could generate 14 variations of the same question.

Without making the change of this question, this question has enough difficulty when doing the conversion, if distractors are closer to the actual answer, this question will be insanely difficult and easy to make algebra mistake during the calculation. It can be reduced difficulty by making distractors away from the actual answer, but that will make this question guessable. Currently, question 8 have the answer choice calculate generated by unknown formula.

Question 8 primarily adds the testing math concept of Base 2 to Base 10 conversion. This question does not belong to any of the four focus areas of this exam. It appears to be testing only mathematical concepts and not programming specific concepts. There are no pre-requisites for the two introductory programming courses at the University of Central Florida. This implies that students who are not eligible to enroll in college level math courses due to their evaluation on the Math Placement Test (MPT) must take a bridge math course but are potential candidates for the programming courses. Questions of this context may exceed the expected math skills for some students. The programming aspect of this may be trivial with some programming languages having the ability to print an integer in binary, octal, decimal, and hexadecimal formats natively. In other programming languages, this might require an algorithm written, which is beyond the scope of this exam. It is our recommendation that this question be removed from the exam due to the lack of programming concepts.

5.2.9 Question 9

Which statement below detects whether the integer variable x is between -4 and 4 (inclusive)?

- a. $(x < -5) \text{ AND } (x < 5)$
- b. $(x > -4) \text{ OR } (x < -4)$
- c. $(x > -5) \text{ OR } (x < 5)$
- d. $(x > -5) \text{ AND } (x < 5)$

e. None of the above

The answer for question 9 is option d. Question 9 does not map well to one of the four focus areas. Table 10 lists the knowledge concepts that are being tested for in question 9.

Table 10: A matrix of programming and math skills tested in Question 9

Boolean operator: AND (logical conjunction)	Relational operator: < (less than)
Boolean operator: OR (logical disjunction)	Relational operator: > (greater than)
Vocabulary: inclusive vs. exclusive	Math Skills: Number Line
Operator precedence: ()	

Using automatic item generation (AIG) for question 9 could be accomplished by randomizing the values of variable x to be between the range -9 to 9 inclusive excluding the trivial case of 0, resulting in 18 distinct values. Since x is defined with two values, an upper and a lower boundary, the number of variations can be increased if desired by setting these boundaries to different values. It is also possible to randomize the Boolean operators, but consideration would be needed to verify that an actual answer exists. With the suggested AIG substitutions, the question could generate at least 18 variations of the same question.

Distractors using Automatic Item Generation could be made by changing the relational operator, Boolean operator or value of actual answer. There are many combinations when changing these operators, therefore, answer option e. None of the above is not recommend to be using in most of the question when there is lots of choice for distractor, it is not a good distractor of using Automatic Item Generation.

Question 9 primarily adds testing the math concept of a number line and understanding the vocabulary inclusive versus exclusive. The design of the problem does not use the relational operators \leq and \geq , and therefore requires the answer to use different numbers than the upper and lower boundary numbers. The answer distractors make use of this and include an answer that is equal to the boundaries.

5.2.10 Question 10

For the pseudo-program below, assume that variables A, B, C, D, E, F and G hold integer values and are all initialized to 0 prior to reaching the pseudo-code below:

```

A = 10
B = 20
C = 30
if (A eq B) then D = 100
    else if (B > C) then E = 120
    else if (C > A) then F = 200
    else if ((A+B) eq C) then G = 250
print D, E, F, G

```

The output of the print statement will be:

- a. 0 0 200 0
- b. 100 120 200 0
- c. 0 0 200 250
- d. 100 120 200 250
- e. None of the above

The answer for question 10 is option a. Question 10 maps to the conditionals focus area. Table 11 lists the knowledge concepts that are being tested for in question 10.

Table 11: A matrix of programming and math skills tested in Question 10

Variables or identifier usage	Conditional statement: IF/THEN/ELSE IF
Assignment operator: = assigning integer	Relational operator: > (greater than)
Associativity: Left to right	Equality operator: eq
Associativity: Right to left	Operator precedence: ()
Math Skills: addition of 2-digit unsigned integers	Screen output / display

Using automatic item generation (AIG) for question 10 could be accomplished by randomizing the values of variable a to the range 1 to 19 inclusive, resulting in 19 distinct values each and the value of variable b to the range 20 to 29 inclusive, resulting in 10 distinct values. A key answer distractor is the case when $a + b = c$, so the variable c should be left to that calculated value. With the suggested AIG substitutions, the question could generate (19)(10) or 190 variations of the same question.

Distractors using Automatic Item Generation could be made by outputs of all IF/THEN/ELSE IF statements. Answer option c is an excellent choice for distractors. It is straight to the point and tests a student's understanding of IF/THEN/ELSE IF

statement. Besides that, other distractors can be generated from the combination of outputs for each IF/Then/ELSE IF statement. Lastly, As mention above, answer option e. None of the above is not recommended as a common multiple choice distractor, especially when most of the questions offer lots of choices for distractor.

Because this answer to this question is a set of numbers, and all variables that waiting for output is initialize to zero. question 10 have the answer choice calculate generated by the following way: A is a set of numbers when only third condition statement would be executed, which is the correct answer of this question. B is a set of numbers when first three condition statements would be executed. C is a set of numbers when last two condition statements would be executed. D is a set of numbers when all condition statements would be executed. E is None of above.

Question 10 primarily adds the testing math concept of the conditional statement IF/THEN/ELSE IF. It is the first question that uses the equality operator `eq`, and its purpose is to be a distractor among the answers to this question. This exam jumped from the simple IF/THEN type of statement directly to ELSE IF conditional statement types. Nowhere on the sample exam did the simple ELSE style construct appear for indication of a mutually exclusive selection or choice.

5.2.11 Question 11

For the pseudo-program below, assume that variables `x` and `y` hold integers and are initialized to 0 before reaching these statements:

```
do (until x eq 20)
  y = y + 1
print y
```

The output of the print statement will be:

- a. 19
- b. 20
- c. 21
- d. 0
- e.

The answer for question 11 is option e. Question 11 maps to the loops focus area. Table 12 lists the knowledge concepts that are being tested for in question 11.

Table 12: A matrix of programming and math skills tested in Question 11

Variables or identifier usage	Screen output / display: <code>print</code>
Assignment operator: <code>=</code> assigning expression	Loop statement: <code>DO (UNTIL ...)</code>
Addition Operator: <code>+</code>	Equality operator: <code>eq</code>
Associativity: Left to right	Infinite loops
Associativity: Right to left	

Using automatic item generation (AIG) for question 11 could be accomplished by randomizing the integer that our variable, `y`, is being incremented by. We could also randomize the number that we are checking for `x` to be equal to. This would be a simple task for this problem since the numbers here really only act as a distraction to the main concept being tested. However, a number between 10 and 20 should be used to check against in our conditional statement to keep the problem simple. For a similar reason, the integer `y` is being incremented by should be no more than 3 and greater than 0. With the suggested AIG substitutions, the question could generate at least 33 variations of the same question.

Distractors using Automatic Item Generation for this question are sufficient as is because it is testing to check if the student can catch the infinite loop. With this in mind, it should be good enough to provide an answer that would appear if `x` was actually being incremented as well as two answers that are close to that answer.

Question 11 primarily adds the answer choice d as a test to see if the student can catch that our value inside the loop is not related to the value being checked to end the loop. The answer would be None of the above because we will never reach the end of this loop as it continues infinitely. The problem with this question structure is that the `DO (UNTIL ...)` pseudocode can be tricky to understand, even with prior programming experience. Even with the given instructions explaining how the pseudocode works, I believe that we could use a better structured syntax. Furthermore, using a `DO (WHILE ...)` loop here seems unnecessary as we could easily use a while loop and remove some of the confusion.

5.2.12 Question 12

For the pseudo-program below, assume that variables `n`, `k` and `p` hold integer values and are initialized to 0 prior to reaching these statements.

```

k = 5
do (until n eq 5)
    p = p + k
    n = n + 1
print p

```

The output of the print statement will be:

- a. 30
- b. 0
- c. 12
- d. 25
- e. 5

The answer for question 12 is option d. Question 12 maps to the loops focus area. Table 13 lists the knowledge concepts that are being tested for in question 12.

Table 13: A matrix of programming and math skills tested in Question 12

Variables or identifier usage	Loop statement: DO (UNTIL ...)
Assignment operator: = assigning variable	Equality operator: eq
Assignment operator: = assigning expression	Multiple variable value tracking
Addition Operator: +	Screen output / display: print

Using automatic item generation (AIG) for question 12 could be accomplished by randomizing the number of times to loop through the two expressions. We could do this by changing the value we check *n* against. Another randomization that could be implemented would be initializing *k* or *p* to some other number. The integer we use to perform a conditional check against *n* with should be on the range 3 to 10, resulting in 8 distinct values. The random integer we choose to set *k* and *p* to should be on the range 1 to 10, resulting in 10 distinct values for both *k* and *p*. With the suggested AIG substitutions, the question could generate $(8)(10)(10) = 800$ variations of the same question.

Distractors using Automatic Item Generation could be created by tracking *p* instead of *n* throughout the loop. Another way would be to have the end results of *n* and *k* as options instead of *p*. The remaining options could result in checking if the student correctly counted the number of loops that the program went through. This could be done by having a different answer that would result from looping one too many or one too few times.

Question 12 is a well-structured problem but, like question 11, fails to provide a simple syntax that requires minimal explanation. This question forces the student to track the values of multiple variables throughout the program's execution. The student will first need to determine the number of times the two expressions loop and then calculate the final value of p that is printed to the screen. This is a simplistic question, but its simplicity lends itself to accurately testing multiple concepts simultaneously.

5.2.13 Question 13

For the pseudo-program below, assume that variables n1 and n2 hold integer values and that these variables have been initialized to 0 a priori. Furthermore, assume that the operator mod computes and returns the remainder of the integer division (i.e., the *modulo*).

```
do (until n1 eq 20)
  if ((n1 mod 2) eq 0) then n2 = n2 + n1
  n1 = n1 + 1
print n2
```

The output of the print statement will be:

- a. 0
- b. 20
- c. 72
- d. 90
- e. 110

The answer for question 13 is option d. Question 13 maps to the loops focus area. Table 14 lists the knowledge concepts that are being tested for in question 13.

Table 14: A matrix of programming and math skills tested in Question 12

Variables or identifier usage	Loop statement: DO ... WHILE
Assignment operator: = assigning expression	Equality operator: eq
Addition Operator: +	Multiple variable value tracking
Binary operator: mod (modulus)	Conditional Statement: IF / THEN
Screen output / display: print	

Using automatic item generation (AIG) for question 13 could be accomplished by randomizing the conditional check integer, the integer being used to mod n1, the integer being checked on the right side of the conditional of the n1 mod 2 expression, or the

integer $n1$ is incremented by. For this to work we would have to guarantee that $n1 \bmod x$ would equal y on at least one of the loops for some integers x and y . The AIG on this would be a little more complicated since we are working with the `mod` operator. The variables $n1$ and $n2$ should remain initialized to 0 to prevent too much confusion. We can change the integer on the right side of the conditional to be on the range from 10 to 20, resulting in 11 distinct values. We could also change the value $n1 \bmod 2$ is being checked against to 1, resulting in 2 distinct values. With the suggested AIG substitutions, the question could generate (11)(2) or 22 variations of the same question.

Distractors using Automatic Item Generation could be created by displaying options where $n1$ is `mod` by an odd number. We could also do something similar to the question 12 where we test if the student correctly counted how many loops to go through to get the correct answer.

Question 13 primarily adds testing the `mod` operator, and is a good question for testing the `IF / THEN` statement, and the ability to keep track of variable values in a loop. This is a difficult question because it requires you to track variables being modified by different operators within a `DO / UNTIL` loop. We could improve this question by changing the `DO / UNTIL` loop syntax and possibly making it easier to do so that you do not need a calculator to complete it.

5.2.14 Question 14

For the pseudo-program below, assume that variables A and B hold integers and are initialized to 0 before reaching these statements:

```
A = 1
do (until A eq 20)
    A = A + 2
    B = B + 1
print B
```

The output of the print statement will be:

- a. 19
- b.
- c. 21
- d. 0
- e. 20

The answer for question 14 is option b. Question 14 maps to the loops focus area. Table 15 lists the knowledge concepts that are being tested for in question 14.

Table 15: A matrix of programming and math skills tested in Question 12

Variables or identifier usage	Loop statement: DO (UNTIL ...)
Assignment operator: = assigning variable	Equality operator: eq
Assignment operator: = assigning expression	Infinite loops
Addition Operator: +	Multiple variable value tracking
Screen output / display: print	

Using automatic item generation (AIG) for question 14 could be accomplished by randomizing the integer A is initialized to, the value that A is checked against inside the conditional, or the values that A and B are incremented by. An important factor to keep in mind when randomizing these values is to make sure that the value being checked against in the conditional is (or is not) divisible by A's initial value plus the value it is incremented by. If we initialize A on the range from 0 to 5 and loop until equal to the range of 10 to 20, we will have 6 and 11 distinct values respectively. We could also increment B by a value on the range from 1 to 5 for another 5 distinct values. With the suggested AIG substitutions, the question could generate (6)(11)(5) or 330 variations of the same question.

Distractors using Automatic Item Generation could be created by printing A instead of B. This question is a trick question so having the answer that would have been correct if A actually ever equaled to 20 is a good decision. Once we have that number, we can add or subtract 1 from it to further distract the student.

This question is a bit confusing because it is not entirely obvious whether the concept being tested is to catch the infinite loop that results in incrementing A past 20, or if that is an error on the question creator's part. If the main concept is catching an infinite loop, then this concept was already covered in question 11. Otherwise, this question is basically a variation of question 12 where we print the number of times the loop was run through instead of the result of an alternate expression. It is also unclear why capital letters were chosen to represent this question's variables whereas lowercase letters were used in the previous examples. We could improve this question by changing the concepts being tested and by solidifying whether we want to test for infinite loops or not. Another option would be to remove this question entirely since, regardless of the possible errors, it is already a derivative of question 11 and question 12.

5.2.15 Question 15

For the pseudo-program below, assume that variables x , y and z hold integers and are initialized to 0 before reaching these statements:

```
x = 0
y = 2
z = 0
do (until z eq 49)
    if (z < 26) then (x = x + 1)
    else (y = y + (x + x mod 2))
```

Within this segment of code, how often does y contain an even integer?

- a. Variable y never contains an even integer.
- b. Variable y always contains an even integer.
- c. Variable y contains an even integer sometimes, but not always.
- d. The pseudo-code does not provide enough information to answer this question.

The answer for question 15 is option b. Question 15 maps to the loops focus area. Table 16 lists the knowledge concepts that are being tested for in question 15.

Table 16: A matrix of programming and math skills tested in Question 12

Variables or identifier usage	Screen output / display: print
Assignment operator: = assigning variable	Loop statement: DO (UNTIL ...)
Assignment operator: = assigning expression	Equality operator: eq
Relational operator: > (greater than)	Multi variable value tracking
Addition Operator: +	Conditional Statement: IF / THEN / ELSE
Binary operator: mod (modulus)	Infinite iteration

Using automatic item generation (AIG) for question 15 could be accomplished by randomizing the integers x , y , and z are initialized to. We could also randomize the conditional check of the if statement, the integer x is incremented by, and the integer used to mod x in the else statement. Variables x , y , and z can be initialized on the range from 1 to 10, resulting in 10 distinct values each. The integer used on the conditional check of the if statement and the number used in the loop until equal check can be on the range from

11 to 50, resulting in 40 distinct values each. The variable x can also be incremented by a number on the range from 1 to 5, resulting in 5 distinct values. With the suggested AIG substitutions, the question could generate (10)(10)(10)(40)(40)(5) or 8 million variations of the same question. The reason we have so many different variations is because this question is only used to check that y is initialized to an odd or even number.

Distractors using Automatic Item Generation are not a good choice for this question because all of the answers are short responses based on the concepts in the question. With that, the answers provided would be sufficient distractors because they cover all possible outcomes of the program.

Question 15 is a tricky question because the student must realize that z is not being incremented and so it will continue to loop infinitely. The variable y is initialized to 2 so we can already rule out a being the correct answer. Variable z will never be greater than 26 and so y will never change values. Thus, y will remain even indefinitely. This question is great if its intention was to see if students could catch the infinite loop, but this concept has already been tested several times. Instead, we could increment z in the THEN portion of the IF statement and see if students can correctly answer the question then.

5.2.16 Question 16

For the pseudo-code program below, assume that an array named `my_array` has 10 cells and has already been populated with floating point numbers.

```
n = 0
x = 0.0
y = 0.0
do (until n eq 9)
    x = x + my_array(n)
    n = n + 1
y = x / n
print y
```

If `my_array` contains the values [2.8, 9.1, 16.7, 24.0, 2.1, 17.9, 30.6, 1.9, 10.5, 4.8], the output of the program will be:

- a. 12.84
- b. 13.38
- c. 25.9
- d. 12.04

The answer for question 16 is option a. Question 16 maps to the loops focus area. Table 17 lists the knowledge concepts that are being tested for in question 16.

Table 17: A matrix of programming and math skills tested in Question 16

Variables or identifier usage	Addition Operator: +
Assignment operator: = assigning value	Division Operator: /
Assignment operator: = assigning expression	Math Skills: addition of 3-digit floating point numbers
Data types: Floating point numbers	Math Skills: division of 3-digit floating point numbers
Data structures: Arrays	Loop statement: DO (UNTIL ...)
Loop statement: Counter controlled iteration	Arrays: indexing with 1 dimension
Screen output / display: print	Debugging: Off-by-one errors
Equality operator: eq	Associativity: Left to right
Associativity: Right to Left	

Using automatic item generation (AIG) for question 16 could be accomplished by randomizing the ten indexes of the array to be any integer from zero to 20 inclusively, resulting in 21 distinct values each. There are no restrictions on which values can be used together in the array, so the total number of variations to this question would be $(21)(10)$, which is 210.

Distractors using Automatic Item Generation could be made by making one of the answer choices be all ten values summed up since the right answer only accounts for the first nine. Another distractor answer choice could be the last nine indexes being added together. This would catch the students who realize it only iterates nine times, but that do not know array indexing. This would maintain the testing concepts but make the correct answer less obvious.

Question 16 primarily adds to the testing concept of loops although it contains array indexing and mixed data type division as well. Since mixed data type division is done differently amongst programming languages, it would be best practice for this examination to either use the same data type or specify in the pseudocode explanation how mixed data type arithmetic is handled. The array contains ten indexes, but the DO (UNTIL ...) loop only iterates nine times. Whether or not this is a part of the testing concept is unsure, but worth bringing up in question development. Once $n = 9$ in the code segment, the loop

does not execute again for that value. The current float values of the array indexes seem unnecessary for the testing concept. It complicates the math any student would have to do and if there is any randomization, some decimal values could be much easier to work with than others. As we are trying to avoid difficulty variance with AIG, this question should be heavily modified in its values.

5.2.17 Question 17

For the pseudo-code program below, assume that we use the same array `my_array` and with the same floating point values of problem 16 above, except that we now know that the numbers are in the range of 0.0 to 1,000.0.

```
n = 0
x = 0.0
do (until n eq 9)
    if (my_array(n) < x) then x = my_array(n)
    n = n + 1
print x
```

The output of this program (x) will be:

- a. 1,000.0
- b. 0.0
- c. 1.9
- d. 2.1
- e. 30.6

The answer for question 17 is option b. Question 17 maps to the loops focus area. Table 18 lists the knowledge concepts that are being tested for in question 17.

Table 18: A matrix of programming and math skills tested in Question 17

Variables or identifier usage	Addition Operator: +
Assignment operator: = assigning variable	Relational operator: < (less than)
Assignment operator: = assigning expression	Math Skills: addition of 2-digit unsigned integers
Equality operator: eq	Screen output / display: print
Data types: Floating point numbers	Math Skills: division of 3-digit floating point numbers
Data structures: Arrays	Loop statement: DO (UNTIL ...)

Loop statement: Counter controlled iteration	Arrays: indexing with 1 dimension
Debugging: Off by one errors	Associativity: Left to right
Associativity: Right to Left	

Automatic item generation (AIG) is not very obvious for question 17. We could try to accomplish it by randomizing the value of x , but this may be tricky as the answer to the question depends on the array values from the previous question.

Distractors using Automatic Item Generation could be made by following the train of thought that would be taken if the student does not catch that the `if` statement is never entered. The distractors of this question would be the last three indexes of the given array: 1.9, 10.5, 4.8. This would provide a good distractor as the student may think the `if...then...` statement is entered and executed each time.

Question 17 primarily adds to the testing concept of loops, arrays, and array indexing. This question design does not work well with the randomized model as it is missing the array values and array definitions which are included in other question stems. If a student had to go between questions for all of the data they needed, it would increase overall test taking time, and affect our possible metrics. As we have discussed prior, time spent on each question could be generated by how long a student spends on each web page. If they need to go back two questions for help on their current question, that metric would not be as accurate or relevant as time goes on. Since $x = 0$ in this question, and every value in `my_array` is positive, the `if` statement is never executed. If this is the intention of the testing concept, then it works. However, if the student is intended to enter the `if` statement at least once, x cannot be 0, or `my_array` needs to contain at least one negative number.

5.2.18 Question 18

For the pseudo-program below, we define two arrays, a one-dimensional array called `array1` which has 10 cells, and a two-dimensional array called `array2` which has 10 rows and 10 columns (10x10). `array2` has been previously populated with arbitrary character values, while `array1` has been initialized with integers valued at 0.

```
n = 0
k = 0
do (while n < 10)
    do (while k < 10)
```

```

        if (n eq k) then array1(n) = array2(n, k)
        k = k + 1
    n = n + 1

```

What does this program do?

