2019

A Semester Long Classroom Course Mimicking a Software Company and a New Hire Experience for Computer Science Students Preparing to Enter the Software Industry

David A. Chamberlain
Bridgewater State University

A Semester Long Classroom Course Mimicking a Software Company and a New Hire Experience for Computer Science Students Preparing to Enter the Software Industry

A Thesis Presented

By

DAVID A. CHAMBERLAIN

AUGUST 2019

Approved as to style and content by:

Signature: _____

Dr. Michael Black, Advisor Date

Signature: _____

Dr. Haleh Khojasteh Date

Signature: _____

Dr. John Santore Date

A Semester Long Classroom Course Mimicking a Software Company and a New Hire Experience for Computer Science Students Preparing to Enter the Software Industry

David A. Chamberlain

Bridgewater State University

Author Note

Submitted in partial fulfillment of a Master of Science in Computer Science

Acknowledgements

I would like to express my special appreciation and thanks to my advisor, Dr. Michael Black, not only have you been a wonderful mentor for me but also my friend. Your constant encouragement and guidance has been invaluable. I would also like to thank my thesis committee, Drs. Michael Black, John Santore and Haleh Khojasteh for serving as my committee members and putting up with my last-minute changes. I would especially like to thank the many undergraduate and graduate students of Bridgewater State University for offering me feedback and asking the many questions you had, and all of the industry professionals who helped shape the thesis direction. A special thanks to my family and friends, words cannot express how grateful I am to my wife for all of the sacrifices that you've made on my behalf. To my sons Dan and Doug, I would like to thank you for being so incredibly supportive and understanding, and to my longtime college friends Stu and Ed for loaning the use of a bedroom to help me save on my commute time. You all have earned this degree along with me and I wish to thank you all.

Table of Contents

Abstract

Students in a Computer Science degree programs must learn to code before they can be taught Software Engineering skills. This core skill set is *how to program* and consists of the constructs of various languages, how to create short programs or applications, independent assignments, and arrive at solutions that utilize the skills being covered in the language for that course (Chatley & Field, 2017). As an upperclassman, students will often be allowed to apply these skills in newer ways and have the opportunity to work on longer, more involved assignments although frequently still independent or in small groups of two to three students. Once these students graduate and enter the software industry they will find that most companies follow specific development methodologies from one of the many forms of Agile through Waterfall. All while working in large groups or teams where each developer is responsible for specific pieces of the functionality, participating in design meetings and code reviews, as well as using code versioning systems, such as git, a program management system, such as Jira, all in a very collaborative environment. This study will develop a course that will allow students to apply these skills in a more realistic setting while remaining on-campus and monitoring the students' beliefs on their preparedness for the world outside of the computer science building.

Keywords: internship, cooperative education, computer science

A Semester Long Classroom Course Mimicking a Software Company and a New Hire Experience for Computer Science Students Preparing to Enter the Software Industry

Long before modern computers and software development, John Dewey (1922) described a classroom with groups of students working collaboratively with the instructor as merely a guide. After initial instruction and skills are presented, this style of cooperative learning is intended to enhance that prior knowledge through group discourse, cooperative problem solving, and continued practice. Lev Vygotsky (1962, p 188) furthered that notion stating that what the learner is able to do in collaboration today, they will be able to do independently tomorrow because, or as Dewey (1922) states it: "An ounce of experience is better than a ton of theory…". Both Dewey and Vygotsky, two pioneers in educational philosophy and still studied today, were referring to continued learning beyond classroom lectures.

In 1956, Benjamin Bloom published what is commonly referred to as Bloom's Taxonomy, which discusses learning from low-level information such as terminology through higher-order thinking skills in order to create and evaluate (Bloom, Englehart, Hill, & Krathwohl, 1956) (Gardner & Motschenbacher, 1993). Where one begins learning a subject with the terminology used and advancing through various levels or

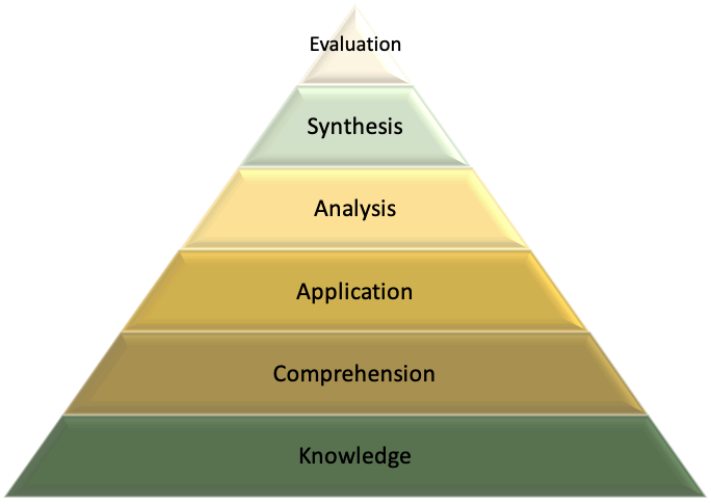


Figure 1 Bloom's Taxonomy

Depths of Knowledge (DoK) progressing from a basic knowledge through evaluating one's own work or

Figure 2 Progression through the Depths of Knowledge

ability to critique work.