- a. Nothing
- b. Takes the average of each row of array2 and puts it in array1
- c. Takes the maximum value of every row of array1 and puts it into array2
- d. After completion, array1 will contain the values of the diagonals of array2.
- e. Takes the minimum value of every column of array2 and puts them in array1.

This question has no correct answer. The inner loop iterator variable is not reset for each outer loop iteration. Question 18 maps to the loops focus area. Table 19 lists the knowledge concepts that are being tested for in question 18.

Table 19: A matrix of programming and math skills tested in Question 18

Variables or identifier usage	Addition Operator: +
Assignment operator: = assigning variables	Equality operator: eq
Assignment operator: = assigning expression	Math Skills: addition of 2-digit unsigned integers
Equality operator: eq	Conditional statement: IF/THEN
Data types: Floating point numbers	Loop statement: Counter controlled iteration
Data structures: Arrays	Loop statement: DO (WHILE ...)
Data structures: two dimensional arrays	Arrays: indexing with 1 dimension
Nested loops	Arrays: indexing with 2 dimensions
Math Skills: knowledge of matrix and diagonal meaning	Associativity: Right to Left

This question cannot use automatic item generation (AIG) because it is testing a concept without any numeric manipulation. This question would be hard to develop variations for.

Distractors using Automatic Item Generation would be harder for this question than others because it has written answers rather than numbers. This question has a bug in it and does not work properly, but with slight modification we could develop distractors. If the question executed as the writer probably intended, and the correct answer was the diagonal, d, then we could develop some distracting elements. One of the other answer choices could be the verbiage of the correct answer but switching array1 and array2.

This question primarily adds to the testing concept of iterating through arrays in loops. There is a nested DO (WHILE ...) loop that only really iterates once because k is never reinitialized to 0 and never enters the second while loop again. The verbiage used in the answer choices may not be clear to all students, such as the supposed right answer including the phrase “values of the diagonal.” This can be inferred based on context and understanding of a two-dimensional array, but because it is not computer science or programming language, it may not be fair to rely on inference. The pseudocode explanation on this question will need to be more extensive to give context to the terminology that is referenced in the answer choices.

5.2.19 Question 19

For the pseudo-code program below and its auxiliary function:

```
x = 9
y = 100 + sqr(x)
print y

define sqr(x)
    a = x * x
    return a
```

The value of y printed out will be:

- a. 81
- b. 181
- c. 9
- d. 100
- e. 109

The answer for question 19 is option b. Question 19 maps to the functions focus area. Table 20 lists the knowledge concepts that are being tested for in question 19.

Table 20: A matrix of programming and math skills tested in Question 19

Variables or identifier usage	Assignment operator: = assigning variable
Data types: Integers (unsigned and signed)	Assignment operator: = assigning expression
Associativity: Left to right	ASCII symbol for multiplication: *
Addition Operator: +	Multiplication Operator: *
Functions: Defining	Functions: Single argument
Functions: Returning integer	Screen output / display: print

Math Skills: addition of 3-digit and 2-digit unsigned integers	Associativity: Right to Left
--	------------------------------

Using automatic item generation (AIG) for question 19 could be accomplished by randomizing the values of x and the integer value in the equation for y . The values of x should range from 2 to 9 inclusively, resulting in 7 distinct values for x . Then y could be any multiple of 10 for simplicity, up to 200 inclusively, resulting in 20 distinct values for y . The total number of variations to this question with AIG would be $(7)(20)$, or 140 variations.

These answer choices already represent good distractors using Automatic Item Generation. The answer choices contain both hard numbers given by the program, 100 and 9, the square of x , 81, the incorrect addition of these numbers, 109, and the correct answer, 181. The only other distractor could be squaring after the addition of 100 and 9, but this larger number, 11,881, may be obviously wrong to a student taking the exam.

The main testing concept of this question is the use and return of functions. Some programming languages have a predefined function `sqr()` which reproduces the square root function. This could be misleading to a student with programming background in those specific languages. Trying to redefine these functions from a library could result in a namespace type error. The distractors should be developed more strategically since this question could be easily guessed correctly. For a function-based exam question, we should target more on function properties than the arithmetic associated with this example.

5.2.20 Question 20

For the pseudo-code program below, assume an array called `arr_in` has 1,000 cells that can hold integer values and initially contains all zeroes. Assume `print` prints the value of its argument in the same line separated by a single horizontal space when called repeatedly:

```
k = 0
n = 0
t = 0
do (while (k neq -1) AND (n < 1000))
    enter{k}
    array_in(n) = k
    n = n + 1
```

```

do (until n eq 0)
  x = sqr(array_in(n))
  n = n - 1
  print x
define sqr(x)
  a = x * x
  return a

```

Assuming that the user enters the following sequence of numbers one by one when prompted: 2, 4, 5, 7, 9, -1, the output of the program will be:

- a. 2 0 0 0 0
- b. 4 16 25 49 81
- c. 4 16 25 49
- d. 4 16 25 49 81 1
- e. 1 81 49 25 16 4

The answer for question 20 is option e. Question 20 maps to the functions focus area. Table 21 lists the knowledge concepts that are being tested for in question 20.

Table 21: A matrix of programming and math skills tested in Question 20

Variables or identifier usage	Relational operator: < (less than)
Data types: Integers (unsigned and signed)	Equality operator: eq
Data structures: Arrays	Inequality operator: neq
Assignment operator: = assigning value	Boolean operator: AND (logical conjunction)
Assignment operator: = assigning expression	Loop statement: DO (UNTIL ...)
Associativity: Left to right	Loop statement: DO (WHILE ...)
ASCII symbol for multiplication: *	Loop statement: Counter controlled iteration
Negation Operator: -	Loop statement: Sentinel controlled iteration
Addition Operator: +	Arrays: indexing with 1 dimension
Subtraction Operator: -	Functions: Defining
Multiplication Operator: *	Functions: Single argument
Screen output / display: print	Functions: Returning integer
Math Skills: addition of 2-digit unsigned integers	Keyboard data entry: enter{k}

Math Skills: multiplication of 1-digit signed integers	Math Skills: subtraction of 1-digit signed integers
Associativity: Right to Left	

Using automatic item generation (AIG) for question 20 could be accomplished by randomizing the values that k and n are being compared to, as well as the binary operators used in the functions. This question in particular could have an infinite number of variations as the value of k could be anything as long as it is the last number in the user input part of the question.

Distractors using Automatic Item Generation could be made for this question by reordering the correct answer or numbers similar. The correct answer is the inverted array with each index squared. Good distractors would be the original array, 2, 4, 5, 7, 9, -1, the original array inverted, -1, 9, 7, 5, 4, 2, the original array squared but not inverted, 4, 16, 25, 49, 81, 1, and then the correct answer, 1, 81, 49, 25, 16, 4.

The testing category of this question is function calls, while also testing Boolean operators and comparison operations. The while loop definitions are used in previous question, which a student may need to go back to for reference. As a take-away seen from other questions as well, the pseudocode definitions should be accessible on every question where they are relevant or not. This question combines a lot of testing concepts which could be useful to see if an incoming student has an understanding how all of these concepts can work together. If the intention of this question is to just test function call functionality, then it may be too complicated.

5.3 Testing Concepts On Sample Exam Summary

Across the four categories of questions, each question targeted specific skills, both programming and math. Table 22 is a summary of all the programming skills being tested for in the twenty-question sample exam and Table 23 lists the mathematical concepts that are being tested.

There was significant overlap in the programming concepts across questions, with only a handful of questions testing unique concepts that no other question tested. The math skills on the other hand had several questions that had unique mathematical concepts tested. There are a few conclusions that can be drawn from these facts. The first is that the test itself is testing a very limited scope of programming concepts, ones so universal that including one often requires others. On the other hand, it suggests that there are redundant

questions that could be removed or reworked to target different concepts that are otherwise not tested.

Table 22: A summary of the programming concepts that have been tested in the example exam and the corresponding list of questions that target those concepts

Programming Concepts	Questions that Employ the Concept
Variables or identifier usage	1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
Multivariable value tracking	12, 13, 14, 15
Data types: Integers (unsigned and signed)	19, 20
Data types: Floating point Numbers	16, 17, 18
Data structures: Arrays	16, 17, 18, 20
Assignment operator: = assigning variable	1, 2, 3, 4, 5, 6, 7, 10, 12, 14, 15, 16, 17, 18, 19, 20
Assignment operator: = assigning expression	1, 2, 3, 4, 5, 6, 7, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
Associativity: Left to right	1, 2, 3, 4, 5, 6, 10, 11, 16, 17, 18, 19, 20
Associativity: Right to left	1, 2, 3, 4, 5, 6, 10, 11, 16, 17, 18, 19, 20
ASCII symbol for multiplication: *	1, 2, 3, 19, 20
Negation Operator: -	2, 20
Addition Operator: +	1, 2, 6, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
Subtraction Operator: -	2, 20
Multiplication Operator: *	1, 2, 19, 20
Division Operator: /	2, 5, 16
Modulo operator: % (remainder)	5, 13, 15
Relational operator: < (less than)	9, 17, 20
Relational operator: > (greater than)	6, 9, 10, 15
Operator precedence: ()	4, 6, 9, 10

Operator precedence: + and *	3
Equality operator: <code>eq</code>	10, 11, 12, 13, 14, 15, 16, 17, 18, 20
Inequality operator: <code>neq</code>	20
Boolean operator: AND (logical conjunction)	4, 9, 20
Boolean operator: OR (logical disjunction)	4, 9
Boolean operator: NOT (logical negation)	4
Conditional statement: IF/THEN	6, 13, 18
Conditional statement: IF/THEN/ELSE IF	10
Conditional statement: IF/THEN/ELSE	15
Loop statement: DO (UNTIL ...)	11, 12, 13, 14, 15, 17, 20
Loop statement: DO (WHILE ...)	18, 20
Loop statement: Counter controlled iteration	16, 17, 18, 20
Loop statement: Sentinel controlled iteration	20
Loop statement: infinite iteration	11, 14, 15
Loop statement: nested loops	18
Arrays: indexing with 1 dimension	17, 18, 20
Arrays: indexing with 2 dimensions	18
Functions: Defining	19, 20
Functions: Single argument	20
Functions: Returning integer	19, 20
Screen output / display: <code>print</code>	1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20
Keyboard data entry: <code>enter{k}</code>	19, 20

Debugging: Off by one error	16, 17
Vocabulary: inclusive vs. exclusive	9
Vocabulary: Latin "a priori"	13

Table 23: A summary of the mathematical concepts that have been tested in the example exam and the corresponding list of questions that target those concepts

Math Skills	Questions that Employ the Concept
Addition of 2-digit unsigned integers	3, 6, 10, 17, 18
Addition of 3-digit floating point numbers	16
Addition of 3-digit and 2-digit unsigned integers	1, 19
Subtraction of 1-digit signed integers	2, 20
Number line	9
Multiplication of 1-digit signed integers	2, 3
Multiplication of 1-digit unsigned integers	19
Multiplication of 3-digit with 2-digit unsigned integers	1
Division of 1-digit signed integers	2, 20
Division of 2-digit unsigned integers with 1-digit unsigned integers	5, 20
Division of 3-digit floating point numbers	16, 17
Remainder calculation	5
Knowledge of matrices and common presentation of 2D matrices	18
Base 2 to Base 10 conversion	8

6 Replacement Exam Content

6.1 Revised general pseudocode notation instructions:

In order to eliminate the consequences of dividing the pseudocode instructions across multiple questions, the instructions will be provided once in their entirety before and will be available to students for the entire time they are taking the exam. They will have the option to open and download a .pdf file that contains all of the notation needed for any question at any point. This makes question writing more flexible for future exam maintenance because any future questions that might deviate from the original design categories can be added and their component parts can be added to the .pdf file instead of requiring the strict ordering enforced by the original exam design.

When the student has signed in and is ready to begin the exam, the first page that will display contains the following general instructions.

All questions in this exam are not written in a real programming language. This exam is designed to test your understanding of programming concepts rather than your knowledge of any specific programming language.

Every question is multiple-choice and there is only one (1) right answer.

These instructions are meant to provide context for the exam and to set up the basic rules of what the exam will look like. After clicking a button to proceed, the next page will be entirely the link to download the pseudocode explanation. Because of the importance of students seeing the description of the pseudocode notation, we will have students select a checkbox that states that they have looked at the document, have understood it, and are ready to begin the test before they continue. The next page displayed will be the first question.

On all subsequent question pages, the .pdf will be accessible as a small link in the upper right. This is to ensure that, despite having clicked the checkbox, if a student did happen to not read the materials or somehow lost access to them, they would be able to redownload the .pdf and gain access to the pseudocode explanation.

The new pseudocode explanation begins with a breakdown of the symbols used on the exam:

Symbols:

= assigns variables to values

+ means addition

- means subtraction or makes a number negative

** means multiplication*

/ means division and on this exam the answer of division is always a whole number

mod means the remainder after division

"eq" means "is equal to" and can be either true or false

"neq" means "not equal to" and can be either true or false

"print" means that the value of the expression of what comes after "print" is shown

"while () " means that what is under and indented keeps happening until what is in the parentheses () is false

The goal of the explicit symbols section separate from explanation of concepts is to break down the barriers that might exist between students who know different programming languages or who might want to refresh their memory on the way that mathematical symbols are used. For example, “+” could mean concatenation and * might be an unfamiliar way to show multiplication when someone is used to the “×” or “.” operator. The “/” operator is given special attention. Students with experience in looser-typed languages might assume that the “/” operator could yield answers with decimals and other students might not be familiar with or remember the way that “integer division” is different than regular division. Using the “/” operator this way also encourages students to think in a way where the modulus operation might make more sense to students who have never been exposed to it under that terminology.

From there, each major component of the testing questions is given a short description on what it is and how it works.

Loops:

A loop repeats the indented lines found under it. All loops on this exam begin with the word while. Next to while is something in parentheses that is either true or false. The loop only repeats the lines under it when the statement in the parentheses is true and does not when the statement is false.

All loops in our exam use a while-loop construction. For-loops and for-each loops are fundamentally while loops and the do-while loop does not exist in several programming languages. There was some initial debate about the use of brackets in the pseudocode. While for some, the use of brackets might help contain the text in a more readable way, for students exposed to languages like Python which feature no brackets, this could interfere with their understanding. The compromise was the use of indentation to clearly delineate when the instructions inside the loop end and begin. Additionally, while the premise of how the while loop works is included in the symbols section, by more explicitly describing the conditions when it runs, any further confusion about when the loop stops can then be boiled down to a misinterpretation of the effects of the statements inside the loop rather than a misunderstanding of how the construction works.

This construction of the loop is one of the greatest points of difference between the pseudocode as created for the sample exam and our pseudocode. The original pseudocode had two types of loops: the `DO (UNTIL ...)` and the `DO (WHILE ...)`. Both constructions relied on conditional statements, but the differences between the two loops were unclear and potentially confusing to students who have had exposure to the `DO (WHILE ...)` loops. Traditional `DO (WHILE ...)` loops execute the block of code at least once before checking the condition, in which case it is the same conditional logic that exists for a traditional `WHILE` loop. Even the instructions themselves point to the redundancy of their constructions. There are examples that show there are simply two ways to write a loop that stops when the counter reaches a certain value. Instead of creating a confusing division between the `DO (UNTIL ...)` and the `DO... (WHILE)` loops that execute the same way, we opted for the single and more familiar `WHILE` loop.

There was a brief consideration of the use of a `FOR` loop as well as a `WHILE` loop, but because the `FOR` loop has more complicated syntax and because the underlying loop can also be written as a `WHILE` loop, the `FOR` loop does not appear in the content of the new exam or in our instructions. However, because the `FOR` loop is an extremely common loop and is often introduced at the same time as a `WHILE` loop in languages that offer both constructions, it may be decided at a later point to include this construction as a worthwhile concept to be tested, but these future decisions about the inclusion of the `FOR` loop or even the `FOR-EACH` loop construction can be determined later.

Arrays:

Arrays store items in cells and each cell is labeled by an index, which is the cell's number in the array. The first index of an array is 0, the second is 1, and so on. In this exam, an array of numbers named `numArray` will be shown like this `[num1, num2, num3]`. `numArray[index]` is the item in that cell, so `numArray[0]` is the same as `num1`.

Describing arrays proved to be a difficult challenge. Arrays are a very specific data structure. The word “array” is used in many languages in the same fundamental way, so any deviation to simplify the array structure while also adhering to the common usage of the array would confuse more students than it would help. If we had wanted to make a simpler list or group object, then a construction could have been made but the name would have had to be different. Using a simple name like “list” or “set” would have mathematical and/or grammatical implications, too; however, because the array structure is so unique, it requires some explanation for students who might not be exposed to a data structure in that particular way, but can work with an array given the basics of the terminology.

The decision to include the specific phrasing of “numArray[0] is the same as num1” was made to help reinforce the array zero indexing as well as to communicate in more concrete terms how to handle the data in an array. By explicitly drawing the equivalence between the contents of the data structure with values that are able to be manipulated, the barrier is further lowered between understanding the exact nature of the array and being used to the concept meaningfully.

Functions:

Functions are blocks of code that do some task and have a name so that they can be used again.

define functionName() means that a new function named functionName does what is underneath and indented. When it is done with the task in the indented lines, it stops and the other code can continue.

The parentheses next to functionName can be empty or have variable names inside them. If there are variable names, functionName can do things to the variables listed in the parentheses.

Functions can return variables back to the place in the code where the function was called. If it does, "return" means that the value of whatever comes after "return" is given back. Functions do not have to return anything.

Functions required the most in-depth explanation because of how specific they are to programming. The exam questions include functions with and without parameters and functions with and without return values. The vocabulary used in computer science to describe the idea of passing values or references to subroutines varies but is one of the first deviations from traditional mathematical thinking that a student may encounter when learning computer science. As such, functions need to be described in a way that describes what they do rather than what they are.

Pseudocode functions can be written in a large variety of ways, depending on the types of functions that exist in a language and the restrictions that may be placed upon them by the syntax. As discussed in the symbols section, the functions description includes that the definition of what the function does is in the indented block underneath. The instructions go further to add that the function executes and then returns back to the calling program, although it is not explicitly called as such.

The next two paragraphs outline parameters and return values. To expand the scope of the original set of questions, our most complex question includes two different functions, one without parameters or a return value, and one that does. Having both function in the same

question forces a student to evaluate the differences between their behavior and the expected output of the program after. It also raises the overall complexity of the program that a student is being exposed to. Bigger programs with multiple interacting function calls will better prepare a student for actual programs they will encounter or write for their introductory courses.

Additional concepts in the domain of functions can also be included in later questions without having to change the instructions. Nested function calls, for instance, are not precluded by the instructions. The original question set included a question that involved user input, a concept that could easily be return to without changing the directions of functions but only by augmenting the Symbols table to include representation for user input.

6.1.1 Instruction language

When approaching rewriting the instructions and explanation of the pseudocode, we paid careful attention to the language that was used to explain these concepts. Despite having simple concepts, ensuring that they are explained simply and accessibly is a challenge. Students come to University of Central Florida from a wide variety of backgrounds, including a wide range of English comprehension skills, whether from lack of access to instructional materials over the course, not using English as their first language, or other reasons. As such, it was important that our instructions be accessible to students who have the capacity to program or who have programmed before in real programming languages, regardless of English skill. We used three different tools to help ensure that the language that we used was as readable as possible.

One tool used to help write the instructions was the “Up-Goer Five” Text Editor created by Theo Sanderson [10]. The text editor is an online text editor inspired by a panel of the comic strip *xkcd* by Randell Munroe that describes the components of the Saturn V rocket using only the thousand most used words in English. This was the most restrictive tool. The effectiveness was undercut by the technical words that were required such as “programming.” The next tool we used was the laxer Hemingway Editor [11] that provided more information about reading time, use of adverbs and passive voice, and distinguished between “hard to read” sentences from “very hard to read” sentences. This helped clean up the instructions further, but there were still concerns about how best to approach communicating technical vocabulary without triggering concern from these rudimentary text editors.

The final tool that we used was Readable’s text readability scoring editor available on their website [12]. Tables 24 and 25 show the different results gathered from the browser text editor. The text editor runs the text through several different readability tests. In the table, we included the Flesch-Kincaid scale and the Automated Reading Index scale. Our revised instructions scored a 6.1 in the Automated Readability Index, indicating that people with an approximately 6th grade reading level should be able to understand the text. We primarily used the Automated Readability Index instead of another Readability index or scale because the Automated Readability Index is better for technical writing, even more so than the widely used Flesch-Kincaid scale, because its formula primarily concerns itself with the numbers of characters in a word rather than the number of syllables [13]. This stems from its history of being used as a readability index that could be tabulated per keystroke on a typewriter. Because it is a reading scale more geared toward technical writing, we believed it was a more accurate measure of the reading difficulty of our work and provided more valuable information about the final version of our instructions. With a more robust scoring system other than the online text editors which could only suggest removing the important technical vocabulary being explained, we were able to make more significant improvements.

Table 24: A comparison between the example exam’s instruction set and the revised instructions in word count, sentence count, and percentage of the passages that contain long sentences that may be confusing or take a long time to read.

Instruction Set	Word Count	Sentence Count	% Long Sentences
Original	563	37	95%
Revised	384	30	53%

Table 25: A comparison between the example exam’s instruction set and revised instructions in readability scores and in sentiment.

Instruction Set	Automated Reading Index	Flesch-Kincaid Grade Level	Measured Sentiment
Original	7.8	8.8	Neutral (Slightly Negative)
Revised	6.1	6.9	Neutral

For comparison, the original instructions across the full exam earned an Automated Readability Index score of 7.8. The greatest source of issues was its count of sentences with a great number of syllables, with 28 total sentences having greater than 20 syllables. Our instructions, in contrast, have only 16 sentences with those same problems. In such short passages, these differences make a significant difference.

Another difference that was not initially detected was the sentiment of the directions. The Readable text analysis tool suggests that the original had a neutral sentiment that trended

slightly negative. The analysis tool was detecting the amount of negation used to describe some of the algorithms used for loops and measuring it as negative sentiment. Thus, the sentiment metric helped us correct the way that we wrote about loops to ensure that we were writing primarily about under what conditions the loop would run as opposed to when it would not run.

These syntactical differences directly affect reading time. The revised instructions have an estimated reading time of 1:42 and the others have 2:30. Because our design right now is to have a student consult the sheet at least once before beginning the exam, or at least at the very start of it, having a short reading time gives students more time to complete their exam should the final design of the exam have a time limit. Additionally, because our design includes an option to look at the .pdf file with the instructions at any point in the exam, the file can be consulted in the middle of an exam. The shorter it is and the easier it is to read the entire thing, the faster it is for a student to access the information they might be searching for directly.

6.2 Question Design

One important question to answer is how to structure the questions. At the least, a multiple-choice question consists of three parts: a question stem, an optional figure, and a list of answers to choose from. There are other variations of questions that could potentially be used as well. These include fill-in the blank, multiple matching, True/False, and short answer questions.

With fill-in the blank questions, text input given from the student would be checked against the correct answer stored in the database. This presents a number of problems such as how to deal with typographical errors or even the risk of security vulnerabilities that arise when taking user input. Matching questions let you click and drag answers to their correct counterparts. While this would be an interesting feature to implement, it is questionable how often this type of question would be used. True/False questions are a derivative of multiple-choice questions by simply only having two choices instead of the normal five. Short answer responses have been ruled out completely as one of the main goals of this project is to streamline the grading process of this exam. If we did add short response questions, additional resources from the Computer Science department would be needed to hand grade each question. One of our sponsors, Professor Guha, has already expressed that only multiple-choice questions should be allowed for this exam due to the shortage of resources from the department. We have also decided to include the option to add graphics to a question so that question creators have greater flexibility when writing questions.

Multiple choice questions are easy to grade automatically and while it still takes time to write thoughtful and well-developed questions that are testing the concepts they are supposed to, multiple choice questions are familiar to write. The difficulty comes in creating good distractors from the correct answer, which is a problem that does not exist for short answer questions, but may also allow question writers to target specific concepts more effectively than a short answer question which could yield answers that are “mostly” or “partly” correct because a student answers the question in a way that is correct under different interpretations or even specific languages that a student may be familiar with already.

Along with the structure of the questions comes the actual content. A problem that comes with designing programming questions is how to read the syntax of the pseudocode. One way to alleviate this issue is to allow the student to choose from different versions of the question based on the language that they are most comfortable with. For example, a problem might contain an action that behaves differently in Java than it would in Python. To mitigate this problem, a versioning system could be put in place so that for each question, the pseudocode would behave as it would according to the selected language. A problem that comes as a result from this implementation is that more time must be put into question creation which takes away more resources from the department. This also has a potential issue that arises with determining the cutoff point for the most popular languages and for compatibility in the future language popularities change. Updating or removing old questions written in languages that are no longer popular would be tedious and also cost the department time and money. The solution that we have currently decided on is the development of our universal pseudocode along with the set of directions explaining how each statement and expression works.

6.3 Testing Concepts Not on Sample Exam

While the sample exam covers a significant amount of material, the development of our new content requires investigating material that is not currently covered. Without any current data, it is hard to be able to know what concepts are going to be the most important to test readiness of students. It will also be hard to determine whether concepts that we want to be tested are ones that are easy to write questions about. The easier it is to write questions, the more likely additional questions will be written. It will also be hard without data to determine whether the written questions target the skills that they are supposed to. Questions must also be determined to actually cover the specific subsection of skills that are most important to know before taking on the challenge of *Introduction to Programming with C*.

Deciding which of the concepts below to test is going to take several semesters of iteration before any strong data trends can be established. Future curricula designed for the classes involved and other relevant coursework in the B.S. of Computer Science program can also help shape future concepts, including ones that go unmentioned here. When generating new content for the exam for our first goal, this list was important for us to consider. We also considered the list of what was currently tested and in which questions different topics were being tested. After consulting these lists, we were able to create replacement questions for ones that were redundant and to add questions to the pool so that the next exam would not have to use only the same twenty questions.

Table 26: A list of concepts not currently evaluated using the sample exam asset and additional notes about each concept

Concept	Additional Notes
Variable declarations	
Data types: Character	
Data types: String	
Data structures: LIST	
Data structures: SET	
Concept of comments in programs	
Concatenation operator	Symbol is language dependent
Relational operator: \leq (less than or equal)	
Relational operator: \geq (greater than or equal)	
Relational operator: \neq or $\lt \gt$ (not equal)	
Increment operator: $++$	
Decrement operator: $--$	
Assignment operators: $+=$ $-=$ $*=$ $/=$ $\%=$	
Exponent operator: $^$ or $**$	Python has this operator, but C, Java, and JavaScript do not.
Bit operators: $\&$ $ $ $!$ \ll \gg	
Operator precedence other than $*$ with $+$	
Conditional statement: IF/THEN/ELSE	Mutual exclusive concept
Conditional statement: SWITCH	
Conditional statement: Check for even number	$(N \% 2) == 0$
Conditional statement: Check for odd number	$(N \% 2) == 1$

Loop statement: WHILE loop	The original design of the pseudocode made this construction awkward on the exam, but the revised instructions will test this concept explicitly.
Loop statement: FOR loop	
Loop statement: DO ... WHILE	As mentioned above in relevant questions, this construction does not exist in all languages and the original version of the instructions had this loop in syntax but not in the same meaning as it generally has.
Loop statement: BREAK	
Loop statement: CONTINUE	
Functions: Returning void	
Functions: Multiple arguments	
Functions: Void arguments	
Functions: Nested levels of function calls	
Functions: Global variables vs. local variables	
Functions: Passing arguments by value or reference	
Functions: Recursion	This is a relatively advanced topic that is usually covered in Computer Science 1 and not Introduction to Programming with C, but because it is introduced in many textbooks early on in chapters about functions, this is a possible subject that could be tested for.
Debugging Skills (language specific)	This is a critical skill, but because it is so language dependent, certain important C or Python debugging skills may not transfer over. Logic flaws could be tested in pseudocode such as off-by-one errors, though. This leads to algorithm logic issues, and thus this is probably a more Computer Science 1 skill.
Debugging Skills: Confusing assignment vs. equality operators	This is a critical skill, but because it is so language dependent, this may be hard to test for with pseudocode.
Formatting output	This is a critical skill, but because it is so language dependent, the pseudocode print

	function may not be able to generate good tests for evaluating the grasp of this concept.
--	---

6.4 New questions:

6.4.1 Function question that uses void and strings

The program below uses the functions `sayHello` and `getName`.

```
x = 0
nameArray = [Yu, Taylor, Raz, Jaylin]

while( x < 4 )
    name = getName(x)
    sayHello()
    x = x + 1

define sayHello()
    print Hello

define getName( index )
    return nameArray( index )
```

What is the output of this program?

- a. Hello Hello Hello Hello
- b. Yu Hello Taylor Hello Raz Hello Jaylin Hello
- c. Yu Taylor Raz Jaylin
- d. Hello Yu Hello Taylor Hello Raz Hello Jaylin
- e. Yu Taylor Raz Jaylin Hello

The answer for this new question is option a. This new question maps to the functions focus area and Table 27 lists the knowledge concepts that are being tested for, with new concepts in bold.

Table 27: A list of concepts and skills tested in the new function question with the new concepts previously untested (strings, void arguments, and returning void).

Variables or identifier usage	Relational operator: <
Data types: Integers (unsigned), strings	Data structures: Arrays

Assignment operator: = assigning value	Assignment operator: = assigning expression
Associativity: Left to right	Loop statement: while
Loop statement: Counter controlled iteration	Addition Operator: +
Arrays: indexing with 1 dimension	Functions: defining
Functions: Void arguments	Functions: Single argument
Functions: Returning string	Functions: Returning void
Screen output / display: print	Math Skills: addition of 1-digit unsigned integers

The two biggest concepts that are introduced in this question are working with void and strings. The idea of strings as a separate data type is not discussed or addressed in any way. With the way that the symbols in the instructions are described, however, assignment does not specify numbers and states only that the assignment operator assigns a variable to whatever is to the right to it. Working with a string data type without a formal definition also avoids any complications that arise from how different languages represent strings, such as character arrays or String objects. Neither “string” nor “void” are used, either, as the concepts of those words as data types is something learned over the course of learning a programming language and studying the particulars of what those mean and how they are handled.

The goal of this question was to test something that had not been addressed at all in previous questions. Working with new types of functions and data types is essentially background noise to the real problems of this question, which require understanding that a returned value is not the same as using the value and that functions require explicit arguments passed to them if they are to be used.

All of the distractors have names in them. Students who guess an answer with names are operating under the assumption that the `sayHello` function, which takes no parameters, is somehow gaining access to the result of the `getName` function and/or that the `getName` function, because it returns a value, must contribute to the output that is being displayed. For the specifics of this question, because working with strings became important, the contents of the strings had to be considered. Words traditionally used to teach programming of some fruits, such as the names of fruits, could have been used, but pairing them with a suitable void argument function was difficult. Because greetings by name, “Hello, ____!”, is an early lesson for learners of new languages and is a common interaction, the concept of what kinds of outputs might be expected is fairly accessible. The use of names also presented another different consideration besides language: gender.

Computer science has been a male dominated field [14] for the last few decades. As such, we took great care to avoid all names being traditionally male names, at least, and we decided that the best solution was to use names that were gender neutral to the best of our ability. This was encouraged by a sponsor at the beginning of our project, but was also a decision made to conjure any particular gendered assumptions, even if the question just contains a list of names without named people performing any sort of action. We also wanted to use names from multiple languages. Additional names could be pulled from a list or .csv file curated beforehand for future questions such that question writers do not need to fill in any names on their own, reducing accidental gender bias.

6.4.2 Revised question 13

Assume x and y are integers.

```
x = 0
y = 0

while (x neq 20)
    if ((x mod 2) eq 0) then y = y + x

    x = x + 1

print y
```

When the program is finished, what is shown on the screen?

- a. 0
- b. 20
- c. 72
- d. 90
- e. 110

The correct answer for this problem is d. Distractor a does not enter the loop or modify y . Distractor b prints x instead of y . Distractors c and e go through the while loop once less and once more than necessary, respectively.

This question is a revised version of the previous question 13 so the testing concepts are still the same. The loop has been restructured so that the confusing DO (UNTIL ...) is now replaced with a simple to read while loop. We have addressed this in our detailed review of each question. Previously, there was a long description explaining what x and y were initialized to. This has been removed and replaced with the first two lines, $x = 0$

and $y = 0$. By initializing the variables like this instead of in a sentence, it makes the question easier to read and less confusing. A description of how the mod operator worked was also removed since we now have that listed in our pseudocode explanation. Another small detail that has been changed is replacing the variable names `a1` and `a2` with `x` and `y`. It has been changed because many students are used to reading variables as `x` and `y` and may find it confusing to read it otherwise.

6.4.3 Operator precedence challenging

Assume `w`, `x`, `y`, and `z` are integers.

```
w = 1
x = 12
y = 6
z = 2

a = w + x / y * z

print a
```

When the program is finished, what is shown on the screen?

- a. 2
- b. 5
- c. 4.333
- d. 1.083
- e. No output is generated

The correct answer for this problem is b. Distractors a, b, and c are possible answers that can be arrived at by following the wrong orders of precedence. This is proposed as a replacement for the original question 3 that tests knowledge of operator precedence. It only uses the most common mathematical operators of addition, multiplication, and division. Many students have learned the memory mnemonic “Please Excuse My Dear Aunt Sally (PEMDAS)” to understand basic precedence rules but fail to remember that some operators are grouped at the same levels of precedence. The revised mnemonic should look closer to “PE(MD)(AS)” where the grouped operators are performed from left to right in the order observed.

The programmatic skill involved in understanding true operator precedence not only encourages the correct interpretation of logical conditions with more complex operators,

but also reinforce the left-to-right precedence that is enforced by many programming languages, including Python and C.

6.4.4 Basic recursion

Assume a, n, and x are integers.

```
n = 4
a = myFunction(n)

print a

def myFunction (x)
    if (x >= 1)
        return x * myFunction(x - 1)
    else
        return 1
```

When the program is finished, what is shown on the screen?

- a. 4
- b. 16
- c. 256
- d. 24
- e. 12

The correct answer for this problem is d. Although recursion is taught in depth in Computer Science I, it is almost always introduced with a sample program in any first course in programming when functions are introduced to the student. The student does not need the skills to be able to write recursive functions but should be able to trace through their execution with a simple understanding of functions. The factorial program is the classic first example of recursion taught in most introductory programming courses along with the Fibonacci numbers as a slightly more complex introduction to recursion.

The distractors can be calculated by students who fail to keep track of the recursive iterations correctly. Distractor a can be calculated by a student by only multiplying the initial condition and the final else condition without the intervening recursive calls. Distractors b and c rely on a student estimating that multiplying by four is a key component of what is being asked but distractor b traps students who mistake the factorial for the square and distractor c catches students who attempt to calculate four to the fourth power. Distractor e can be calculated if a student only goes one call deep into the recursion but does not continue until the real exit condition is met.

Because `myFunction` is not named in such a way that it gives away that it is basic factorial calculations, it avoids a situation where a student may be able to ignore the programming concept involved and simply perform a mathematical calculation, but rewards students who are able to decipher that the program and can recognize what mathematical calculation is being performed to avoid the tedious task of manually mentally keeping track of the numbers being multiplied together, even if students who do so correctly get the same answer. This question can thus be answered correctly by students who are able to comprehend and recognize code for its algorithm and students who are able to trace correctly through code as written, which are both valuable skills for students to develop in any introduction to programming.

6.4.5 Basic recursion

For the pseudo-program below, assume that variables `a`, `b`, `c`, `d` and `e` hold integers.

```
a = 10
b = 11
c = 10
d = 15
e = c + d

if (a neq c) then
    print a
else
    print e
if (b > a) then
    print b
else
    print a
if (c > d) then
    print c
else
    print d
```

The output of the print statement will be:

- a. 25
- b. 25 11 10
- c. 10 25 11 10 10 15
- d. 25 11 15
- e. 10 25

The answer for this new question is option d . This new question maps to the conditionals focus area and Table 28 lists the knowledge concepts that are being tested for, with new concepts in bold.

Table 28: A list of concepts and skills tested in the new function question with the new concepts previously untested (**IF/THEN/ELSE**) are bolded.

Variables or identifier usage	Addition Operator: +
Assignment operator: = assigning integer	Relational operator: > (greater than)
Assignment operator: = assigning expression	Relational operator: neq (not equal)
Associativity: Left to right	Conditional statement: IF/THEN/ELSE
Associativity: Right to left	Math Skills: addition of 2-digit unsigned integers
Operator precedence: ()	Screen output / display

After our discussion of the original question 8, our group decided that the question ultimately was poorly designed and did not fit the exam. We decided to write a new question to replace the question 8 on the sample exam. This new question primarily adds the testing basic concept of the **IF/THEN/ELSE** condition because the sample exam lacked a question that covered that particular logical flow of **IF/THEN/ELSE**; it jumps from **IF/THEN** to **IF/THEN/ELSE IF**, an arguably more advance logical flow. By adding this question, the exam will also more smoothly build on the concepts introduced earlier in the testing material.

All distractors in this question is reasonable given different approaches. Students who understand the **IF/THEN/ELSE** concept can easy pick out the right answer from distractor. Distractor a will only print right output of the first **IF/THEN/ELSE** condition. Distractor b will catch students who stop when the first condition is true, which is when $b > a$, and print both the **THEN** and the **ELSE** output, demonstrating that they do not understand the difference between the triggers for executing either the **THEN** or the **ELSE**. Distractor e will print out all out without any regard of the if/then/else condition of the first condition. Distractor c will print out all values regardless of any condition.

Testing students' understanding of logical flow is incredibly important to understand the basics of how computers operate on any level. Being able to tease out the subtle difference between the **IF/THEN/ELSE** and the **IF/THEN/ELSEIF** logical flows will not only help students learn to read code better, but be able to write better code more efficiently.

7 Project Milestones

01/23/2020 – Project request forms submitted
01/25/2020 – Senior design bootcamp
01/31/2020 – Sponsor kickoff meeting
02/07/2020 – TA check in 1
02/12/2020 – Initial design document due
02/21/2020 – Sponsor Meeting
02/28/2020 – Project status and document review meeting
03/06/2020 – Sponsor meeting
03/20/2020 – Sponsor meeting
04/03/2020 – TA check in 2 / Sponsor Meeting / new exam material distributed
04/17/2020 – Sponsor meeting
04/21/2020 – Final design document delivered (Senior Design 1)
04/27/2020 – Final examination period ends
04/28/2020 – End of semester break begins
05/11/2020 – Start of Summer-2020 semester
05/15/2020 – First day of classes for Senior Design 2
05/18/2020 – Project status meeting with coordinators for Senior Design 2
06/01/2020 – Critical design review presentation
06/22/2020 – Demonstration to Dr. Heinrich / Dr. Leinecker
07/20/2020 – Final presentation
07/31/2020 – Classes end for Summer 2020 semester

7.1 Shift in Goals

After the turn in April when our first deliverable was given: the new exam content and the in-depth guide for future question writers. Our attention now turns to the remainder of our project: the web portal.

The website must be able to offer similar or better services than what Webcourses could offer the Computer Science department faculty in terms of administering an exam. Webcourses allows instructors to design their own tests and quizzes, automatically grades multiple choice questions, and is a platform readily accessible to students who are currently enrolled within the system. It also has systems for scrambling multiple choice answers so that they do not appear in the same order from instance to instance. Tests can be timed, also. Questions can appear individually in sequence or tests can appear in their entirety.

In order to replace such a system, our website must be able to offer all of those capabilities. Additionally, because it is not Webcourses, our website must be able to facilitate elements like the NID single sign-on that an implementation through Webcourses assumes is possible.

The administrative side of our website is going to offer the greatest difference in functionality than what Webcourses offers. The administrative views of the website are going to display data pertaining to students, individual questions, and exams. There will be defined categories that are content-based. Most importantly, there will be ways of visualizing the collected data that Webcourses does not offer, including the generation of charts and graphs to show data trends. These data trends can be used not only to investigate the exam for its intended purpose of testing readiness, but can also be used to evaluate and track different student performance, compare testing concepts over time, and direct future exam content.

8 Specifications and Requirements

8.1 Hardware Summary

The CS Placement Exam website must run on the existing server used by the computer science department. The server's hostname is `eecsserver.cs.ucf.edu`. The server is currently a Dell PowerEdge R710. The Dell PowerEdge R710 server model is designed as a 2U rack form factor system. It is currently reporting 24 CPUs of type Intel® Xeon® X5670 @ 2.93 GHz based on the `/proc/cpuinfo` file contents. This does not imply 24 physical CPUs, as additional CPUs found multi-core processors and logical hyper-threading units show up as individual CPUs when using the `/proc/cpuinfo` file. The physical Intel® Xeon® X5670 processor is a 6-core, 12-thread processor. This implies that the server motherboard has two processor slots.

The `/proc/meminfo` file is reporting a total memory amount of 24 GB RAM. The file is also reporting 8 GB RAM reserved for swap space. Therefore, the server appears to contain 32 GB of physical memory installed.

The server is reporting two SCSI disk installed using the Linux `lsblk` command. The first SCSI disk `/dev/sda` has a capacity of 837.8 GB. This is used for the root file system, known as `/`, with 543 GB of that space currently free. The second SCSI disk `/dev/sdb` has a capacity of 1.8 TB. It is used for the `/export` file system and currently reports 1.2

TB of free space. However, the document root folder for our website is being placed on an externally mounted file system mounted from another server with the name `thumper`. The external file system is reporting a total size of 11 TB with 2.1 TB of free space currently. It is expected that the total disk space requirements to host the CS Placement exam are negligible, that is less than 50 MB.

8.2 Software Summary

The computer science department server is currently running Ubuntu 18.04.3 LTS (Bionic Beaver) as the operating system. The current kernel version number is listed as `4.15.0-55-generic`. Software updates are generally installed using the standard Ubuntu software repository which provides the following items relevant to this senior design project as described in Table 28.

Table 28: The software versions of important products and packages required for our website to function

Product / Package	Currently Installed Version	Latest Version
Apache 2 Web Server	2.4.29	2.4.43
MySQL Database Server	5.7.29	8.0.19
PHP scripting language	7.2.24	7.4.5
MailUtils	3.4	3.9

Actual package version numbers currently in use are subject to change in the future when Ubuntu operating system patches are applied. The latest stable production versions are generally not available using the standard Ubuntu package repositories. They must be installed by retrieving the software directly from the owner's websites.

Additional Ubuntu operating packages available in the standard repository include the following items that may be of interest: Nginx, Lighthttpd, Gunicorn3, Node.js, MariaDB, MongoDB, PostgreSQL, Python, Perl, C, BASH, Node Package Manager, and Java. These software packages could bring a lot of opportunity to the design model. Package availability is limited by the ability of the computer science IT systems staff to support those packages. Forward compatibility with later versions of packages has not been established as an important consideration.