Hiring companies, in the Computer Science domain space, are looking for graduates that have a working knowledge of the development lifecycle and methodologies, the ability to analyze various approaches, write effect code, and participate in code review processes, let alone using current industry *tools of the trade*. Modern Computer Science programs must help prepare a student and bring a student through all of the depths of knowledge to a point where the learner meets the expectations of the hiring

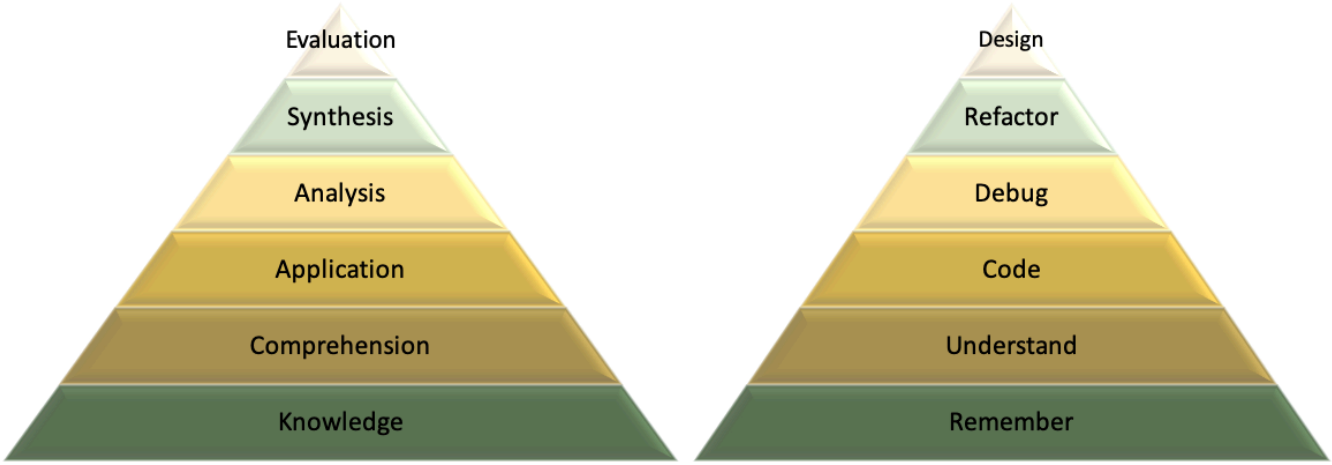


Figure 3 Depth of Knowledge in Computer Science

companies. Applying the pyramid created by Bloom's Taxonomy to the DoK in various computer science topics allows a visual for each level. In interviews with current industry professionals, graduating undergraduate students are able to debug and may be able to go further but at a minimum

will be able to debug their own code while graduate students have the expectation that they will be able to apply more abstract thinking and participate at higher levels.

In essence: *Given the limited classroom time and inexperience of the student, how can an internship-like course be offered during a single undergraduate semester?*

Background

When graduating from college, many students from a variety of different disciplines have gone through a clinical field-work portion of their college degree program, often referred to in terms such as practicum, internship, corporative education model, to name a few. According to Price (1987) the practicum is intended to link "theory with practice by providing regular structured and supervised opportunities for students to apply and test knowledge, skills and attitudes, developed largely in campus-based studies, to the real world …." Similarly, Liu, Xu, and Weitz (2011) refer to an internship as a form of experiential learning which allows students valuable opportunities to apply classroom knowledge to practice and prepare themselves for their entry into a competitive job market as they will now have essential experience. While the definitions may appear, similar there are often hidden distinctions such as paid or unpaid, or whether or not under direct supervision (Gardner & Bartkus, 2014).

There is likely no argument that says that professionals should be ill-prepared to enter the work-force upon graduating from a college or university. Therefore, the task of preparing students must be the responsibility of these colleges and universities (Goodlad, 1984 as cited by Walker & Wimmer, 2008). In order to prepare professionals, the outcomes of a practicum or internship should be authentic learning which is defined as the application of relevant knowledge, thinking, and interpersonal skills to arrive at solutions to real world problems (Renzulli, Gentry, & Reis, 2004). Problems are said to be *real world* if they have the following four characteristics:

1. They have personal frame of reference for the person or group solving the problem
2. They do not have a current or existing solution
3. They provide motivation to the person or group working toward a solution such that they will be contributing to a new or existing product
4. They are directed toward a real audience

Without these four characteristics being present, it is likely that the student working toward a solution will classify this problem merely as a training exercise and therefore, authentic learning is unlikely to occur (Renzulli, Gentry, & Reis, 2004).