8.3 Website Design Summary

Website security is a very critical issue since this will be hosted on a public facing internet portal. Various website security risks must be planned for and mitigated against to include the following: SQL injection flaws, cross site scripting, cross site request forgery, security misconfigurations, and unvalidated redirects or page forwards. The risks of failing to defend against these kinds of attacks are the release of information protected under FERPA, the denigration of the integrity of the exam, and personal risk to individual students and faculty members.

In order to mitigate these risks, the website must utilize a database to store exams, questions, student network identification numbers (NIDs), results, and statistical data. There must be an administrative interface that is secured and to be used exclusively by instructors, and thus must have a different interface than the one needed for students to take the exam. The administrator interface should allow for the management of exams, students, questions by University of Central Florida personnel.

The system should also provide relevant statistics information to gauge the effectiveness of the exam. The specific statistical capabilities that will be included have not been explicitly defined at this time. Possible metrics would be the time taken by an individual per question, average time taken per question, pass/failure rate per question, and pass/failure rate of student overall scores. Time taken could be used as one measure of difficulty while the pass/failure rate is another. If a question takes a long time but students who do well in the exam pass it often, then the question could be considered a strong question that requires real thought but can eventually be successfully completed. That question could serve as a marker of the concepts that students really work to understand once they have been given time and motivation to do so. Likewise, if a question takes very little time and many students pass it, then the question could be too trivial and the question can be reevaluated for difficulty and concept behind the question.

The output of this data could be in the forms of tables and various graphing techniques, such as scatter plots, bar graphs, and pie charts. Being able to translate the digital data to a print medium may also influence some of the design decisions. The data should always be able to include information about the question, the student, time taken, and pass/failure rate. Additional metrics can be decided as time goes on and based on the input of the administrators who benefit from them.

The University of Central Florida has specific web styles, colors, logos and themes that must be followed. There is no set web template that must be adhered to, but the style must remain consistent and maintain a level of professionalism worthy of a website attached to an institution of higher learning.

Changes required to enhance accessibility will be considered in accordance with standards set by the university. These kinds of accessibility features include high color contrast and colorblind-friendly options, scaling text, screen reader support, and keyboard-only controls. Fundamentally, the more accessible that the placement exam is, the more accessible that the Computer Science degree program is. This aligns with our goals and is a strong example of the broader impact this website can have in shaping student outlook and success in the entry-level courses.

8.4 Systems Design Summary

The website must interface with existing University of Central Florida computer systems. A system for student single sign-on must be used to validate accounts. The website must receive and process a nightly batch text files containing a listing of network identification numbers (NID) indicating which students are eligible to take the exam. The website must produce a text file output of the results suitable to be transmitted and loaded into existing University of Central Florida systems. The formats of the inbound and outbound text files have not been specified currently.

9 Design Considerations

Certain considerations must be made throughout the development of our site that will fundamentally shape how it behaves. Some of these decisions can be added after completing the required aspects of our site that will provide basic functionality. However, there are a few design decisions that must be made prior to any development to ensure the site works as expected.

9.1 JavaScript Framework

There are many JavaScript frameworks that are used together across all different areas of web development. The React framework is currently the most popular and the most sought-after in the industry as of 2020, with job listings for developers who can program with it

reaching over 25,000 [15]. Other frameworks that are popular currently are Angular and jQuery, with an increasing interest in Vue.

React is maintained by Facebook and is open source [16]. React uses UI components that are relatively separate from one another and can be developed and debugged separately. One of the most significant differences between React and other frameworks is its Virtual Domain Object Model. Components of the UI are cached so during development only changes are rendered as opposed to the entire Domain Object Model like in other frameworks [17]. React uses a syntax extension to JavaScript, JSX, that adds readability. Angular is another framework that is popular. While not in demand as much as React is, Angular is still a dominant framework in the industry. Angular is maintained by Google rather than Facebook and uses TypeScript, which has stricter typing than JavaScript [18]. Angular also uses HTML to declare UI components.

Vue is an up-and-coming framework that is completely open source. Vue, like React, has the Virtual DOM, but because React is closer to a library than a framework, Vue offers a more focused and integrated set of plugins and companion libraries that are included and recommended [19]. Vue also, like Angular, uses HTML or HTML-like templates.

Plain JavaScript usually relies on a backbone of HTML and is usually integrated into the same file or nested in another file elsewhere [20]. A Domain Object Model is created and serves as the main skeleton of what the browser interacts with.

Our application is going to run on the University of Central Florida Computer Science Department server which has certain software installed on it currently. Node.js is installed on the server, but the Node Package Manager is not. Without the Node Package Manager, the server's Apache is not able to properly integrate and would be difficult or impossible in some cases to get UI components to behave as they should. Thus, our group decided to use plain JavaScript.

9.2 Single Page vs. Multi Page

Single page applications offer easy navigation and better mobile compatibility, but it comes at the cost of slower loading times and heavier JavaScript overhead. Multi-page applications require refreshes and reloads and can be harder to maintain cross-site changes, but they are more easily extended for future development.

Because our web application has a significant split across two different types of users, having at least two different pages made sense for our design. After the single sign-on verifies the user's authentication, then the user's type will cause the application to redirect to the appropriate portal. For students, this portal will be to either the first page in the exam process or will be a splash page that tells the user that they are not currently signed up to take the exam and, if they wish to take the exam, must contact an administrator. The administrators will be taken to a portal that allows for the addition, subtraction, updating, rearranging, etc. of questions and exams.

9.3 Browser Compatibility

The placement exam will not be taken on campus or at a testing center which brings in the possibility of different browsers being used to access our site. We want to make our system reliable and maintainable post deployment, so verifying the compatibility of our system on different browsers is essential because students may not have reliable access to a particular browser, even the most popular ones, because of how they might access the Internet. A few popular browsers to keep in mind are Chrome, Safari, Firefox, Opera, Edge, Internet Explorer, and Chromium. These browsers would also preclude mobile browsers, as students will not be able to take their exam on a smart phone or another mobile device that can connect to the Internet via a web browser.

Once we test and verify that our site work in all of these browsers, that should mitigate any technical errors post deployment for students trying to take the test. We have discussed possibly developing a browser test page to be accessible on the home screen of our application or included in the instructions. The purpose of this page would be for the student to validate the JavaScript and cookie capabilities prior to entering the exam. This test could include a simple press of a button and test file download.

9.4 Linux, Apache, MySQL, PHP (LAMP) Stack

While the Computer Science department server has all of the ingredients for the LAMP stack available to us, there were other possible design decisions that were necessary to determine if there were better alternatives.

Python Django is a web framework based in Python. Django handles the backend interaction on a more abstract and high level. It uses Object Relational Mapping (ORM) to wrap database queries into objects, structures, or variables of the programming language

being used [20]. Django still requires a database, with MySQL and PostGres being two of the most popular, also uses Linux, and often uses Gunicorn as a webserver.

Python Flask is another Python-based web framework, but it is not a replacement for the full stack [21]. It is based on the Jinja2 template engine, which would require additional installation onto the server, and lacks some of the security features that Django offers as well as the ORM system and would still require queries in order to access the database.

The decision to go with an SQL or NoSQL database also had to be considered. A traditional relational database would allow for complicated queries, but would require a stricter and more organized structure. NoSQL would allow for a greater degree of flexibility, but it would offer less validation of data. While we are working with sensitive personal student information, including information protected by FERPA, as well as information that will directly affect a student's curriculum, security is an important issue, and accidentally leaking personal information or sending out the wrong results to students would drastically affect the integrity of the exam. As such, we opted for the SQL, relational database model.

9.5 Administrator Portal

A web page for test creators, and those who would like to analyze test results, is necessary to manage the exams. We have determined what we will need to complete for the portal to be functional, as well as what would be nice to have. These include tools to create an exam, view past and current exams, and to analyze exam metrics among other various tools.

We have considered different functions the administrator portal should provide for computer science faculty. They should be able to access past and current exams and the metrics for both. We may be able to include these past results in our statistical analysis section for additional data. There will be a section to view the list of students registered to take the test so that students can be manually validated. It can also be used to see who all is signed up to take the current exam. Administrators should also be able to manually add students to the registration list. To provide a means of technical support, we have also decided to allow administrators to reset a student's attempt due to certain circumstances. An option to add an administrator to the account would also be useful in case additional faculty needs access to the exams.

More tools may be added to allow for a smoother exam creation process, but as of now, we have decided on a few core functionalities that must be implemented. When the administrator goes to view the question bank, they should have ability to edit or add

questions. Within this question creation form, administrators will be able to fill in the question stem, category, five multiple choice answers, and the corresponding correct answer. The list will be categorized by type and potentially be sortable by creation date or other analytics like pass/fail rate. We would like to include HTML or Markdown formatting within the question stem input box to provide more flexibility when creating questions. This would make the system vulnerable to certain attacks so we would need to sanitize the input before storing it. We would like to add the option to parametrize questions with certain variables that change based on how the administrator sets them to. Another consideration is to provide an additional description section for question creators to use to describe the concepts tested and other optional information. This could be used to help facilitate the generation of new content by new authors after original attributors leave. For instance, if our sample exam asset had come with additional materials behind the development of the exam asset, then writing more content in line with that original vision, or being able to redirect that vision, would have been easier. This could also be used as a resource for metadata about questions that could be transformed into meaningful metrics, such as the appearance counts of different words in the description field.

Beyond individual question editing, administrators need to be able to designate which questions create the exam for a given semester or set of students. If the pool is larger than the number of items that will exist on the test, then there needs to be some way of choosing which questions appear. Administrators should have the option to manually add questions to an exam, such as through the use of checkboxes, in order to control for specific concepts being tested or just to ensure that the question items with the greatest capacity for testing readiness are included. Having the ability to “lock” questions as questions that will appear in all future exams could be one such manual feature. A feature that allows for administrators to add questions that are not counted for a student’s score would also be a good way to field test new questions without affecting consistency across semesters, and this process would be manual because of its deliberate nature.

However, in order to reduce manual effort, there should be a capability for the administrator to be able to click a button or for the system itself to automate question to exam mapping by pulling a random sample of questions from each category in the pool. This would be the easiest solution for creating new exams with the least amount of effort, so this functionality is one of the top priorities for our system, in order to encourage administrators to want to use our system as well as just to offer some of the same capabilities that Webcourses can offer.

9.6 Instruction Presentation

Having written new instructions for students to have before they take the exam, the question about how best to present this information arises. Having a .pdf file for students to download and/or print out before the exam is the solution that we have chosen. Having instructions appear before relevant questions becomes an issue if we are using a multipage layout that assumes a single question on a page. If the questions build up in difficulty, then a student should not have to go back through earlier questions to get information about small pieces that make up harder questions. Having all of the data from the beginning is the fairest solution for students; however, because a single file can either be lost or discarded accidentally, the option to access this file would not be constrained to the very first page of the exam, but will be an option on each and every question page that exists. This way, there is no additional space taken away from the question presentation itself, but the student can still have access to the materials on the items when they might need it the most. If the option is present on every question page, too, it stresses to the student taking the exam that it is a resource that they are encouraged to use.

When designing the instructions, the entirety of the instructions ended up being text explanations of the various symbols and the way they interact in a basic logical flow, but one earlier idea was to add flow charts to the instruction set to explain visually what might be hard to capture in text. For ease of reading, this idea was not selected: having large figures would make the document unwieldy.

9.7 Statistical Analysis

An additional tool that we would like to add is the option to view statistical analysis of past and current exam results. This analysis would cover averages for exam scores, exam completion time, time per question type, and score per semester. Analysis on individual questions such as pass/fail rate and percentage of each answer choice would also provide information on how to modify current questions or how to create future ones. Our site could also interface with the university's statistics of *Introduction to Programming with C* and Foundation Exam passing rates for comparison.

This data could be analyzed and converted into different graphics using some external JavaScript libraries. Otherwise, we could simply provide the data in a table and allow administrators to save the data into a spreadsheet. Research into what technologies we can use to implement this are still ongoing as our main priority is to complete all required functionality.

9.8 Question Resources

Because the exam is being used to determine readiness for *Introduction to Programming with C*, the students upon completion of the course are important resources for data gathering, but the students themselves could be used for future content development. Students who have just taken the course can be motivated to write questions for the placement exam for internal reasons such as a sense of wanting to help others who are about take the same class they just finished or for an external reason such as a grade on an assignment. Many students are in any given section of *Introduction to Programming with C*, and therefore many questions could be gathered very quickly should there be a student-based crowdsourcing effort for questions.

There are pros and cons to this approach. One of the pros is the act of writing the questions themselves as a teaching tool. Forcing students to think about how to simplify their own knowledge and get to the kernel of what the concept they are trying to test forces students to really understand the material. Additionally, any input about how to write good questions, which could be incentivized to consider should this be the rubric for a class assignment for a grade, would force students to consider even more the content that they are tasked to write questions about. Creating distractors in particular is a challenging exercise, as a student must consider not only what the correct answer is, but the steps behind getting the right answer and analyzing where those steps could go wrong.

Another pro is simply that this would be a very good way to generate a lot of questions very quickly. Even without an incentive, if even one student a semester contributed a new question, that is more questions than would have existed otherwise. As time goes on, with each concurrent semester, the pool of questions would grow.

To facilitate student contribution, creating a student portal with access to submitting question content, potentially to its own separate table in the database, would require further front-end development and consideration. If the resource gathering was done manually, such as taking user submissions on Webcourses of a text file, then more effort would need to be taken to input those questions to the database and would require even more effort to sift through all of the student-suggested content to ensure that what was being contributed to the pool was a valid potential question for a future exam.

Thus, the biggest con of this approach is the overhead. After getting many questions, with even one section of *Introduction to Programming with C* with each student contributing one question each, this would be dozens of questions that then an instructor would have to evaluate for viability. Because these questions are not just a learning tool, too, even

students who have carefully considered the concept they are trying to write a question for and do the best job they can in writing a new question, as students they simply do not have the expertise to be able to determine fully whether their question is a good one. Expert opinion is necessary.

The development of an automated tool for sifting through these questions, developed by experts to replace manual effort performed by experts, would be significant. Even just scanning question contents for acceptable vocabulary comes at the cost of having to predetermine what words or phrases are likely to result in good or bad questions. Overall, it seems extremely unlikely that the cost of developing such a tool would outweigh the benefit of being able to get a first pass through a mountain of student-generated content that would then still require additional manual effort to go through.

10 Diagrams

10.1 Block Diagram

Our basic system relies upon the CS Department Server and the interaction with the server and the user browser, the admin browser, the UCF PeopleSoft System, and the UCF Single Sign-On system.

Figure 1, see next page, illustrates the various computer systems being used in this design and the interfaces between them.

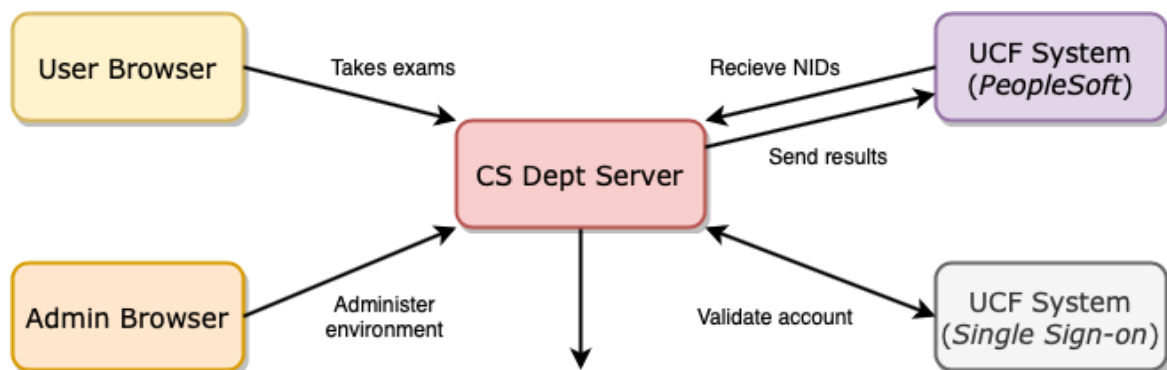


Figure 1: Interfaces Between Site Technologies

The User Browser block represents individual students taking the CS Placement exam from anywhere in the world, on their own computer or on someone else's computer. They will be using their own hardware and software to take the exam using a variety of potential different browsers such as Chrome, Firefox, Safari, Opera, Chromium, and others. The

Admin Browser block represents the browser of a validated University of Central Florida faculty or staff member account that will have privileged access to the CS Placement exam website. Privileged access will allow complete management of the system such as the ability to create, to add, to delete, and to update the detailed components of exams and questions. It will also have features to manually enter NID numbers and remove an exam attempt due to technical issues that require a retake. The privileged access would also allow view of statistical analysis such as average scores of exams, average scores for individual questions, time spent on exams and questions, along with any other statistics to be determined later. The UCF System (PeopleSoft) represents a bi-directional interface by which NIDs are received nightly and results are sent back to University of Central Florida systems. The UCF System (Single Sign-On) block represents the system used to validate username and password authentication since those items are not validated or stored locally. The CS Dept Server block represents the website hosting the CS Placement Exam and its database. The down arrow from that block references the second block diagram listed below that shows the inner details of that block.

Figure 2 below shows an initial design of the entities to be created within the CS Placement Exam environment to be hosted on the University of Central Florida computer science department server.

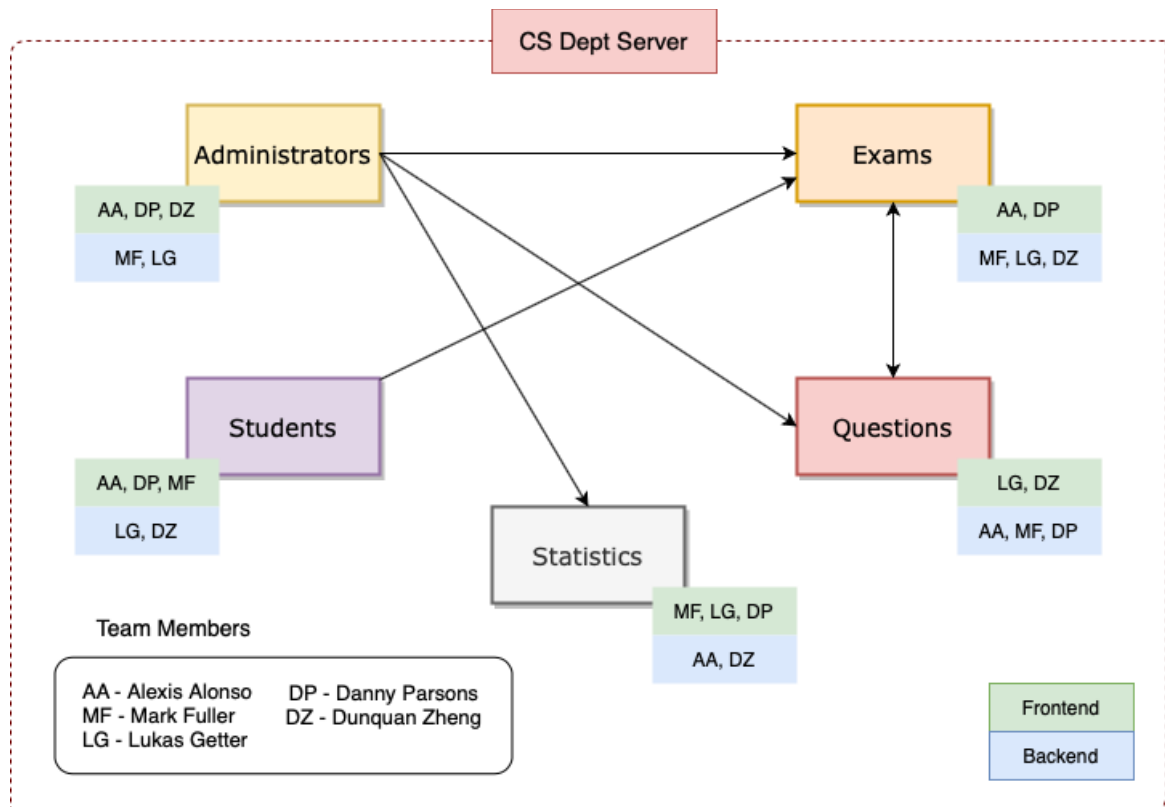


Figure 2: CS Department Server Database

These entities will correspond to a series of frontend web pages, backend application program interfaces, and to entities within the database. A more formal entity relationship diagram will be created later when the attributes for each entity have been identified.

The legend provides the initials of each team member and identifies the areas that they will work on. This preliminary assignment of task is subject to change once development starts based on the skills and rapid mastery of new concepts by the individual team members as well as with the solidification of overall design features and additional sponsor input.

10.2 Use Case Diagrams

The two following figures are the use case diagrams for two users that have access to our system: students who are registered to take the exam and valid administrators. The student's primary motivation for using our system is to physically take the placement exam. The administrator's primary motivations are to edit the exams and to gather metrics based on previous administrations of the exam.

10.2.1 Registered Student

Because the student's primary motivation for using the site is exclusively to take the exam, the student use case diagram focuses on the entry point to the exam and the potential branching points of how the student will complete their main goal of using the site. In our diagram, the student side of test system is depicted as a main field Test inside the Test System (Figure 3).