Internship and Cooperative Education

> I don't know what you mean by 'glory,' " Alice said.
> Humpty Dumpty smiled contemptuously. "Of course you don't—till I tell you. I meant 'there's a nice knock-down argument for you!' ”
> "But 'glory' doesn't mean 'a nice knock-down argument'," Alice objected.
> "When I use a word," Humpty Dumpty said, in rather a scornful tone, "it means just what I choose it to mean—neither more nor less.
>
> (Carroll, 1900)

According to the Agile Manifesto, the main source of truth for the Agile Development Process: “The best architectures, requirements, and designs emerge from self-organizing teams.” (Beck, et al., 2001) This translates to students needing to learn to work in collaborative groups or teams.

The idea of cooperative learning, which utilizes social skills and group work to organize activities (May & Doob, 1937) and acquire social skills (Gillies, 2016) began in the mid-1960s according to Johnson and Johnson (as cited in Johnson, Johnson, & Smith, 2007) and studies that were conducted since have made a comparison between cooperative, competitive, and individualistic learning

models; results showing that students who participated in cooperative learning environments achieved a higher level of individual achievement than other learning models (Johnson, Johnson, & Smith, 2007). Here, the term *cooperative* is seen as a group learning or group work exercise to provide additional learning above what one may learn by oneself; a useful skill in the global world of the computer sciences where one may find themselves on a team partially made up of those working in-office, those working from home, and remote teams potentially in a different timezone. For example, a single distributed team may be working on a common components all from various regions around the world (Crampton & Webber, 2005) thus, experience with group and teamwork, particularly in computer programming and software engineering, prior to entering the professional software industry can be beneficial to all members of the group.

However, according to Barker (2005), "students generally lacked any understanding of how to work well in groups, students selected their roles based on expediency or familiarity" which "worked against the benefits of collaborative learning and the learning of new skills or concepts." Generally speaking, care must be taken to encourage both equal learning and prepare students for real-life collaborative teams.

> Ideal student work groups are heterogeneous and their group tasks require that all students share their knowledge and expertise as well as their questions and uncertainties in ways that lead to peer learning. However, this ideal assumes that students take equal responsibility for the roles of teacher and student; and that tasks focus on learning through dialogue and hands-on activities. (Barker, 2005)

Therefore, students must have clear instructions on the expectations of working in a group, which includes moving through roles and open communication.

Moving learning from the classroom environment into a work environment where the learning is not accredited and includes what happens organically from actual experiences through "specific work issues, as a result of workplace training or coaching, or to further work-related aspirations and interests…and overlaps with, but is not the same as, experiential learning" can be defined as work-based learning (Lester & Costley, 2010). This work-based experiential learning goes by many names including the internship or cooperative learning model.

Not to be confused with the cooperative learning model, the framework for the cooperative education model, or co-op, dates back to 1905 at the University of Cincinnati. After initial resistance, the university tested a theory designed to form a connection between academic knowledge in engineering and real world work in a model defined by Herman Schneider, a Lehigh University graduate, and has since been employed by more and more universities, where this model can be seen at universities such as Northeastern, which made the commitment to cooperative education by moving college after college to this model (Smollins, 1999). This program, intended to "bridge the gap between theory and practice in engineering education" was introduced to Canadian universities in the 1950s and met with equal resistance but soon served as models for other higher learning institutions (Haddara & Skanes, 2007). According to Reinhard and Pogrzeba (2016), the modern cooperative education model is constructed through an agreement between an industry company known as the "sponsoring company" and the university where students will work in a professional environment for period of time while continuing their coursework, this general model typically has two main implementations:

1) Where the student works full-time for the sponsoring company and attends courses on alternating semesters or terms, and
2) Where the workday at the sponsoring company is intended to be during days or at times when the student does not have to attend class

Like cooperative education experiences, the internship, a term often used interchangeably, involves the student working in a position related to their degree as a temporary employee (Schambach & Dirks, 2002). The definition typically differs between the internship and the co-op in that an internship is often set up by a company and the student may or may not receive credit for participating in an employer's internship program. Essentially "The lack of clearly defined and generally accepted guidelines, such as found in more formally established programs such as cooperative education, presents a challenge for internship program administrators" (Gardner & Bartkus, 2014).

Whether it is a co-op opportunity or that of an internship, the benefits go both to the learner and the sponsoring company a.k.a. the employer. The employer may look at the student's time with the company as an elongated interview where they can see the student learning and applying their skills while evaluating the student for future employment. When the student is ready to graduate, the employer may offer a position within the company. For the student, they are given the opportunity to apply the learned skills in new ways which can lead toward mastery (Huggins, 2009). This mastery applies to broad skills, such as general coding and problem-solving, as well as specialized areas of knowledge as in cloud-based software, user interface (UI), and back end services or databases. Without this real-world application, one may have difficulty applying newly acquired knowledge to new situations outside of academic assignments (Eyler, 2009). Referring back to Bloom (1956), this is not surprising as first knowledge must be acquired and understood before it can be applied. In other words, recently acquired knowledge serves as the foundation to further understanding, but does not imply a full understanding, nor does it lead directly to the ability to apply the skill, evaluate it for the current situation, or compare other possible approaches to the problem (Bloom, Englehart, Hill, & Krathwohl, 1956). Eyler (2009) reiterates this explaining that: "Recall and reproduction of material taught in the classroom do not

constitute understanding … to be usable, it has to be acquired in a situation. Otherwise, it is segregated from experience and unlikely to be remembered or transferred to new experiences."