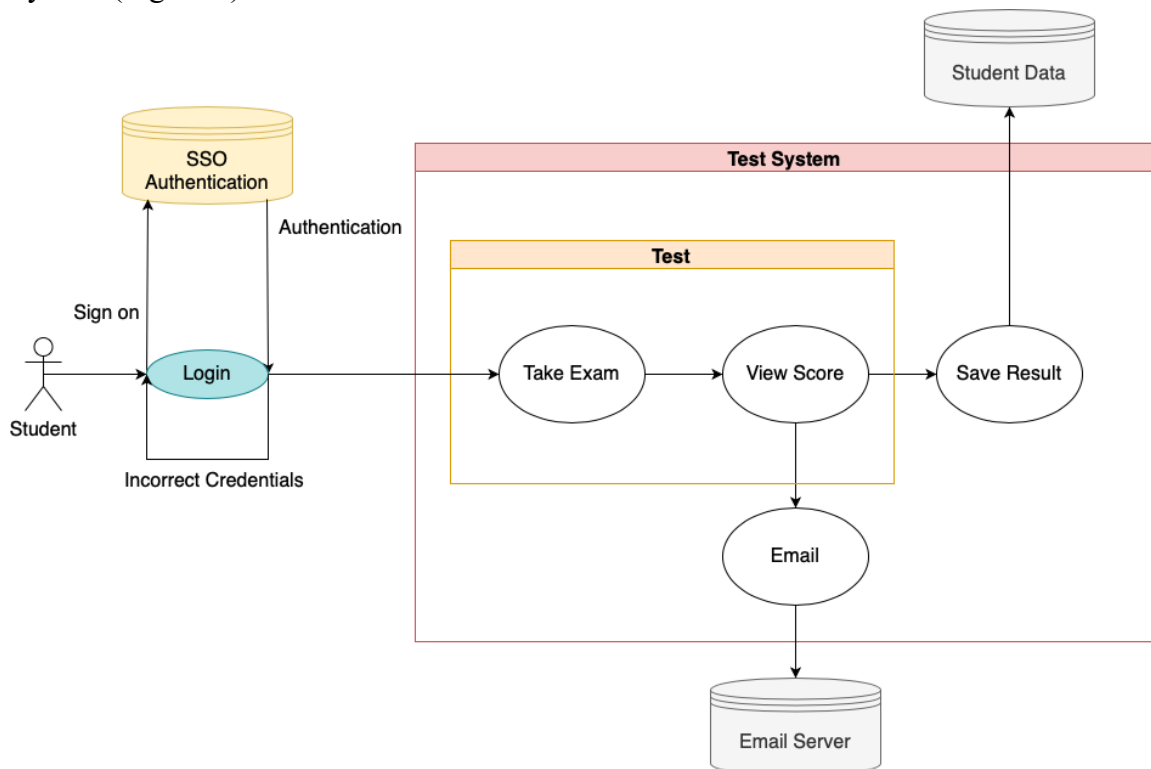


Figure 3: Registered Student Use Case Diagram

Students will be able to enter the system after entering correct credentials that are then validated by the university's single-sign on. The other requirement to being able to see the exam itself and take it is that we have accounted for them to take the exam from PeopleSoft's data drop. Once the student enters the environment, they will have an introduction/instructions page, followed by the exam questions, and then a results page. The results will only be available to our system, the student, and PeopleSoft after the student submits the exam. On the results page, the student will have the option to email their result to them upon submitting an email address. Test results will automatically be saved upon submission, so the student will have no other functionality available to them. From there, students will be able to leave the page or close the browser window and not have to interact with the system ever again.

10.2.2 Administrator

The administrative side of the test system will have two main functionalities: exam editing and exam metrics (Figure 4).

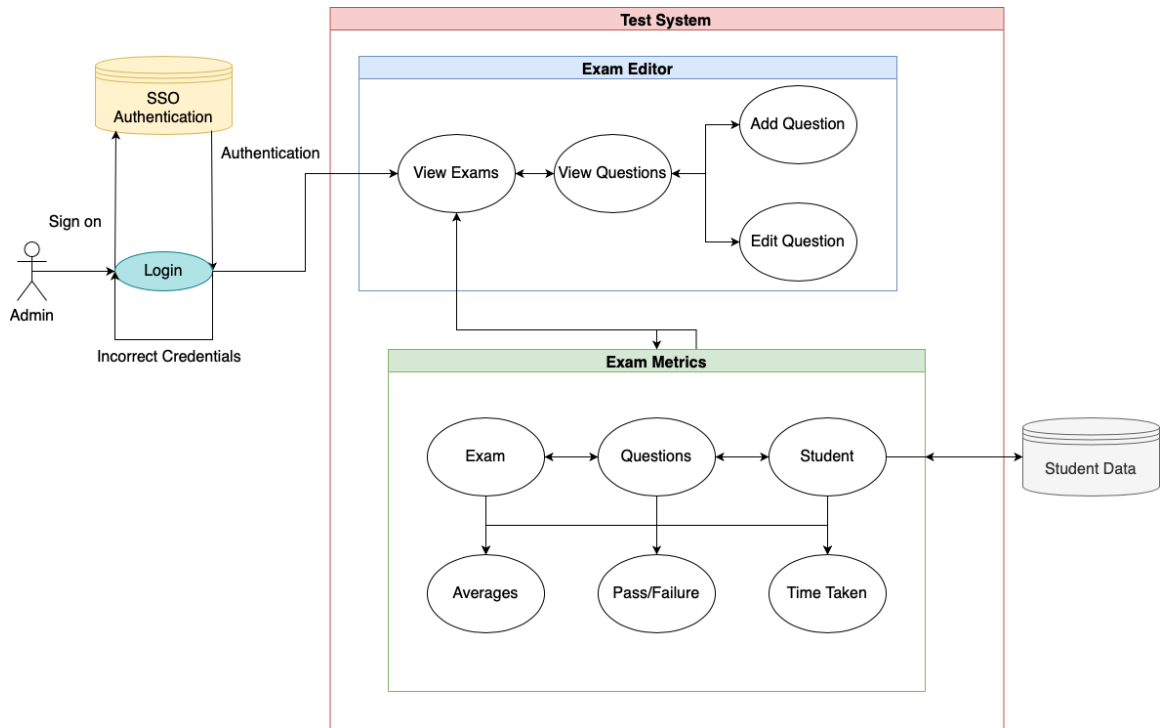


Figure 4: Administrator Use Case Diagram

At this time, it is unsure how exactly administrators will sign into our application. The diagram above assumes the administrators will also gain access through the university's single-sign on that students use with their NID. In the future we may redesign this entry to allow for additional administrator authentication methods. We want to ensure anyone who needs access to these metrics can get to them, but also maintain the security of the students' data. This is an important security feature to maintain the integrity of the test. Having stronger or additional authentication methods would also encourage the access of the web portal to a limited number of people rather than any faculty member. Because of the specialized nature of the content that this web application is providing, more than a handful of people utilizing the administrative side of our system may be unnecessary.

The exam editor part of the system would be where the administrators can manipulate the items in the exam itself. Questions can be assigned to exams, exams can be assigned per session or per semester or per student, and then there should be the ability to add questions and edit questions that are already in the pool of possible questions. Viewing exams could

also allow the user to auto populate the exam with certain types of questions or to manually select items, such as to try a particular question out or to test a new concept that may not have been addressed before.

The exam metrics side of the system would be used by the administrators to shape future exams based on the statistics of the previous ones. Metrics can be taken such that exams could be examined for averages, time taken, and pass/fail rate, and the same metrics can be taken for individual questions as well. Student averages would not apply as a student is not a population, but potentially being able to map and identify a student as part of a visual representation of several averages would be useful. These metrics can be used to guide future iterations of certain questions, of the exam itself, or of the curricula that the placement exam is supposed to test readiness for. By offering a dashboard of statistical and data visualization, administrators will be able to not only plan for each exam, but to evaluate the effectiveness of the exam as a whole semester to semester or year to year.

11 Storyboards and Wireframes

11.1 User Storyboards

11.1.1 Student View

When a student navigates to the placement exam, they will first start at the login page shown in the Figure 5 (see next page). Here, they will be prompted to enter their NID. Once submitted, the NID will be processed in the backend to determine whether or not the student is currently flagged to take the exam this semester. If the student's NID is found, they will be sent to a University of Central Florida hosted sign-on server to be authenticated. However, if the student's NID is not found, an error message will be displayed explaining that the student is not currently registered and to see an advisor for further instructions. Additionally, if the user has a recognized NID in the system, but has already taken the test, they will receive an error message as well. Once the UCF server confirms that the student has been authenticated, the student will be sent to an exam welcome page where they will be given a pseudocode overview .pdf and instructions for the exam. The pseudocode overview .pdf will describe the rules to follow when reading pseudocode in the exam questions and is detailed above. The exam instructions will give the time limit for the exam and other instructions about what to do after completion.

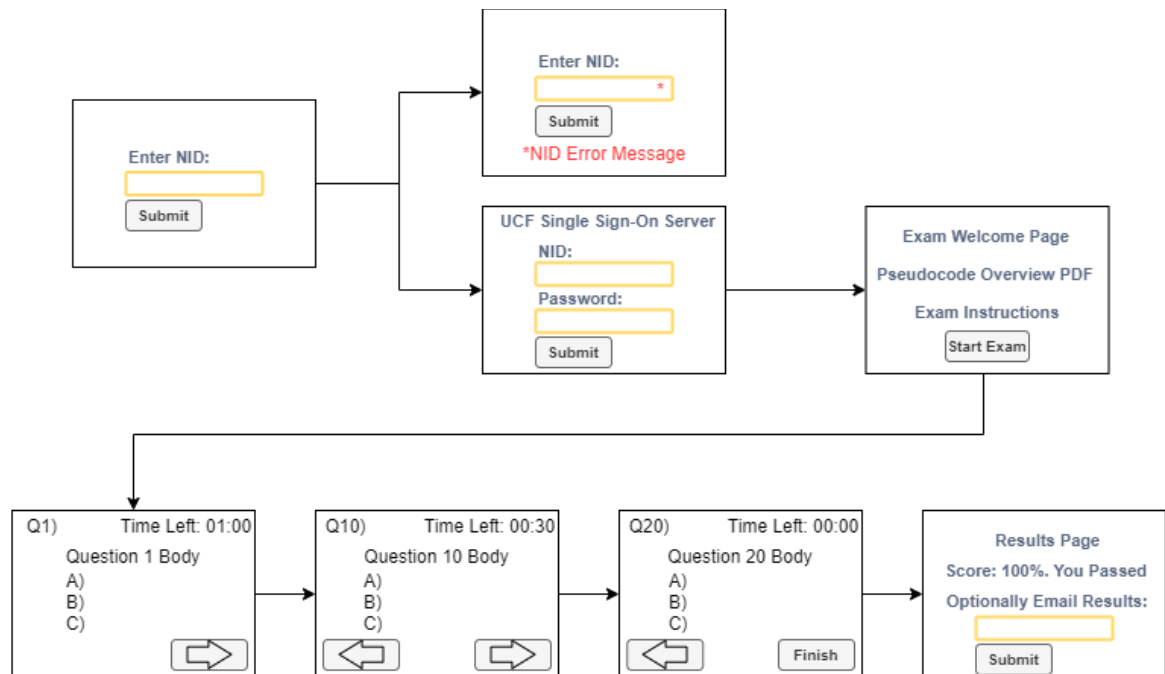


Figure 5: Student Question Taking View

After starting the exam by clicking the “Start Exam” button, the student will begin the exam and be directed to the first question. A time remaining countdown will be displayed in the top right corner of the question page and will begin at one hour. The first question page will only have a forward navigation button that will lead the student into the next question. From the second question until the second to last question, a forward and back button will be displayed to navigate in either direction. At the final question page, a back button and a finish button will be shown instead. The finish button will submit the exam to be graded and send the student to a results page. On this page the student’s score and pass or fail status will be displayed. Optionally, the student may enter an email address to be forwarded the results of their exam for future use. From there, either after a time out or after the user submits their email, either a popup will alert the user that the window is closing or the user will be taken to a screen that directs them to close the window or tab. That should complete the entirety of the student user’s experience with the application.

11.2 User Wireframes

Users will be automatically redirected to the University of Central Florida’s single sign-on, as depicted in Figure 6 below. They will enter their network credentials and be taken to one of three different pages depending on where their identification is stored in our

system. They will either be a student who is not registered to take an exam, a student who is registered to take the exam, or a faculty member that we have record of.

UNIVERSITY OF CENTRAL FLORIDA

UCF SIGN IN

Search UCF

UCF Federated Identity

Account

Username: NID or DTC-Atlas ID

Password

Password

Sign On

By signing on, you agree to the terms of the [UCF Policies & Procedures](#).

my.ucf.edu

You have asked to login to my.ucf.edu

[Look up your UCF NID username.](#)

[Reset your account password.](#)

[Trouble Signing On?](#)

UCF UNIVERSITY OF CENTRAL FLORIDA

Figure 6: University of Central Florida Single sign-on Page

11.2.1 Nonregistered Student

The students who return from single sign-on with an account that we have no record of in our database will be brought to a splash page similar to the one depicted in Figure 7. There will be some simple verbiage along these lines. No additional functionality will be available on this screen.

The NID entered has not been flagged to take this exam.
Please try again later or contact your advisor.

Figure 7: A splash screen with an error message for nonregistered students

11.2.2 Registered Student Who Has Taken the Exam

It is possible that either by mistake or as an attempt to get a different score, a student user may attempt to log onto the site again to try and take the exam again. This could be the case if a student has forgotten their scores or if a student would like to try for a different score should they get an undesirable result. In this case, a user will already have their scores located in the database, and they will be taken to a screen similar to the nonregistered user as shown in Figure 8.

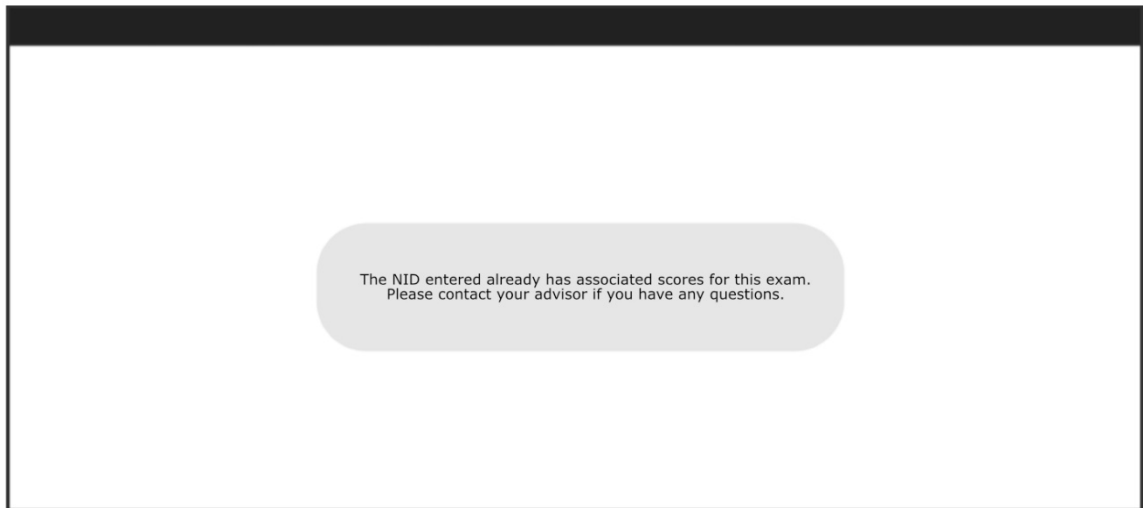


Figure 8: A splash screen with an error message for students returning to the page after they have completed their exam

11.2.3 Registered Student Who Will Take the Exam

The students who return from single sign-on with an account that we have a record of in our database will be brought to a page similar to the one depicted in Figure 9 (see next page). This page will welcome the student and show the instructions for the test in a text box. The instructions will include the grading scheme, time constraints, and instruction to download the pseudocode .pdf. When they click the .pdf icon, the pseudocode attachment will download to their machines. Once they click the Begin Exam button, they will be brought to the next page where the exam will start with question 1.

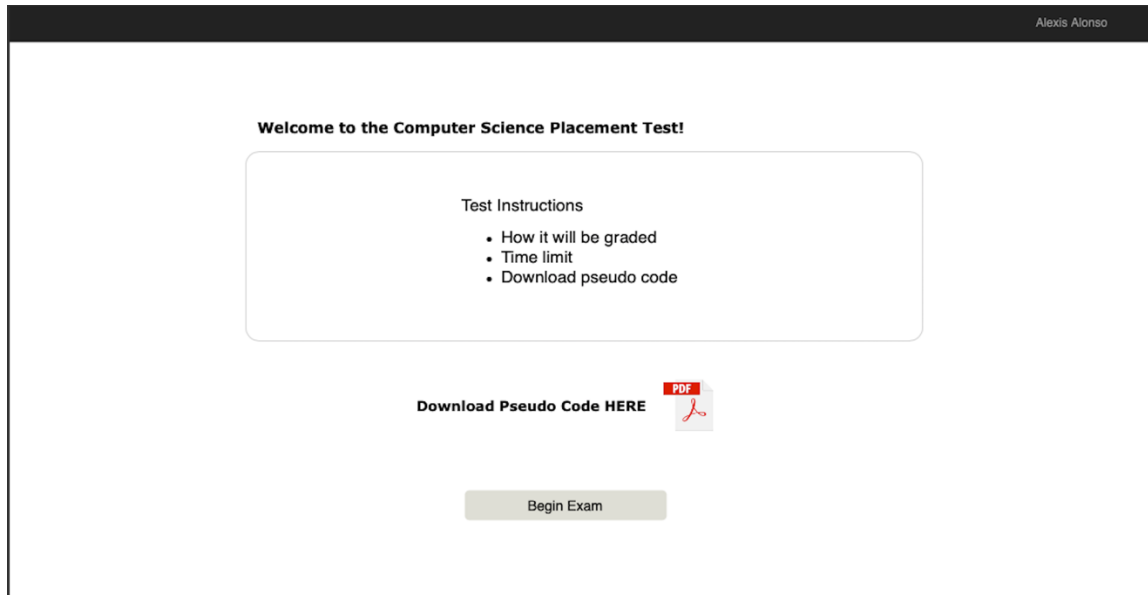



Figure 9: The first page of the exam that features the test instructions, a link to the pseudocode explanation, and a button to start the exam

Once the student clicks the Begin Exam button, they will be brought to the first question of their exam with a layout similar to that of Figure 10 (see next page). The question pages will all have the same format and layout. They will show the question in a text box as well as all five answer choices next to radio buttons. The overall exam timer will be visible on every question page as well as the .pdf icon again to download the pseudocode explanations in the case that a student skipped this on the instruction page. Every page will have a Back and Next button except for the first and last question. Question 1 will only have a next button and the last question will have a Back and Submit button. Once the student completes every question and clicks the Submit button, their answers will be graded immediately, and they will be brought to the results page.

Alexis Alonso

00 : 00 : 19



Question 1

For the pseudo-program below, assume that variables A, B, C, and D hold integer values.

```
A = 120
B = A + 100
C = B + 30
D = C * 10
print D
```

The output of the print statement will be:

☐

2500

☒

0

☐

100

☐

120

☐

250

Back

Next

Figure 10: A sample question page of the exam with a box for a question, space below for multiple choice answers with radio buttons, a link to the pseudocode instructions, a timer, and a forward and back button to navigate through questions.

Once the student submits their exam, they will be brought to the page depicted in Figure 11. This exam is not deciding their coursework, but rather recommending it. For that reason, the verbiage on this page will have to represent a recommendation and not display a new course requirement. Students will be able to screenshot this page for their records or we will be providing an email option for them. Rather than receiving all the student's email addresses from PeopleSoft, we will only save the ones that students give us. The student can submit an email address for us to send official result documentation, or they can close this page. At this point the results and total time. Are stored in our database and ready to be sent back to PeopleSoft.

Alexis Alonso

Congratulations on completing the exam!

Based on your test results, we recommend that you take COP____, Intro to C or Python. If you would like these results emailed to you, please provide your email address in the box below.

Email Address

Figure 11: The page a student would see after the completed exam with an option to email their scores to an email of their choosing.

11.3 Administrator Portal

If the UCFID returned from single-sign-on is flagged as an admin user account in our database, then they will be redirected to the administrator portal upon logging in. From this interface, the user will be able to perform the CRUD (create, read, update, and delete) functions on almost all of our database entities.

This is the current idea for the admin portal landing page, see Figure 12. The menu across the top of the page will provide easy navigation between user, question, and exam data records. The Users tab will be the first entity the admin can view. Each of the yellow tabs represents a different filter of the user data. The first user group consists of the students registered to take the exam. This data will be sent to our system from PeopleSoft. Basic student information will be visible here. The functions available on this screen are create and search. If a student is not included in the PeopleSoft transfer but needs to take the exam, an admin will be able to add their record to this table via the New button near the top right. The search function is also available here to verify that any particular student is pending in our system to take the exam. Deleting and editing will not be available for this entity to preserve data authenticity from PeopleSoft.