Martin Trow, in *Thoughts on the White Paper of 1991*, comments that "education is a process pretending to be an outcome" (as cited in Little & Brennan, 1996) meaning that learning is a life-long process and education prepares you, but an individual's education is never complete. Therefore, one has not learned all that they need when they have received a degree and ready to enter a professional work environment thus necessitating the need for the type of cooperative eduction or internship being discussed here.

Engineering and Computer Science Specifics

Dating at least as far back as 1976, the issue of computer science education, combined with working in the industry or at least working on more realistic problems, was recognized by Buck & Shneiderman (1976) as they stated that "Computer science graduates who become professional programmers will have a direct and substantial influence on the impact of applications, but little in traditional computer science training curriculum prepares them for this serious responsibility." Just under twenty years later, a study conducted at Michigan State University reported that, in a sample of 600 graduates, 79% of the students had some form of pre-workforce experience (Co-op, summer employment, and internships) in the engineering field prior to entering the workforce post-graduation (Gardner & Motschenbacher, 1993).

Even with previous studies showing the value of real-world application of learned skills, we find that it is still the case that graduating computer science and software engineering students are lacking skills or having an inability to properly apply them to real-world situations. Similarly, students often lack the non-programming, or soft-skills, such as communications, knowledge of current development

tools, or general knowledge of software development lifecycle (SDLC) processes, all of which are needed in order to function as a productive member of a software engineering team (Radermacher, Walia, & Knudson, 2014).

Why is there a skill gap or Why isn't This Learned in the Classroom?

As the evidence from past research shows, there are obviously skill gaps that can be bridged through one of the work-based learning methods. However, not all colleges or universities are able to provide co-ops and therefore these skills must be learned in the classroom. Computer science skill gaps are likely to exist due to the shape the computing industry has taken as it has become a very broad member of the sciences (The Joint Task Force on Computing Curricula, 2015) with more specialty areas to cover over the same duration. Furthermore, the research has shown that it is likely many of these gaps are due to the majority of courses being taught by academics, who are primarily researchers, and not often by current industry professionals. Many of these academics have little experience developing software in a modern industrial environment. While computer science faculty do often write software, the way in which they do so differs from than the way it is done in the industry using approaches and methodologies from their education, which may not have kept current with the software industry (Chatley & Field, 2017) – "We need to prepare kids for their future, not our past" (Pink, 2006).

These faculty projects are often small, may not be intended for the mass market, were created by a single developer not a group, thus creating a deficiency in experience with the material being taught (Chatley & Field, 2017). Industries change rapidly, making experience from earlier in one's career less relevant today, this is one cause that leads to the skill gap between graduating students and industry expectations (Noga & Rhoades-Catanach, 2014). In a direct correlation to the benefits of the student internship benefiting the student for entry into the workplace, faculty internships can be beneficial to

full-time faculty to expand their knowledge of the subjects they teach and find real-world examples to bring back to the classroom. In some cases, these faculty internships provide professors, who are primarily in the traditional teaching role, with their only source of work experience (Lohman, Austin, Borgen, & Salo Wolff, 2015). In fact, one is likely to find that today's software engineering tasks performed in the industry are very similar to those from several decades ago but in a different context. New and more powerful tools and languages to use, interest and input from various parts of the organization (The Joint Task Force on Computing Curricula, 2015).

In a study conducted by Radermacher, Walia, & Knudson (2014) the top six deficiences identified, through a series of interviews with various hiring managers, of graudating students about to enter the job market are:

- Project Experience,
- Configuration management tools,
- effective oral communication,
- problem solving abilities,
- understanding job expectations, and
- software testing.

This was echoed through interviews with software industry professionals in the Boston area and included items such as: How to interview and conduct ones self during the interview (Boran, 2017), No point of reference for software development methodologies, how to dissect a problem (Blackburn, 2017), Recent graduates often feel that all they need is to attend class and do an assignment, they don't exhibit the passion or commitment (Anuszczyk, 2018).

Much of the previously cited literature indicates the benefits of an internship, co-operative education, or other form of work-based learning. This is engrained and expected in some educational

careers such as teaching and the medical field. Yet, other programs, such as Computer Science often do not have it built into their curriculum and may leave it to the student to find a position and coordinate with the college or university to receive credit or work with an advisor. Sometimes these schools are in areas that may not have many local employers to work with and when going to larger cities, those employers are often working with other schools. Graduates from these schools will surely be competing for positions with candidates who attended school that did offer work-based learning as part of its curriculum; this poses the question, and the impedus for this research: How can a school without a formal internship or co-operative education program help close those skill gaps to help their students better succeed?

Research

Bridgewater State University (BSU), founded in 1840, is in a suburban community approximately 25 miles south of Boston. Of the 114 institutions of higher-learning in Massachusetts, BSU ranks as the 10th largest (Bridgewater State University, 2019). During the Spring Semester of 2018 a course was offered, entitled *Industry Oriented Software* ("Course meets on campus Thursdays; remainder online. Students will work in Scrum teams to create a cloud service that will act as a backend server for a game.") (Bridgewater State University, 2018) and was open to all matriculated students after completing their *Data Structures and Algorithms* course and all prerequisite coursework. This course was designed to build the backend of a game system in the cloud, and be produced by various Scrum teams comprised of students and guided by the instructor. The course curriculum is included in Appendix A.

Course Delivery

The course was designed to start off as though the students were a group of newly hired graduates at a company. As new employees, they would need to begin by configuring their computers and installing the appropriate software. This was done as a checklist of items to install and how to test. The documentation was available online and students were given the ability to adjust or add notes as needed to assist other students with any issues they encountered.

This delivery method continued throughout the semester. The students were divided into Scrum Teams mixing students with different amounts of experience together as you may find at a company as opposed to allowing students to choose their groups. Each class meeting, during a two-week Sprint, began with a Daily Scrum meeting to talk about accomplishments or blockers for each student while the instructor participated as Scrum Master and Product Owner for each group. As the semester progressed,

additional tools, such as Jira, were introduced to track User Stories and Technical Tasks and assigned to members to facilitate ownership or remain in the Sprint Backlog to be picked-up when tasks were completed.

Slack was a required tool for communication outside of the scheduled meeting times as well as GitHub. Students were somewhat familiar with Git, however not in a shared codebase. Where the scheduled class meeting time was set for one day per week with a second meeting to be web-based, Zoom and shared team Google Calendars were used to schedule smaller team meetings which could be attended remotely.

A long running assignment was given, which was to create the backend service for a game. This service was defined prior to the class with a RESTful API and service-based architecture. Scrum Teams were to implement the code, following the design given dividing the work into smaller stories which could be worked independently but would require integration amongst the team members as well as common code. This introduced a more realistic scenario into the curriculum, that of shared code and the introduction of merge conflicts to the common codebase.

The services were to be developed using PyCharm as the Integrated Development Environment (IDE), Python 3.6 which required a Virtual Environment, and the Django framework to handle a database that was shared amongst the services. The various services were to have the ability to be deployed in Amazon Web Services (AWS) Cloud, locally on their laptops through Virtual Box and an Ubuntu installation, or in combination with their laptop joining the service cluster in AWS Cloud. The shared database for the services was SQLite when running locally and AWS Relational Database Services (RDS) when running in the cloud thus requiring configuration files that could be overridden for running in a local or development environment.

Additional Agile Ceremonies were added such as Sprint Planning, Grooming, Kickoff, and the Sprint Retrospective as the semester progressed as well as allowing the teams to do more self-management and self-organization. Likewise, additional topics were included such as:

1. Interviewing
    a. How to Research a Company and Prepare for the Interview
    b. Collaborative Interview Process
    c. Sample Interview Questions
2. Various Development Roles, Guest Speakers from the Industry were Invited
    a. UX/UI
    b. Architecture
    c. Front-end
    d. Cloud Developer
    e. Backend
    f. Real-time Software
    g. Quality Assurance (QA), Software QA (SQA), Software Development Engineer in Test (SDET)
3. Methodologies
    a. Agile
        i. Values
        ii. Pillars
    b. Waterfall
    c. Scrum vs Kanban

Throughout the course, reading assignments were given as well as tutorials and research topics were assigned around tools and trends that are currently going on in the industry.

The Assignment

These are four of the basic elements needed by a generic game service, the ability to:

- Allow players to register and authenticate
- Allow players to create, update, and retrieve profile information
- Allow for scores and other game attributes be stored, updated, and retrieved for each player and game
- Allow for content to be shared (such as player inventory data)

During the development of the services, discussions were had regarding Application Programming Interface (API) styles such as Representational State Transfer (RESTful), CRUD (Create, Read, Update, Delete) aspects of the proper verbs being used, such as a POST to register a new use and a PUT to update information.

Data Collection and Analysis

Anonymous surveys were given several times throughout the semester with data collected and analyzed to look for student beliefs in how they are progressing and their knowledge of the subject areas beyond the academic approaches taken to initially learn the topics but by applying additional practical experience in a "work-like" environment. Standard assessments administered during the course of the semester to more concretely measure their understanding and communication about subjects was used to inform instruction and identify areas where additional experience was needed and applied to further measure the student DoK in the various subject areas. Due to surveys being anonymous, individual perception of their progress was not monitored, instead it is being applied as a group.