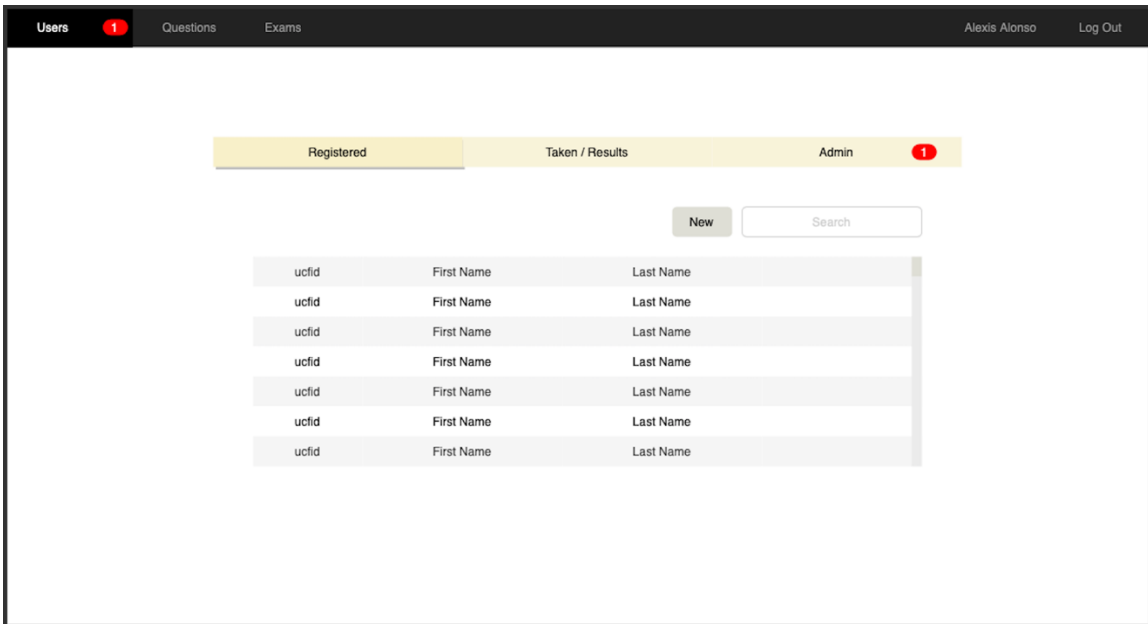


Figure 12: The Registered students tabs of the Users page on the admin portal

The second user group visible under the Users tab will be a record of the students who have already taken the exam and their results, see Figure 13 on the next page. We will be sending this information back to PeopleSoft to update the university's data. The only function available on this screen is searching. To preserve historical data, we do not foresee the need to edit or delete any of this information. Scores should not be manually changed or added into our system. Some fields on this table could include the students name, id number, score, date taken, and/or exam name.

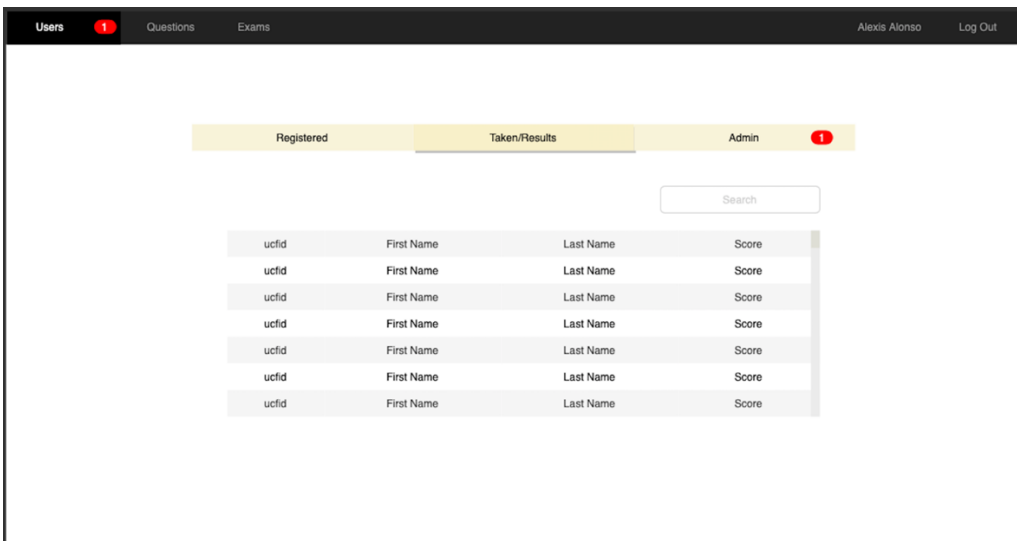


Figure 13: The Taken/Results tab of the Users page on the admin portal

The third and final user group visible under the Users tab will be the system administrator accounts. We have not finalized the process of adding or maintaining admin users. However, at the time this wireframe is being created, we are predicting an approval process for new faculty to be monitored by current admins. In the user interface shown in Figure 14, we implemented an approve/decline functionality for new admin trying to create an account. We have red notification bubbles on each menu bar to signify and bring attention to the pending admin account seeking approval. Each user will have the ability to change their own user information, but they will not be able to change other users' information. The creating and searching functions are available on this page as well in the upper right-hand corner.

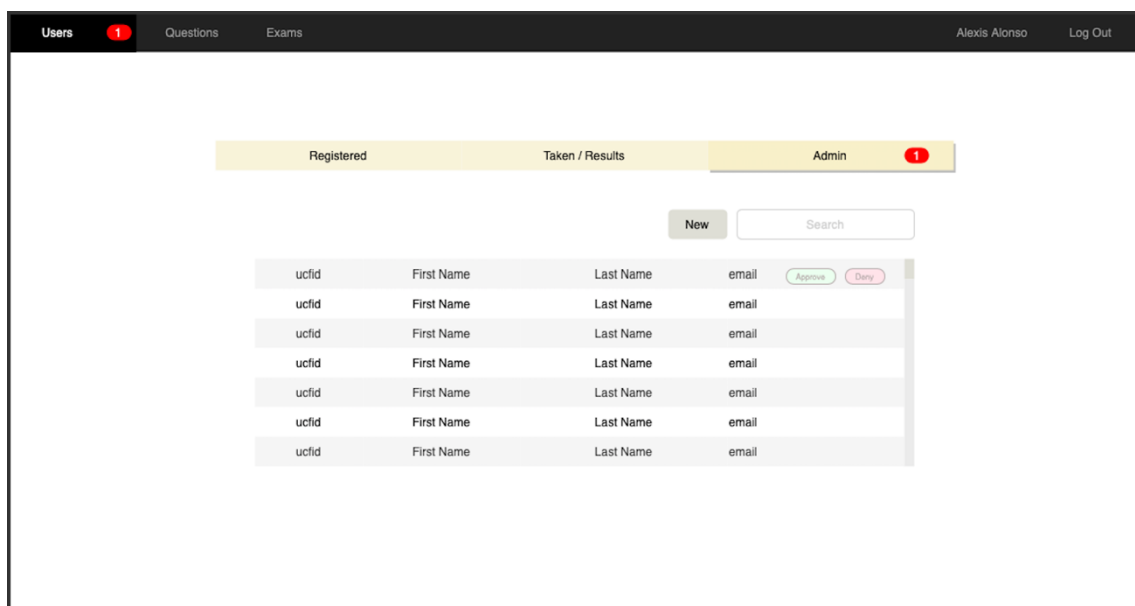


Figure 14: The Admin tab of the Users page on the admin portal

The second category of the main menu is the Questions tab. The complete question bank will be visible and editable from this screen as seen in Figure 15 (see next page). Each question falls under one of the testing concept categories pictured below. The question bank will be organized and condensed by category: conditional, functions, loops, or sequential. When clicked on, the category drop downs will expand with all the questions currently existing in that group. We want the ability to expand and collapse the question bank to reduce scrolling space and page clutter. Especially once additional questions start getting added, this list of total questions could become extensive. The functionality from this view is searching existing questions and creating new ones. Both functions are available in the upper right-hand side of the screen.

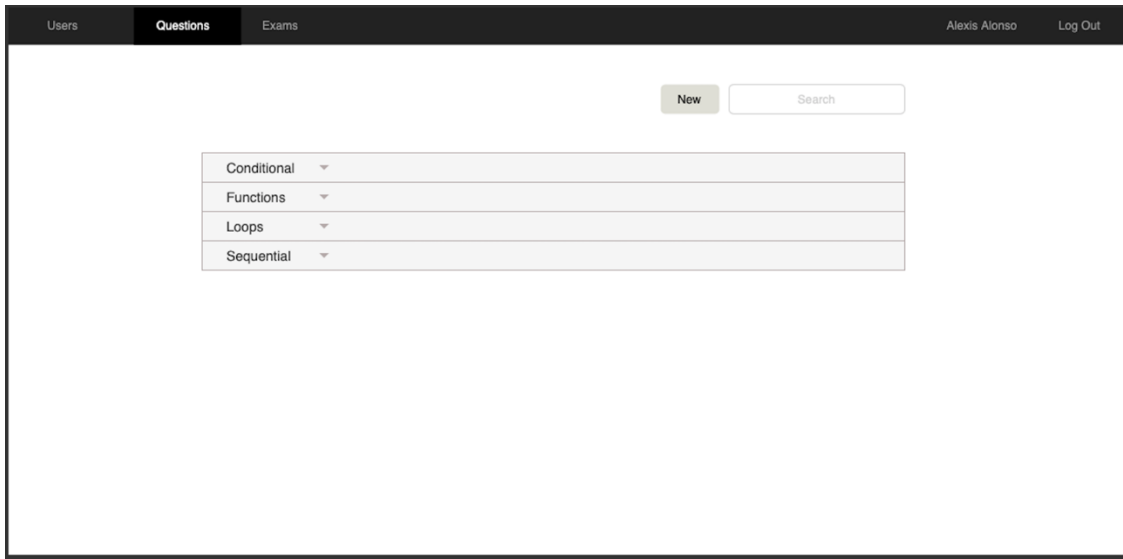


Figure 15: Four category bars that open up to reveal question banks when clicked on

From the previous page, an admin user can create a new question by clicking on the New button, which takes them to the page shown in Figure 16. This is the basic user interface for adding a new question. The database fields to be filled-in will include: the question stem, correct answer, other answer choices, and the testing concept category. The category drop-down will provide the active options we have stored in our database: conditional, functions, loops, or sequential. Once all the fields are filled in, the user can select the save button to add it to the bank or hit cancel at any point to leave the create function.

Figure 16: The new question page with text boxes for the question stem, correct answer, distractor multiple choice options, a category menu, and save and cancel buttons

From the Questions tab of the main menu, the admin will be able to expand the category drop downs as shown in Figure 17. When one or more of the categories are expanded, the question records will become visible and scrollable. On this page the user will be able to perform all of the CRUD functions. As mentioned earlier, new questions can be created from this page. A user can also search questions or delete them. Only basic information about each question will be visible from this list view. However, the user can click the View/Edit button on the question record to be brought to a page with all of that particular question's information.

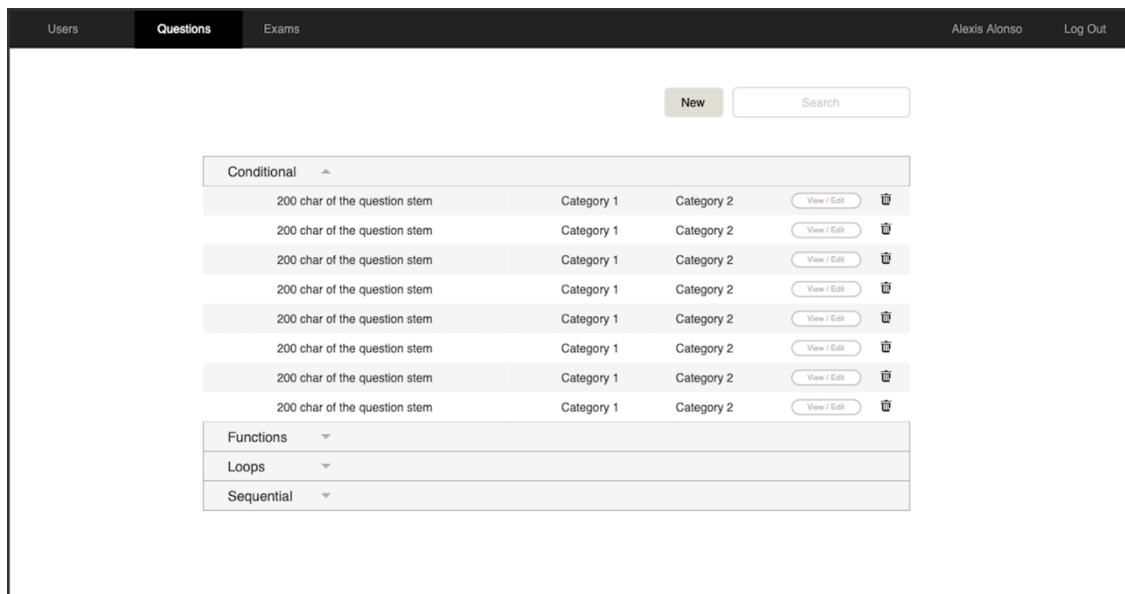


Figure 17: When a category is opened, questions of that type will be displayed along with their additional category listings, a button to view/edit a question, and a trash can icon to delete a question

In Figure 18 on the next page, you can see the question View/Edit interface. It is identical to the create page but each field is filled in with the data we have saved in our database. If the user is just viewing the question and associated information, they can click the Cancel button on the lower right-hand side. If they are trying to make any changes to the question stem or answer choices, they can modify the appropriate fields and then click the Save button on the lower right-hand side. Once either of those buttons are clicked, the user will be redirected back to the question bank page. If information were updated, that will then be visible on the table record as well.

Users Questions Exams Alexis Alonso Log Out

Question Stem:

For the pseudo-program below, assume that variables A, B, C, and D hold integer values.

```
A = 120
B = A + 100
C = B + 30
D = C * 10
print D
```

The output of the print statement will be:

Correct Answer:

Category:

Other Choices:

Figure 18: An example question page with the content of a question filled in, including the question stem, correct answer, distractors, and category.

12 Backend Development

12.1 Single Sign-On

Our site will utilize Shibboleth, the single sign-on service currently being used by the university. This tool will allow us to authenticate users based on their UCFID to categorize them into students, administrators, or others. There are six steps to the Shibboleth single sign-on service:

Step 1: User Accesses the Resource

When a user attempts to access the resource, a resource monitor will check if the user currently has a valid session. If the user does not have a valid session, then they will be forwarded to the service provider to begin the single sign-on process.

Step 2: Service Provider Issues Authentication Request

The service provider will forward an authentication request, along with the user, to an identity provider for user authentication..

Step 3: User Authenticated at Identity Provider

At the identity provider, the user will be checked for an existing session. If they do, they continue to the next step. Otherwise, the identity provider will authenticate the user by prompting for a username and password. Once the username and password are authenticated, the user will proceed to the next step.

Step 4: Identity Provider Issues Authentication Response

After the user has been identified by the identity provider, an authentication response and the user will be sent back to the service provider.

Step 5: Service Provider Checks Authentication Response

When the user arrives at the Service Provider with an authentication response from the Identity Provider, the Service Provider will validate the response, create a session for the user, and send other information from the response to the protected resource. After this, the user is sent to the protected resource they originally tried to access.

Step 6: Resource Returns Content

The user is now at the protected resource but now they have a valid session and the resource knows who they are. Since the user has a valid session, the resource will return the protected content for the user to access.

12.1.1 Apache .htaccess File

Apache `.htaccess` files provide a way to configure changes to a server on a per-directory basis. This file can enable/disable functionality and features that the Apache Web Server software has to offer. These functionalities include redirect and content password protection functionality. If a `.htaccess` file contains one or more configuration directives, those directives will apply to that directory and all subdirectories thereof.

Within our root directory, we have placed a `.htaccess` file with the following content:

```
AuthType shibboleth
ShibRequireSession On
require shibboleth
```

The first line sets the type of user authentication to shibboleth, the single sign-on service being used to authenticate our users. `ShibRequireSession On` forces the user to have an authenticated session before control is passed to the protected resource. `require`

shibboleth is required to activate the Shibboleth authentication module and here we are using the placeholder name shibboleth.

A problem we are currently facing with our single sign-on process is that users are not limited to what directories they can access. If a student signs in through the University of Central Florida single sign-on server, they will be authenticated and proceed to the student landing page. However, if the student then navigates to the admin portal by typing out the path name directly into the address bar, they will successfully gain access. This is obviously an important security vulnerability to address as it leaves the exam creation tools open to any student.

One potential solution to this problem is to add a PHP script to validate each user based on their UCFID. We would need to convert each HTML page into a PHP page and check the user's UCFID every time they accessed a new resource. An example of this would look like:

```
<?php
// Validate type of user
if ( $_SESSION["userType"] == "faculty"){
    http_response_code(401);
    echo '{"message" : "Authorization required" }';
    die(); }
?>
<html>
    <head>
        <title> Admin Page</title>
    </head>
<body>
    ...
</body>
</html>
```

This code checks the `_SESSION` array returned from the Service Provider. If there is no value set for the type of user, then the process is ended. The only problem with this approach is that it may be a bit overkill to start a new PHP thread any time a user needs to go to a new page.

Another solution may be to include a `.htaccess` file in the root path of the admin portal. This file would force any user attempting to access that directory to log in using an ID and password. A `.htpasswd` file would need to be saved somewhere that would store admin usernames and encrypted passwords. Unfortunately, this solution would require that

admins sign in a second time whenever they begin a new session. This is not ideal since we are trying to reduce redundancy.

A third solution would be to add another `.htaccess` file in the root of the admin portal to require that the user be a part of a certain group. This group would be stored in a file on the server and admin IDs would need to be added to it to allow access to the portal. The site should store a cookie on initial authentication that would be accessed when a user tries to navigate into the admin directory. This would prevent the need for a secondary sign-on, however, this method requires more research to implement.

12.2 PHP Development Environment

A constraint for development is that the technology stack provided by the University of Central Florida already exists and the stack must be supported by university staff. The current mechanism for installing software is to use the official Ubuntu package repository. However, this often results in software versions that are significantly behind the current production versions. The current version of the Ubuntu operating system installed on the computer science server is 18.04.3 LTS (Bionic Beaver). The Ubuntu 18.04 branch was released on April 26, 2018. This is a long-term support (LTS) release and is scheduled to reach end of life support status in April 2023.

The version of PHP installed is PHP 7.2.24. This is subject to change each time an Ubuntu patch is installed. The PHP 7.2 code branch was released on November 20, 2017. The PHP 7.2 code branch is scheduled to reach end of life status with suspension of security patches in November 2020. The current stable production release is PHP 7.4.5 which was last updated on April 16, 2020. PHP is used extensively as the backend development tool. It will be used for dynamic web page templates and RESTful API endpoints. With these version constraints, there are PHP development features that cannot be used during development.

PHP 7.3 New Features (Not allowed)

- Flexible Heredoc and Nowdoc Syntaxes
- Allow a trailing comma in function calls
- `JSON_THROW_ON_ERROR`
- `list()` Reference Assignment
- `is_countable` function
- `array_key_first()`, `array_key_last()`
- Make compact function reports undefined passed variables
- Argon2 Password Hash Enhancements
- Same Site Cookie

- PHP 7.4 New Features

PHP 7.4 New Features (Not allowed)

- E_WARNING for invalid containers
- Numeric Literal Separator
- Covariant Returns and Contravariant Parameters
- Allow throwing exceptions from __toString()
- Spread Operator in Array Expression
- Arrow functions 2.0
- FFI - Foreign Function Interface
- Typed Properties 2.0
- Null Coalesce Equal Operator
- Password Hash Registry
- mb_str_split() Split multibyte string
- Change the precedence of the concatenation operator
- Escape PDO "?" parameter placeholder

12.3 Database Environment

The database supplied by the computer science department server is MySQL with version 5.7.29 installed. MySQL 5.7 was initially released on October 21, 2015. The MySQL 5.7 code branch is scheduled to reach end of life status in October 2023. The current stable production release of MySQL is 8.0.19 last updated on January 13, 2020. The list of new features introduced with MySQL 8.0.x is exceedingly long and significant. The following is a small subset of new features not able to be used that could be relevant to our coding needs:

MySQL 8.0 New Features: Not Implementable

- Auto-increment counter value persistent across server restarts
- Default character set has changed from `latin1` to `utf8mb4`.
- JSON `->>` (inline path) operator
- JSON aggregation functions `JSON_ARRAYAGG()` and `JSON_OBJECTAGG()`
- JSON utility function `JSON_PRETTY()`
- Partial, in-place updates of JSON column values
- JSON function `JSON_TABLE()`
- Regular expression support completely reimplemented for Unicode support
- Multi-valued indexes

- CHECK constraints

An empty database was created for the project by computer science department support staff and was named `cspe_db`. An administrator account was created with the username `adm_cspe`. The database resides on the server `eecsserver.cs.ucf.edu`. Access to the database is provided by SSH accounts either on-campus or through Virtual Private Network (VPN) connections using the `cspe` username. Passwords will not be listed in this design document for security reasons.

12.3.1 Schema Design

The schema design for the computer science placement exam includes the following relations: `users`, `exams`, `categories`, `questions`, `results`, and `examContent`. Additional relations may be added during the development cycle. These relations or tables can be visualized with the following entity-relationship diagram.