Assessments used to measure individual DoK was administered as a series of questions, some were matching or identifying what category given items were members of (DoK-1, Ability to Identify) ranging through Analysis (DoK-4) which asked for students to compare and make a decision. These assessments were given periodically throughout the semester to help inform instruction, not applied toward any grading criteria, and class progress and identify areas where individual students may need additional instruction or experience. This allowed for individualized instruction where needed and tailoring of lessons, assignments, and discussions to focus on deficient areas.

q = number of questions in the assessment

n = number of participants

t = total number of responses examined

p = percentage of students responding at that particular level

Question	**Initial Assessment**					**Final Assessment**				
Understanding of Agile Process	DoK1	q=5	n=16	t=80	p=10%	DoK1	q=4	n=16	t=64	p=86%
	DoK2	q=3	n=16	t=48	p=6%	DoK2	q=4	n=16	t=64	p=81%
	DoK3	q=2	n=16	t=32	p=0%	DoK3	q=3	n=16	t=48	p=70%
Application of Development Tools and Their Usage	DoK1	q=3	n=16	t=48	p=63%	DoK1	q=3	n=16	t=48	p=96%
	DoK2	q=3	n=16	t=48	p=56%	DoK2	q=3	n=16	t=48	p=88%
	DoK3	q=2	n=16	t=32	p=38%	DoK3	q=2	n=16	t=32	p=88%
Efficient Use of Linux and Git Commands & Concepts	DoK1	q=10	n=16	t=160	p=56%	DoK1	q=8	n=16	t=128	p=78%
	DoK2	q=5	n=16	t=80	p=48%	DoK2	q=5	n=16	t=80	p=72%
	DoK3	q=10	n=16	t=160	p=20%	DoK3	q=8	n=16	t=128	p=71%
	DoK4	q=4	n=16	t=64	p=13%	DoK4	q=3	n=16	t=48	p=64%

In most cases, it was obvious that students did indeed have academic knowledge (DoK-1) of most subject areas prior to taking this course, however the amount of application (DoK-3) level knowledge that students had, or the ability to adopt or evaluate tools or commands based on specific circumstances (DoK-4) was lacking and correlates to their own perception of their readiness to enter the work-force. After taking this course, however, improvement in both the student perception of their knowledge and their ability to apply and evaluate (DoK-3 and -4) had risen substantially.

Open Ended Questions and Typical Responses

Two open-ended questions of note, "When my program has a bug, I…" and "I think I need to know about…" also showed growth in student perception of their knowledge-base. When posed with a question about a program with a bug, prior to this course, typical student responses were non-specific in what they could do with resorting to staring at the code for further inspiration or adding print statements to trace the code. When compared to the responses at the end of the course, we can see the introduction of additional debugging methods from their bag-of-tricks including using the debugger, the use of log statements, reaching external resources and recreating the error condition.

When my program has a bug, I …

Initial Survey: • Set flags in my code • Read and try to solve it by myself first, and then try Google • I reread the program. I put in code to see if it executes, meaning that the program reached a certain section • Run program with print markers, check that individual functions are working correctly, compare with stack overflow • Read error code to see if it is a simple fix, search for any typos, then attempt to read through.
Final Survey: • Panic, check StackOverflow religiously, relax, and slowly debug the issue • Debug using debugging tools • Check error log • First try to find where in my program the bug is occurring. I then try to figure out what is causing the bug to occur. Once that is diagnosed, I start brainstorming solutions that may fix the bug and implement them. Use a debugger to help diagnose the cause of the bugs. • Try to recreate the bug, follow the stack trace.

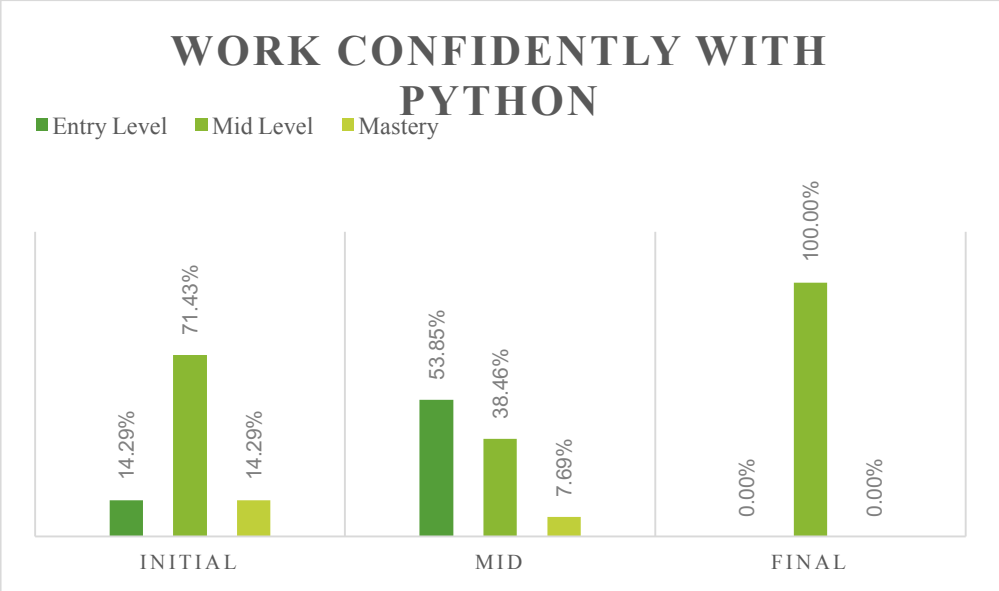
In a second open-ended question giving the student the opportunity to speculate on what they need to know; prior to the course, a typical response as fairly generic and revolved more around continued experience with coding. The end-of-course survey showed students seeking information about more specific aspects of the software development team and the hunt for deeper information about specific subject areas.