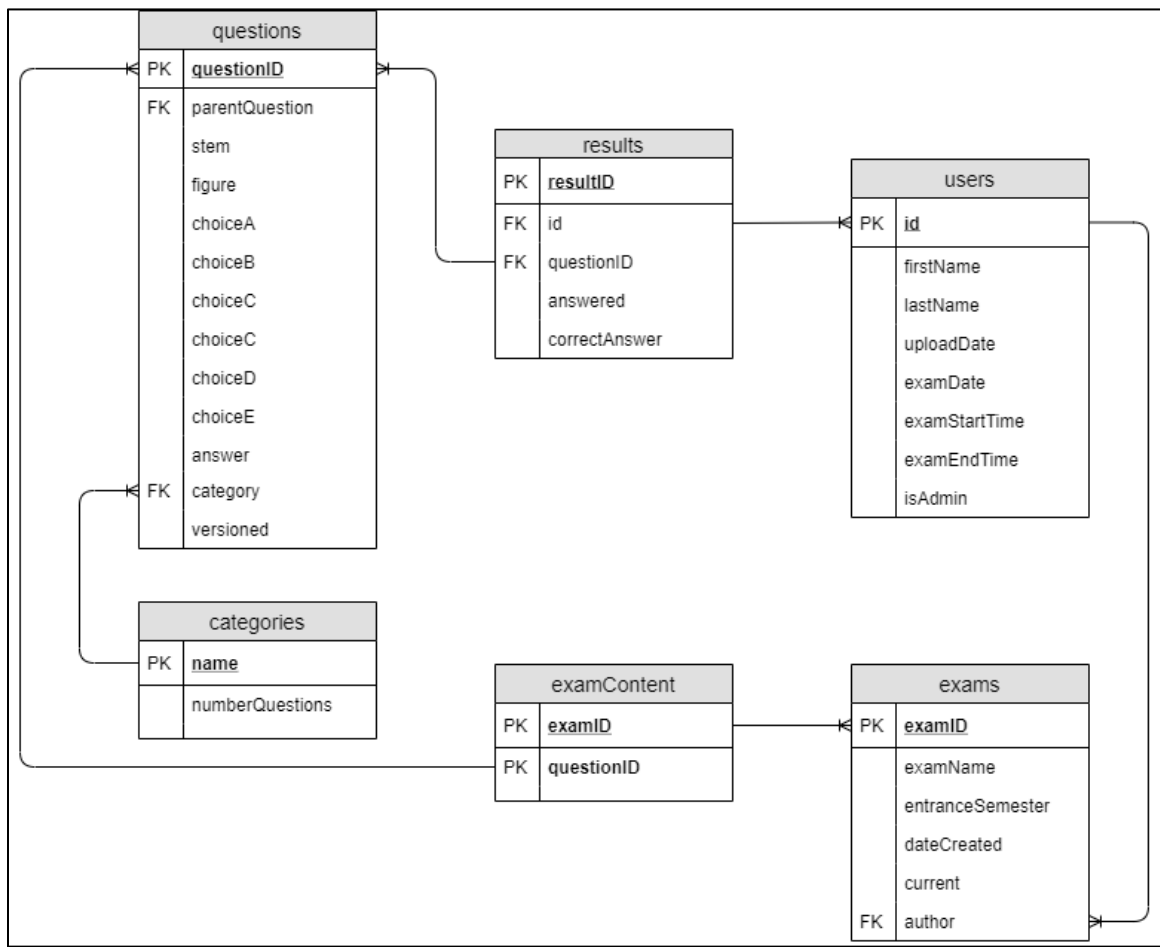


Figure 19: Entity-Relationship diagram.

The users relation is created with the following SQL code:

```
CREATE TABLE users (  
    id                varchar(8) NOT NULL,  
    firstName         varchar(50) NOT NULL,  
    lastName          varchar(50) NOT NULL,  
    uploadDate        date,  
    examDate          date,  
    examStartTime     datetime,  
    examEndTime       datetime,  
    isAdmin           tinyint(1) DEFAULT 0,  
    PRIMARY KEY (id)  
);
```

The users relation will track all types of end users that must access the website. Four types of user accounts have been identified. They include:

- Students unknown to system (no row in the table)
- Student id information received from University of Central Florida enrollment waiting to take exam
- Student id information for those that have already taken the exam once
- Faculty members with administrative access to the web portal

A list of student identification information will be uploaded into this relation every night by processing an inbound batch file. Passwords will not be stored in our database. All users will go through a single sign-on page before accessing the web portal. The single sign-on mechanism will provide as output the UCFID, NID, first name, and last name information. We will store only the information provided by the single sign-on mechanism. Some users will log in with the UCFID and others will login with their network identification (NID). The `id` field has a length to support the format for both identification methods. The `uploadDate` field provides the date that the student first appeared in this table using the nightly batch mechanism. It will be populated with the following trigger.

```
DELIMITER $$  
CREATE TRIGGER userUploadDate BEFORE INSERT ON users  
FOR EACH ROW  
BEGIN  
    IF (NEW.uploadDate IS NULL) THEN  
        SET NEW.uploadDate = CURDATE();  
    END IF;  
END IF;
```

```
END $$
```

The `CREATE TABLE` command in MySQL does not have a native Boolean data type. The `BOOLEAN` keyword can be used, but it is simply syntactic sugar for the data type `tinyint(1)`. This can be verified by showing the table definition after creation with the `DESCRIBE` command. The native data type is used to provide clarity and to avoid a special circumstance where the `TRUE` and `FALSE` keywords fail to translate to the integer values 0 and 1. The four user types can be determined using the `examDate` and the `isAdmin` fields. Since the exam has a time limit, the `examStartTime` and `examEndTime` fields store the time boundaries for taken the exam and provide for exam duration calculations. This information will become part of the statistical metric pages planned for faculty administrators.

The exams relation is created with the following SQL code:

```
CREATE TABLE exams (
    examID          integer NOT NULL AUTO_INCREMENT,
    examName        varchar(50) NOT NULL,
    entranceSemester varchar(50) NOT NULL,
    dateCreated     date,
    author          varchar(8) NOT NULL,
    current         tinyint(1) DEFAULT 0,
    PRIMARY KEY (examID),
    FOREIGN KEY (author) REFERENCES users(id)
);
```

Each exam has a descriptive name such as “*Webcourses Preview Exam*” or “*Guha High School Group*” and is stored in the `examName` field. The system must track the entrance semester and this information may be included in the nightly batch information for students. Each student is assigned an exam to take, and the entrance semester will have a value such as “*2020-Summer*.” This information is currently associated with the `exams` relation in the `entranceSemester` field. This information may be relocated to the `users` relation if it is determined to be a better fit. A student will only ever take one exam, so there is a one-to-one relation between a student and the exam a student takes. Additional columns `dateCreated` and `author` track information on who and when an exam was defined. To make the inserts easier to perform, a trigger will automatically populate the `dateCreated` field if it was not provided in the insert command. It will be populated with the following trigger:

```

DELIMITER $$
CREATE TRIGGER examsCreateDate BEFORE INSERT ON exams
FOR EACH ROW
BEGIN
    IF (NEW.dateCreated IS NULL) THEN
        SET NEW.dateCreated = CURDATE();
    END IF;
END $$

```

Only one exam is classified as the current exam and is identified by the single row containing a value of 1 in the `current` field. All other exams will have this set to `FALSE` or 0. A manual mechanism will allow faculty members to assign a specific exam to a group of students such as an exam given at the start of the *Introduction to Programming with C* course to gauge the skills of students at the start of the class. This would involve extra maintenance to perform this mapping. The `current` field is used for automatic mapping of the current exam to all students uploaded by the nightly batch operations so that no maintenance is needed. Faculty can override this with manual mapping if desired.

The `categories` relation is created with the following SQL code:

```

CREATE TABLE categories (
    name                varchar(50),
    numberQuestions     int DEFAULT 0,
    PRIMARY KEY (name)
);

```

Each exam has questions that test the skills of a student over specific categories. At this time, four specific categories of testing have been identified and are tracked in the `name` field. The four categories of test questions include:

- General
- Conditionals
- Loops
- Functions

Storing this information in a separate relation allows for future categories to be created with a name such as “*Ungraded*” or “*Field-Testing*.” An algorithm will automatically map some number of questions from the total pool of question to a specific exam. The algorithm must ensure that an even distribution of questions come from each category. The `numberQuestions` field will be used by this algorithm for automatic maintenance free mapping. Faculty members will have the option to manually adjust the mapping of

questions to a specific exam if desired, but the automatic solution will not require this. Manual mapping will allow for specific exams to have different numbers of questions. The initial rollout will have an exam of 20 questions.

The `questions` relation is created with the following SQL code:

```
CREATE TABLE questions (  
    questionID      integer NOT NULL AUTO_INCREMENT,  
    parentQuestion  integer,  
    stem            varchar(1536) NOT NULL,  
    figure          blob,  
    choiceA         varchar(160) NOT NULL,  
    choiceB         varchar(160) NOT NULL,  
    choiceC         varchar(160),  
    choiceD         varchar(160),  
    choiceE         varchar(160),  
    answer          varchar(1) NOT NULL,  
    category        varchar(50),  
    versioned       tinyint(1) DEFAULT 0,  
    PRIMARY KEY (questionID),  
    FOREIGN KEY (category) REFERENCES categories(name)  
);
```

There are two types of questions that need to be stored. Simple individual questions that do not contain any variations of themselves are identified with the `versioned` field set to `FALSE` or `0`, along with the `parentQuestion` field containing a `NULL` value. These are the easiest types of questions to input into the system. External programs written in languages such as Python or JavaScript can employ Automatic Item Generation (AIG) where a question becomes a base template, and many variations that randomize things like integers in the question and the corresponding answer exist. These variations are basically the same question using slightly different numbers. The `questions` relation must support these types of questions. An AIG style questions will have the `versioned` field set to `TRUE` or `1`. The base template used for a versioned question can be identified by `questionID = parentQuestion` fields. Its many variations or children will be mapped to the base using the `parentQuestion` field, such as `questionID=21` is associated with `parentQuestion=5` (the base template). Although this is a relationship, a self-referencing foreign key was not used in the table creation since not all questions will have this feature. The text of the question is stored in the `stem` field. The maximum length of a question was determined from the sample exam asset provided, and this length was further increased to allow for longer questions. It was decided that a maximum of five answers be allowed for each question, and those are stored in the `choiceA`, `choiceB`, `choiceC`,

choiceD, and choiceE fields. The length of these fields was determined from the sample exam asset provided, and this length was further increased to allow for longer answers. The answer for a question is stored in the answer field as a single letter such as 'a' or 'e'. The sponsor initial kickoff meeting indicated that a nice feature to have is the ability to store an illustration or figure with each question. The figure field is for storing that illustration. Currently there are no questions in the sample pool that have illustrations associated with them.

The examContent relation is created with the following SQL code:

```
CREATE TABLE examContent (  
    examID          integer NOT NULL,  
    questionID      integer NOT NULL,  
    PRIMARY KEY (examID, questionID),  
    FOREIGN KEY (examID) REFERENCES exams(examID),  
    FOREIGN KEY (questionID) REFERENCES questions(questionID)  
);
```

This is a mapping table designed to implement the many-to-many relationship between the exams relation and the questions relation. When a new exam is created by a faculty member, an automatic algorithm will assign a set number of questions to that exam following a distribution model that pulls a subset of questions from each category of questions. If the question pulled is a versioned question, then the algorithm will select a random version of the same question to be used. The output of that algorithm is stored in the examContent relation. A faculty member though will be presented a grid of this mapping to allow for manual adjustments, such as guaranteeing that a specific question be included on the exam instead of relying on a randomization algorithm.

The results relation is created with the following SQL code:

```
CREATE TABLE results (  
    resultID        integer NOT NULL AUTO_INCREMENT,  
    id              varchar(8) NOT NULL,  
    examID          integer NOT NULL,  
    questionID      integer NOT NULL,  
    answered        varchar(1),  
    correctAnswer   varchar(1),  
    PRIMARY KEY (resultID),  
    FOREIGN KEY (id) REFERENCES users(id),  
    FOREIGN KEY (examID) REFERENCES exams(examID),  
    FOREIGN KEY (questionID) REFERENCES questions(questionID)  
);
```

When taking the exam, students are presented with a web page containing a single question due to the possible length of the question and the answers. That page will have forward and backward buttons to advance through the questions. Each time the student clicks the navigation buttons, an API call will update the `results` relation. This operation is known as an “*upsert*,” i.e. insert a row in the table if it does not exist otherwise update the row if it already exists. A typical row in the `results` relation maps the information of a student with `id=1234567` on `examID=1` supplied `answered='c'` for `questionID=10`. As this row is being inserted into the table, a trigger will retrieve the correct answer for that table and populate it immediately. This is to simplify the API call to determine the score at the end of the exam. The answer is not to be found anywhere on the data sent to the client browser and will be populated with the following trigger:

```
DELIMITER $$
CREATE TRIGGER getanswer BEFORE INSERT ON results
FOR EACH ROW
BEGIN
    SET @var = NULL;
    SELECT answer INTO @var FROM questions
        WHERE questionID = NEW.questionID;
    SET NEW.correctAnswer = @var;
END $$
```

The API can then determine the number of correct answers directly from the table by counting the rows for a given student where `answered == correctAnswer`. The number of incorrect answers is simply the number of rows for a given student where `answered != correctAnswer`.

A nice to have feature is to track the time students spend on each question for statistical metrics. This will require two timers to be running simultaneously, one for the overall exam time limit, and another for the time spent on an individual question. Tracking this time will present a challenge since students can use the back and forward buttons while taking the exam and return to previous questions if they did not answer them or to change their answer. Since a row in the `results` relation represents a student’s interaction with a single question, additional fields may be added to this table to hold accumulated time spent on a question. It might also be easier and perform better to have a temporary table to track timestamp entering a question and leaving a question. The rows of the temporary table could then be grouped and summed for each question.

This is our current entity-relationship schema but as the project develops, it will understandably change and grow and reshape itself in order to offer the best possible system.

12.4 RESTful API

12.4.1 RESTful API Organization

Representational State Transfer (REST) was defined by Roy Fielding in his 2000 PhD dissertation at UC Irvine titled “*Architectural Styles and the Design of Network-based Software Architectures*.” Web service Application Programming Interfaces (APIs) that adhere to REST standards are called RESTful APIs.

The most relevant guiding constraints, defined in the dissertation, behind RESTful APIs for our project are the goals of a uniform interface of a layered system. Because we have many different kinds of users who will be able to access our website, a uniform interface ensures that our components are general enough to be reusable, maintainable, and configurable while our layered interface protects users from accessing privileged information that they should not be able to access without the proper credentials. Although we will strive to be within the bounds of all six of the guiding constraints, those two are the most important behind our API design considerations.

The computer science department server `eecsserver.cs.ucf.edu` uses an external file system `thumper:/storage/home/nonhuman/cspe` mounted locally onto `/home/cspe` as the location for this project. The external file system has a capacity of 11 TB currently with 6.5 TB free space. This external file system appears to be used to store the personal home directories of University of Central Florida faculty members and selected graduate students in addition to hosting this project. Subdirectories have been created to store all the assets needed for the computer science placement exam and are organized as shown in Figure 20, pictured below.

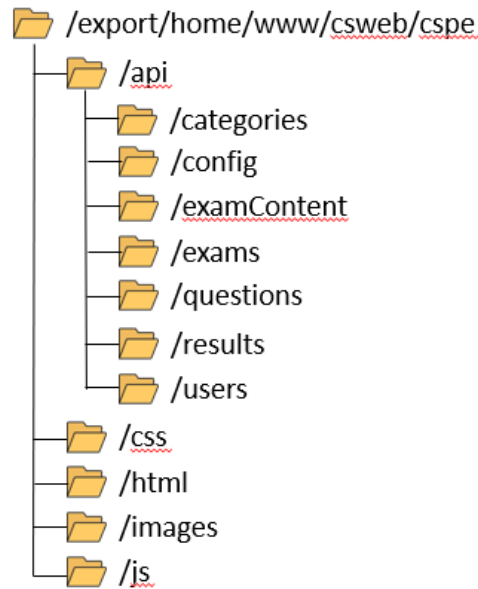


Figure 20: A depiction of our directory structure on the CS department server

12.4.2 RESTful API Endpoints

The RESTful API is written in PHP and there is a folder for each relation as well as a `config` folder to store common configuration files used by all relations. The following REST API endpoints have been created and tested this semester. Additional REST API endpoints will be developed later in the life cycle of this project.

Table 29: A mapping of HTTP methods, REST API endpoints, and their intended use

HTTP Method	REST API Endpoint	Purpose
GET	<code>./api/config/login.php</code>	Development Security
GET	<code>./api/config/logout.php</code>	Development Security
GET	<code>./api/config/database.php</code>	Database Commonality
GET	<code>./api/config/rebuild.php</code>	Development Refresh
GET	<code>./api/config/viewlogs.php</code>	Troubleshooting
GET	<code>./api/config/users/read.php</code>	CRUD – Read all rows
GET	<code>./api/config/users/readByKey.php</code>	CRUD – Read one row
POST / PUT	<code>./api/config/users/insert.php</code>	CRUD – insert row
PUT / PATCH	<code>./api/config/users/update.php</code>	CRUD – update row
DELETE	<code>./api/config/users/delete.php</code>	CRUD – delete row
GET	<code>./api/config/exams/read.php</code>	CRUD – Read all rows
GET	<code>./api/config/exams/readByKey.php</code>	CRUD – Read one row
POST / PUT	<code>./api/config/exams/insert.php</code>	CRUD – insert row
PUT / PATCH	<code>./api/config/exams/update.php</code>	CRUD – update row

DELETE	<code>./api/config/exams/delete.php</code>	CRUD – delete row
GET	<code>./api/config/categories/read.php</code>	CRUD – Read all rows
GET	<code>./api/config/categories/readByKey.php</code>	CRUD – Read one row
POST / PUT	<code>./api/config/categories/insert.php</code>	CRUD – insert row
PUT / PATCH	<code>./api/config/categories/update.php</code>	CRUD – update row
DELETE	<code>./api/config/categories/delete.php</code>	CRUD – delete row
GET	<code>./api/config/questions/read.php</code>	CRUD – Read all rows
GET	<code>./api/config/questions/readByKey.php</code>	CRUD – Read one row
POST / PUT	<code>./api/config/questions/insert.php</code>	CRUD – insert row
PUT / PATCH	<code>./api/config/questions/update.php</code>	CRUD – update row
DELETE	<code>./api/config/questions/delete.php</code>	CRUD – delete row
GET	<code>./api/config/examContent/read.php</code>	CRUD – Read all rows
GET	<code>./api/config/examContent/readByKey.php</code>	CRUD – Read one row
POST / PUT	<code>./api/config/examContent/insert.php</code>	CRUD – insert row
PUT / PATCH	<code>./api/config/examContent/update.php</code>	CRUD – update row
DELETE	<code>./api/config/examContent/delete.php</code>	CRUD – delete row
GET	<code>./api/config/results/read.php</code>	CRUD – Read all rows
GET	<code>./api/config/results/readByKey.php</code>	CRUD – Read one row
POST / PUT	<code>./api/config/results/insert.php</code>	CRUD – insert row
PUT / PATCH	<code>./api/config/results/update.php</code>	CRUD – update row
DELETE	<code>./api/config/results/delete.php</code>	CRUD – delete row
POST	<code>./api/config/results/getScore.php</code>	Calculate exam grade

The complete source code to all RESTful API endpoints will be stored in the private University of Central Florida GitHub account. A repository named `spring20-group8` was created for this purpose. Some RESTful API endpoints were created specifically for testing purposes and function on individual developer machines where full access including root authority is available. These include `login.php`, `logout.php`, `rebuild.php`, and `viewlogs.php`. The `login.php` and `logout.php` files allow local development by simulating the effects of single sign-on when not running live on the University of Central Florida production server. The `rebuild.php` file drops all tables, purges all data, rebuilds all tables, and repopulates the tables with sample development data. This will not be placed on the production server since dropping tables would purge valid historical data that should always be preserved. The `viewlogs.php` file performs a dump of various critical troubleshooting log files useful to development, but access to the equivalent files may not be available in production.

The remaining RESTful API endpoints developed this semester provide the basic functions of persistent storage known as Create, Read, Update, and Delete (CRUD). Their creation allows for the front-end development to begin immediately when the next semester starts. The only front-end development that took place this semester was using the single sign-on mechanism and then directing the user to one of four webpages based on their role. Once

the CRUD operations are established, then the front-end can begin to develop the first capabilities of our website to interact with our database. Full design of how the data gathered from the database will be presented can begin development shortly after.

There are not many open source PHP API tutorials of high quality available, especially for API development without the complexities of custom frameworks. Some of the commercial PHP API frameworks commonly used are Slim, Lumen, Laravel, Symfony 4, and Silex. This project has a budget of \$0 and it is expected that no commercial software can be purchased.

Some team members have experience with a PHP API framework from the website Code of Ninja owned by Mike Dalisay [22]. Their terms of use state, “*The resources on The Code of a Ninja can be used freely in personal and commercial projects. Summarized, use it freely, integrate it, make it your own, but don’t copy and paste our work and sell it or claim that it’s yours, stay fair.*” It is one of the more comprehensive open source PHP API tutorials available that illustrates full end-to-end PHP API creation. The tutorial uses an object-oriented approach to PHP development and creates classes that represent each database relation. These classes define object variables that correspond to the database relation attributes (table columns). They also define class methods that represent the basic CRUD functionality for each relation. However, a separate API file such as `./api/users/create.php` is used to instantiate the class object `./api/objects/users.php` and invoke the class methods passing information to the methods and processing the returns. This implies that every CRUD operation has its logic split between at least two files, and sometimes more files. Also, any errors or bugs that find their way into an object file often has the effect of breaking every CRUD API call that uses that common object file. The manner of one class calling methods of another class also limits the ability to respond with a multitude of HTTP status codes.

Another website that has been greatly beneficial in the PHP RESTful API development is the website PHP Delusions. The site author claims to be the only person to hold a gold badge in PDO, MYSQLI, and SQL-Injection on Stack Overflow. This site goes to great lengths to show that most references and examples on using PHP PDO access to a database are wrong and the site claims to offer “*(The only proper) PDO tutorial*”. It points out fallacies and poor code examples that are even referenced on the official PHP.net website. Our approach to PHP RESTful API development is a simplistic procedural approach rather than a more complex-object oriented approach. It is also a vanilla approach that it not constrained by the usage of a PHP framework. Even though the majority of the code is procedural in nature, the usage of PHP Data Objects (PDO) are in fact classes.

Once the basic RESTful APIs for CRUD operations are developed for a single relation, they in essence become templates that are basically the same for all other relations in the database. Therefore, this design document will not expose the source code for every RESTful API Endpoint that was defined earlier. Instead, this design document will explain and document the basic templates that were developed.

12.4.3 Database.php

The first and common template is the `database.php` file. The source code is explained in sections as follows:

```
<?php
// Database Configuration
$host = "localhost";
$database = "cspe_db";
$user = "adm_cspe";
$password = "<not listed here>";
$charset = 'utf8mb4';
```

The above section of the `database.php` file is used to create the Data Source Name (DSN) for the PDO connection. It is a common file that is included in every other RESTful API file.

```
// Development Settings
// ini_set('display_errors', 1);

// Production Settings
ini_set('display_errors', 0);
ini_set('log_errors', 1);
```

The above section of the `database.php` file determines how error messages are displayed. On a development server, this causes a full stack trace dump to be displayed to the screen. This information could contain privileged information. On a production server, the display of errors is turned off and the errors are redirected to a log file instead.

```
// Data Source Name (DSN) for PDO connection
$dsn = "mysql:host=$host;dbname=$database;charset=$charset";

// PDO::ERRMODE_EXCEPTION: Throw exceptions.
// PDO::FETCH_ASSOC: returns an array indexed by column name //
as returned in your result set
// PDO::ATTR_EMULATE_PREPARES: to try to use native prepared
```

```
// statements (if FALSE)
$options = [
    PDO::ATTR_ERRMODE           => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES  => false,
];
```

The above section of the `database.php` file defines the Data Source Name (DSN) for the PDO connection and defines various PDO options.

```
// The only try catch clause needed. PDO errors are
// automatically handled
try {
    $pdo = new PDO($dsn, $user, $password, $options);
} catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e-
    >getCode());
}
```

Contrary to almost every PHP PDO tutorial, it is not necessary to surround every database call with a Try/Catch clause. The only time this is explicitly needed is when the connection is first created. After that, PHP PDO automatically handles exceptions and will report on any errors that occur.

```
// Convert errors into exceptions
set_error_handler(function ($level, $message, $file = '', $line =
0)
{
    throw new ErrorException($message, 0, $level, $file, $line);
});
```

Some problems that occur such as foreign-key constraint violations appear as errors instead of exceptions and are not gracefully handled by PDO. The above snippet defines an error handler that captures those errors converts them into exceptions preserving the original message.

```
// Report errors for development settings, otherwise give
// clean message if in production
set_exception_handler(function ($e)
{
    error_log($e);
    http_response_code(500);
    if (ini_get('display_errors')) {
        echo $e;
    }
});
```



```

    } else {
        header("Content-Type: text/html; charset=UTF-8");
        echo "<h1>500 Internal Server Error</h1>
            An internal server error has been occurred.<br>
            Please try again later.";
    }
});
?>

```

The final snippet of code from the database.php file sets how exceptions are handled. On a development server, the exceptions are displayed but on a production server an alternative web page is shown to the user that will not reveal any sensitive information. The alternative web page protects against potential security issues that arise from giving the user too much information about the underlying source code, including stack traces.

12.4.4 Read.php (Basic CRUD template)

```

<?php
// Start the session
session_start();

// required header
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");

// include database and table objects
include_once '../config/database.php';

// Validate type of user
if ( isset($_SESSION["userType"]) &&
    $_SESSION["userType"] == "faculty" )
{
    http_response_code(401);
    echo '{"message" : "Authorization required" }';
    die();
}

// Retrieve all rows
$data = $pdo->query('SELECT * FROM categories')->fetchAll();

// Package as JSON for the client browser
if ($data) {
    http_response_code(200);
    echo json_encode($data);
}

```

```

} else {
    http_response_code(200);
    echo '{"message" : "No rows found" }';
}

?>

```

The `read.php` template is a simple template and uses the HTTP GET request method. The headers are used to implement Cross-Origin Resource Sharing (CORS) for additional security needs. The only changes that need to be made to this template when used against other database relations is the change the table name in one location of the SQL SELECT statement.

There are multiple layers of security. All these RESTful API files sit in a directory on the web server that is first protected by the University of Central Florida single sign-on mechanism. CORS headers adds another layer of security, and PHP session variables add yet another layer of security.

The CORS headers serve as a barrier against potential security issues that come from accessing material from different origins [23]. JavaScript blocks attempts at cross-domain calls as a security feature, but sometimes cross-domain access is a requirement or a desirable configuration for development purposes such that issues related to cross-origin requests are handled gracefully.

12.4.5 ReadByKey.php (Basic CRUD template)

```

<?php
// Start the session
session_start();

// required headers
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: access");
header("Access-Control-Allow-Methods: GET");
header("Access-Control-Allow-Credentials: true");
header("Content-Type: application/json; charset=UTF-8");

// include database and table objects
include_once '../config/database.php';

// Validate type of user
if ( isset($_SESSION["userType"]) &&

```

```

        $_SESSION["userType"] == "faculty" )
    {
        http_response_code(401);
        echo '{"message" : "Authorization required" }';
        die();
    }

    // Determine if required fields have been set
    if ( isset($_GET['key']) ) {
        $key = $_GET['key'];
    } else {
        http_response_code(400);
        echo '{"message" : "Missing required data" }';
        die();
    }

    // Process SQL
    $stmt = $pdo->prepare('SELECT * FROM categories WHERE name = ?');
    $stmt->execute([$key]);
    $data = $stmt->fetch();

    // Send results back as JSON
    if ($data) {
        http_response_code(200);
        echo json_encode($data);
    } else {
        http_response_code(200);
        echo '{"message" : "No rows found" }';
    }
}

?>

```

The `readByKey.php` template adds a little more as compared to the `read.php` template. It also uses the HTTP GET request method. Additional CORS headers are shown. This template uses a PDO prepared statement and binds a key to it when called. The key is specified on the URL with a call such as:

<https://eecserver.cs.ucf.edu/cspe/api/readByKey.php?key=Loops>

This can easily be converted to a HTTP POST request method to avoid putting any extra information on the URL, but this design will be called by an AJAX method from the browser and not be displayed on the screen. Even if it were to be displayed, the phrase “key” is being used even though the actual primary key column of the database relation

has a different name. The difference between the two read templates is that `read.php` returns every row of the table and `readByKey.php` only returns a single specified row.

Among many implementations, the `ReadByKey.php` file will be used for accessing the information of different questions as a student proceeds through their exam, on the administrative portal when selecting edit/view options, and on the initial page where students enter their NID for verification that they are enrolled to take the exam.

12.4.6 Delete.php (Basic CRUD template)

```
<?php
// Start the session
session_start();

// required headers
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: access");
header("Access-Control-Allow-Methods: DELETE");
header("Access-Control-Allow-Credentials: true");
header("Content-Type: application/json; charset=UTF-8");

// include database and table objects
include_once '../config/database.php';

// Validate type of user
if ( isset($_SESSION["userType"]) &&
    $_SESSION["userType"] == "faculty" )
{
    http_response_code(401);
    echo '{"message" : "Authorization required" }';
    die();
}

// Determine if required fields have been set
$postData = json_decode(file_get_contents("php://input"));

if (empty($postData->name)) {
    http_response_code(400);
    echo '{"message" : "Missing required data" }';
    die();
}

// Debugging Code to display content of PHP variable
// echo json_encode($postData); die();
```

```

// Process SQL
$stmt = $pdo->prepare('DELETE FROM categories WHERE name = ?');
$stmt->execute([$postData->name]);
$data = $stmt->rowCount();

// Send results back as JSON
if ($data) {
    http_response_code(200);
    echo '{"message" : "Deleted ' . $data . ' row" }';
} else {
    http_response_code(200);
    echo '{"message" : "No rows deleted" }';
}

?>

```

Once a basic template using PDO prepared statements has been implemented, there is little change needed whether it is a SQL query, insert, update, or delete command. PDO methods do not distinguish between the types of SQL being executed. The `delete.php` template simply changes the SQL in the previous `readByKey.php` template to delete the record being referenced by the same key. It uses the HTTP DELETE request method.

12.4.7 Insert.php (Basic CRUD template)

```

<?php
// Start the session
session_start();

// required headers
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: access");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Allow-Credentials: true");
header("Content-Type: application/json; charset=UTF-8");

// include database and table objects
include_once '../config/database.php';

// Validate type of user
if ( isset($_SESSION["userType"]) &&
    $_SESSION["userType"] == "faculty" )
{
    http_response_code(401);
}

```

```

        echo '{"message" : "Authorization required" }';
        die();
    }

    // Determine if required fields have been set
    $postData = json_decode(file_get_contents("php://input"));

    if (empty($postData->name)) {
        http_response_code(400);
        echo '{"message" : "Missing required data" }';
        die();
    }

    // Process SQL
    $stmt = $pdo->prepare('INSERT INTO categories(name) VALUES(?)');
    $stmt->execute([$postData->name]);
    $data = $stmt->rowCount();

    // Send results back as JSON
    if ($data) {
        http_response_code(200);
        echo '{"message" : "Inserted category - ' . $postData->name
        . '" }';
    } else {
        http_response_code(200);
        echo '{"message" : "Unable to insert row" }';
    }

    ?>

```

At this point there should be little surprise that the template `insert.php` is almost identical to the other templates. The `categories` relation is a simple table with just two columns. The only change to the above template for other relations is the addition of other columns that must be accounted for. This template also returns the primary key value of the row that was just inserted. In the `categories` table, the `name` column serves as the primary key. For other relations that use an `AUTO_INCREMENT` primary key column, the PDO function `lastInsertId()` returns the integer number that was used by the `AUTO_INCREMENT` feature.

12.4.8 Update.php (Basic CRUD template)

```
<?php
// Start the session
session_start();

// required headers
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: access");
header("Access-Control-Allow-Methods: PUT");
header("Access-Control-Allow-Credentials: true");
header("Content-Type: application/json; charset=UTF-8");

// include database and table objects
include_once '../config/database.php';

// Validate type of user
if ( isset($_SESSION["userType"]) &&
    $_SESSION["userType"] == "faculty" )
{
    http_response_code(401);
    echo '{"message" : "Authorization required" }';
    die();
}

// Determine if required fields have been set
$postData = json_decode(file_get_contents("php://input"));

if (empty($postData->name) ||
    empty($postData->numberQuestions) ) {
    http_response_code(400);
    echo '{"message" : "Missing required data" }';
    die();
}

// Process SQL
$stmt = $pdo->prepare('UPDATE categories SET name = ?,
                      numberQuestions = ? WHERE name = ?');
$stmt->execute([$postData->name, $postData->numberQuestions,
               $postData->name]);
$data = $stmt->rowCount();

// Send results back as JSON
if ($data) {
    http_response_code(200);
    echo '{"message" : "Update row for category - ' . $postData->name . '" }';
}
```

```

} else {
    http_response_code(200);
    echo '{"message" : "Unable to update row" }';
}

?>

```

The `update.php` template is almost identical to the `insert.php` template. Instead it uses the HTTP PUT request method. It also requires the primary key to be supplied to identify which row to update whereas the `insert.php` template often does not use the primary key attribute since the database will usually automatically assign the primary key value.

12.4.9 Getscore.php

```

<?php
// Start the session
session_start();

// required headers
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: access");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Allow-Credentials: true");
header("Content-Type: application/json; charset=UTF-8");

// include database and table objects
include_once '../config/database.php';

// Validate type of user
if ( isset($_SESSION["userType"]) &&
    $_SESSION["userType"] == "" )
{
    http_response_code(401);
    echo '{"message" : "Authorization required" }';
    die();
}

// Determine if required fields have been set
$postData = json_decode(file_get_contents("php://input"));

if (empty($postData->id) || empty($postData->examID)) {
    http_response_code(400);
    echo '{"message" : "Missing required data" }';
}

```



```

        die();
    }

    // Debugging Code to display content of PHP variable
    // echo json_encode($postData); die();

    // Process SQL
    $stmt = $pdo->prepare('SELECT AVG(marks) as score
        FROM (SELECT CASE WHEN answered=correctAnswer THEN 1
            ELSE 0 END marks
            FROM results WHERE id=? AND examID = ?) marks');

    $stmt->execute([$postData->id, $postData->examID]);
    $data = $stmt->fetchAll();

    // Send results back as JSON
    if ($data) {
        http_response_code(200);
        echo json_encode($data);
    } else {
        http_response_code(200);
        echo '{"message" : "Unable to get score" }';
    }
}

?>

```

The `getscore.php` API file is used to calculate the grade when the student submits the exam. This is able to be done with a simple SQL `SELECT` query due to the trigger on the `results` relation that populates the answer of each question the moment a row is inserted into the table.

12.5 RESTful API Testing

Various tools are being used to test the RESTful API calls during development, passing any required POST bodies to the API and observing the response bodies received. These tools include:

- POSTMAN by [postman.com](https://www.postman.com) (Free version for individuals)
- OpenAPI (formally known as Swagger)
- cURL command line (Open Source)
- Browser Extensions to format JSON output received for easy reading

The OpenAPI tool formally known as Swagger is supported by SmartBear Corporation, but the free open-source edition is being used. Swagger-UI provides a nice web interface

that serves both as API documentation and API testing, but it requires a `project.yml` file to be created to describe the API. This file can be rather lengthy at around 700 to 1000 lines. The `.yml` file extension sometimes listed as `.yaml` means “*YAML Ain’t Markup Language*.” This is planned to be delivered as part of the final documentation, but during development phases POSTMAN is more agile and easier to change.

13 Steps Going Forward

For the future content of the exams going forward, the pool of questions for the exam needs to either be greatly expanded or more carefully tailored based on the data that is generated from the first few sittings of this exam. Baselines have to be established across various groups of participants: middle school students who are taking programming camps, high school students taking programming classes and competing in programming competitions, students who have taken *Introduction to Computing* and who will be enrolling in *Introduction to Programming with C*, students who are about to enroll in *Introduction to Programming with C* without having taken *Introduction to Computing*, and students after they have completed *Introduction to Programming with C*. Establishing the different scores across these different populations guarantees that the exam tests the right concepts, measures the faculty of students to understand and utilize these concepts, and accurately predicts success in *Introduction to Programming with C*. After this data has been gathered and analyzed, then the decision about current content and future content for the exam, as well as the cut off scores for passing, can be made properly. Writing questions will probably be a continual part of the process as we also develop the website as well. The authors of these questions will be ourselves along with volunteers such as Prof. Guha and further discussion about the content for the Fall 2020 semester will need to be had.

During Senior Design 2, to complete this project, we will be completely developing the front-end web application to work effectively with the APIs and database we currently have laid out. Once our front-end and back-ends can talk to each other, we will look at the design as a whole and make any small modifications we see fit. These changes will be geared to ensuring scalability, manageability, accessibility, and maintainability.

Additional tasks include the bidirectional communication and protocols between the CS Placement exam website and the University of Central Florida enrollment systems. The specifications of this data exchange have not been finalized as of this date of this paper. We have sent multiple emails related to this topic and we are dependent on University of Central Florida employees to define their side of the communication in this important task. Without this information, then the initial student verification step to guarantee that the right

students are taking this exam cannot be made and the integrity of the scores and data collected will be seriously compromised.

We have also not had any serious discussions of how University of Central Florida support staff are going to maintain the system going forward. This includes applying security patches to the underlying operating system and technology stacks, changing versions of the software with validation testing, providing live support to students taking the exam such as student browser issues, loss of connectivity during the middle of the exam, and accommodating students with disabilities. It also includes performing backup and recovery tasks and guaranteeing the retention of the historical student information that might be considered a part of their academic records since it determines eligibility to take specific classes. These discussions will need to take place in order for us to create relevant documentation to help ensure that these underlying systems changes will still yield all of the correct functionalities and capabilities that the current one does.

For the administrative metric collections, deciding what data to be collected exactly, how it will be collected and measures, and how the data will be visualized is another step that must be taken to coincide with the development of the front-end to develop such capabilities. Gathering input from faculty about what kinds of metrics would be useful to see and what metrics would be likely to be actually used to make decisions regarding the exam needs to be done to make sure that we do not develop functionalities that will go untouched or lack ones that would be used often.

14 Project Summary

Concluding this semester, one of our two project goals has been completed. We have reviewed, critiqued, and contributed to the question bank for the Summer 2020 Webcourses placement exam. This temporary exam location is independent from the system we will be bringing to life next semester, but those testing concepts and exam format were part of our overall project scope. The content of this exam has been carefully crafted from the detailed analysis of the initial sample exam asset we were given and research into fundamental programming concepts that are taught in textbooks and in the curricula of the courses that this exam is relevant to. A new revised set of instructions has been created, the pseudocode has been formalized more rigidly, replacement questions were provided, and suggestion for new content beyond this has been provided. This document itself serves as a guide for future authors of exam content as well as for future developers to see what was considered and why decisions were made in order to execute the second part of our project.

For the next part of our project, the website, this semester we were able to complete the initial design and implementation of our database entities, as well as develop API endpoints for the CRUD functions on these entities. We also completed the front-end wireframes and storyboards to map the design and user experience for nonregistered students, registered students, and administrators in order to guide us through the process of developing the front-end. We are all eager for the tasks and challenges that lay before us.

15 References

- [1] A. Doebler and H. Holling. "A processing speed test based on rule-based item generation: An analysis with the Rasch Poisson Counts model", In *Learning and Individual Differences*, pages 1-8, 2015. doi: 10.1016/j.lindif.2015.01.013
- [2] S. Hines. "The development and validation of an automatic-item generation measure of cognitive ability". Dissertation, 2017. Retrieved from <https://digitalcommons.latech.edu/dissertations/71>
- [3] Irvine, S. "The foundations of item generation for mass testing", In S. H. Irvine & P. C. Kyllonen (Eds.), *Item generation for test development*, pages 3-32. Hillsdale, NJ: Erlbaum, 2002.
- [4] M. J. Gierl and H. Lai. "The Role of Item Models in Automatic Item Generation", In *International Journal of Testing*, 12:3, pages 273-298, 2012, doi: 10.1080/15305058.2011.635830
- [5] H. Geerlings et al. "Modeling rule-based item generation." In *Psychometrika*, 76, pages 337-359, 2011. doi: 10.1007/S11336-011-9204-X
- [6] A. Kosh et al. "A Cost-Benefit Analysis of Automatic Item Generation", In *Educational Measurement: Issues and Practice*. 38, 2018. 10.1111/emip.12237.
- [7] L. Zavala and B. Mendoza. "On the Use of Semantic-Based AIG to Automatically Generate Programming Exercises." In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 14–19. Baltimore, Maryland, USA: ACM, 2018.
- [8] R. B. Frary. "More Multiple-Choice Item Writing Do's and Don'ts," In *Practical Assessment, Research, and Evaluation*: Vol. 4, Article 11, 1994. Available at: <https://scholarworks.umass.edu/pare/vol4/iss1/11>
- [9] C Operator Precedence. P12. Cppreference.com. Accessed Feb., 2020. Retrieved from https://en.cppreference.com/w/c/language/operator_precedence
- [10] T. Sanderson. THE UP-GOER FIVE TEXT EDITOR. Retrieved from <https://splasho.com/upgoer5/>
- [11] B. Long and A. Long. Hemingway Editor. Retrieved from <http://hemingwayapp.com>

- [12] Readable. Text Analysis Tool. Retrieved from <https://app.readable.com/text/>
- [13] L. Kelly. "How did the Automated Readability Index become an essential tool for technical writers?", 2019 Retrieved from <https://readable.com/blog/how-did-the-automated-readability-index-become-an-essential-tool-for-technical-writers/>
- [14] B. Ortutay. "Why is there so much attention but so little progress for diversity in tech?". *chicagotribune.com*. Chicago Tribune. 25 Jan 2017.
- [15] E. Elliot. "Top JavaScript Frameworks and Topics to Learn in 2020 and the New Decade", On "Medium", 2020. Retrieved from <https://medium.com/javascript-scene/top-javascript-frameworks-and-topics-to-learn-in-2020-and-the-new-decade-ced6e9d812f9>
- [16] P. Balbier. "What Is the React.js Framework? When and Why Should I Use React.js in My Project?" On *Netguru*, 2019. Retrieved from <https://www.netguru.com/blog/what-is-react-js>
- [17] N. Pandit. "What and Why is React.js", On *C#Corner*, 2020. Retrieved from <https://www.c-sharpcorner.com/article/what-and-why-reactjs/>
- [18] S. Arora. "8 Proven Reasons You Need Angular for Your Next Development Project", On *Grazitti*, 2018. Retrieved from <https://www.grazitti.com/blog/8-proven-reasons-you-need-angular-for-your-next-development-project/>
- [19] A. Rusev. "React vs Vue: The Core Differences", On *Mentormate*, 2019. Retrieved from <https://mentormate.com/blog/react-vs-vue-the-core-differences/>
- [20] C. Achard. "React vs. Plain Javascript", On *Framer*, 2019. Retrieved from <https://www.framer.com/blog/posts/react-vs-vanilla-js/>
- [21] Mindfire Solutions. "Flask vs Django", On *Blogs @ Mindfire Solutions*. Retrieved from <http://www.mindfiresolutions.com/blog/2018/05/flask-vs-django/>
- [22] M. Dalisay. "Terms of Use," On *Code Of A Ninja*, 2019. Retrieved from <https://www.codeofaninja.com/terms-of-use>
- [23] S. Hobbs. "CORS Tutorial: A Guide to Cross-Origin Resource Sharing", On *Auth0*, 2019. Retrieved from <https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/>