I think I need to know about …

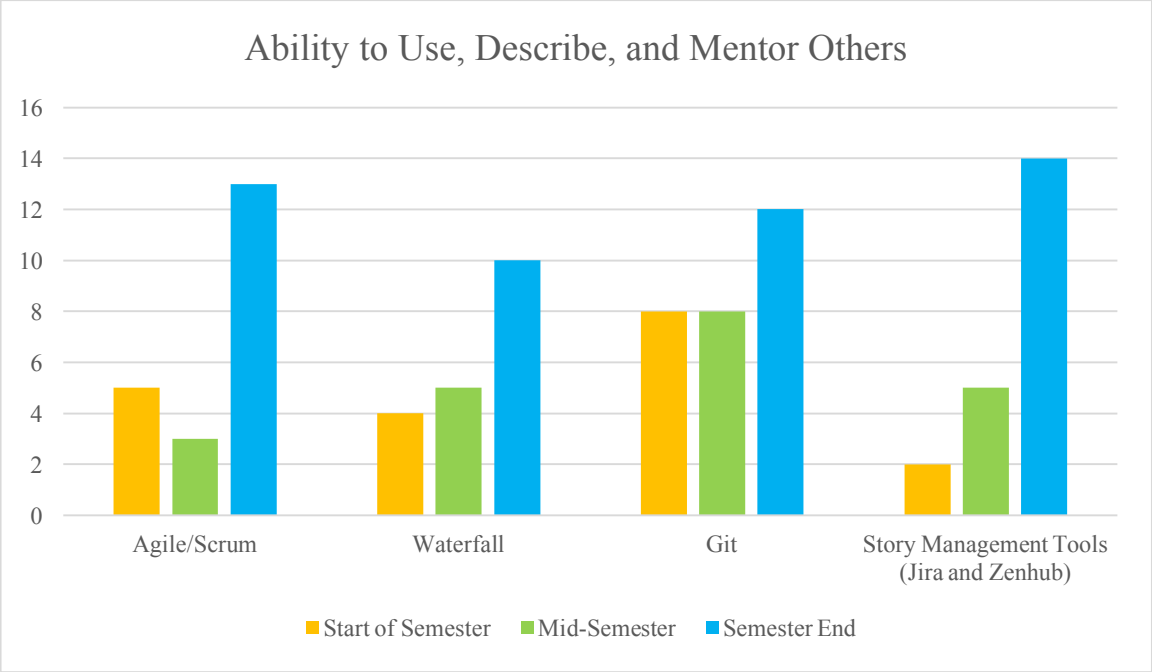
Initial Survey: • Coding can be done within assignments, but I want more on-trend coding topics in the industry • Getting a career in this field. Real world skills. • Job expectations, interviewing. • A lot of programming • I don't know what I don't know
Final Survey: • More potential interview questions • Integration Tools • Technical Business Analysis • Software Quality Assurance • Software Design

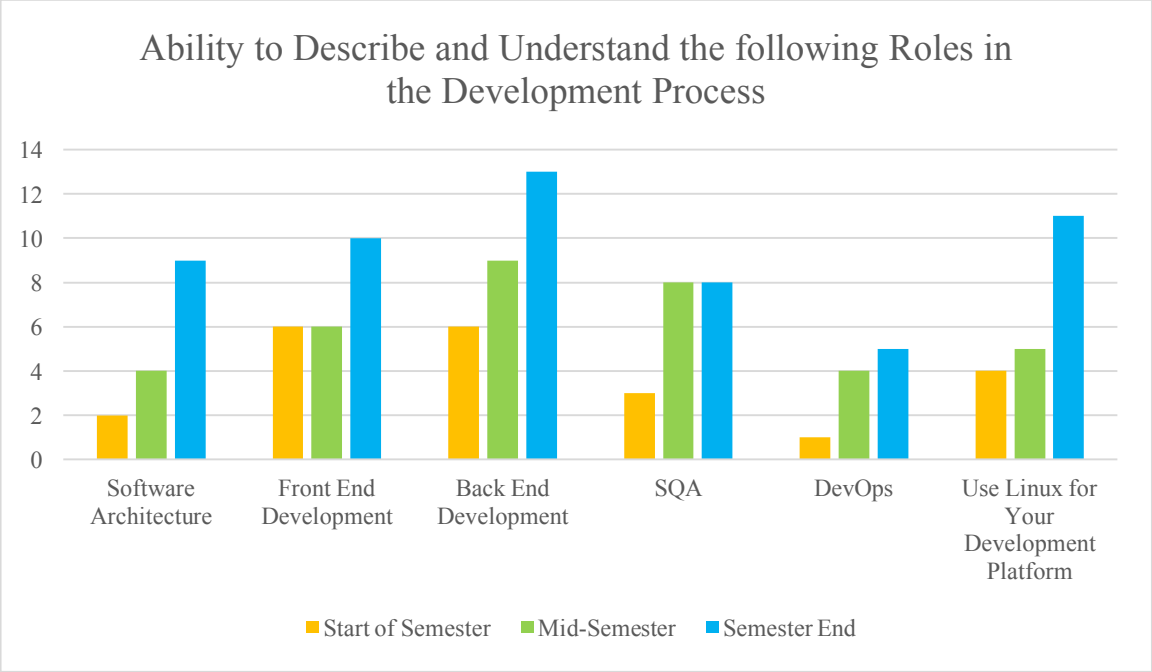
Additional survey questions were given, among them the more notable ones included the student rating themselves on various scales to determine if they felt they were a entry-, mid-, or expert-level in a variety of subject areas. One such question involved their confidence level in using Python as their primary language. This question was given three times throughout the semester. When the semester began, a few students can be seen who believe they are an entry level Python user or believe they are at the expert level. As the project continued at the mid-semester timeframe students changed their view on where they stood with Python as their primary language with some realizing they may not be as confident as they once thought. However, by the end of the semester we no longer see any student

holding onto the belief that they are a novice, nor do we see any students believing they are an expert user. Instead we see all students placing themselves as a mid-level Python developer.

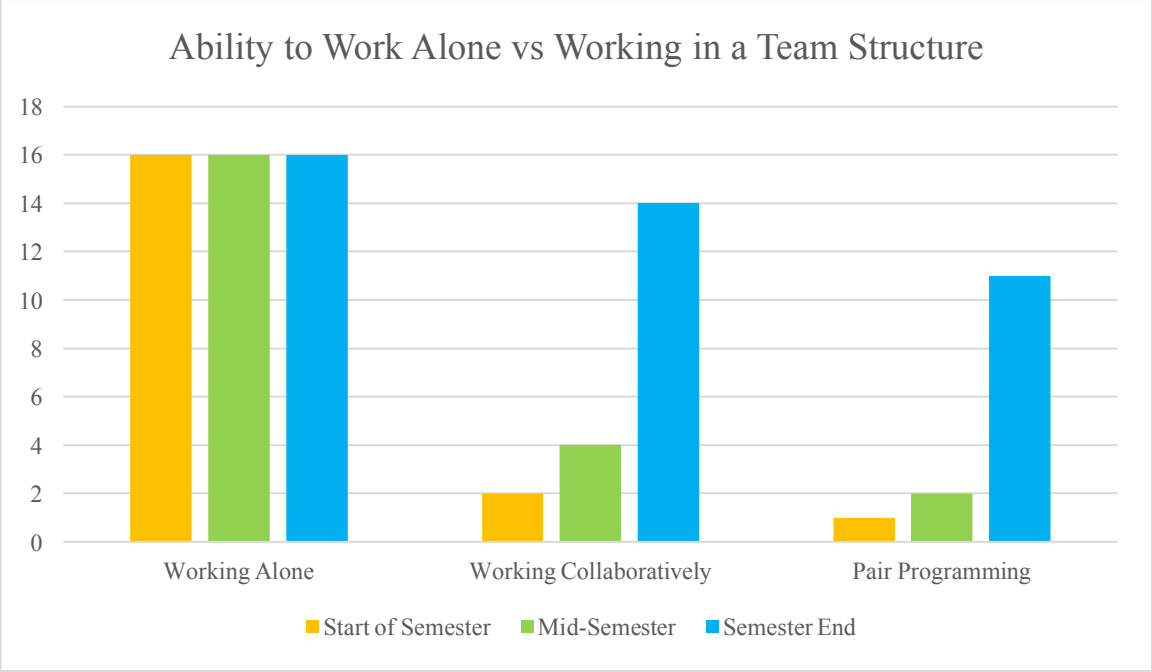


Using a different scale, and showing only those responding within the top categories, it can be seen here that Software Development Life Cycle (SDLC) models for both Agile and Waterfall improved with more than double the number of students placing themselves in the top categories between the beginning and end of the semester. Likewise, the self-rating on the knowledge of Git and User Stories & Product Management Tools also went up along with their knowledge of various disciplines within a software product.


Ability to Use, Describe, and Mentor Others
16
14
12
10
8
6
4
2
0
Agile/Scrum
Waterfall
Git
Story Management Tools (Jira and Zenhub)
Start of Semester
Mid-Semester
Semester End


Ability to Describe and Understand the following Roles in the Development Process
14
12
10
8
6
4
2
0
Software Architecture
Front End Development
Back End Development
SQA
DevOps
Use Linux for Your Development Platform
Start of Semester
Mid-Semester
Semester End

Finally, we see one of the larger distinctions between what is normally done as an assignment and what happens in the work place, is that of the *team*. It is clear that all students found that they knew how to work by themselves or in a group doing an individual piece. However, their ability to work collaboratively or in a Pair Programming activity was severely lacking and out of alignment with current software development best practices.



Conclusion

The research for this thesis involved the creation of a course based on modern software industry trends, the tools that are used, and how teams function within a software company. While the course was offered on-campus and was not a product that was ultimately deliverable and therefore there were not real "stakes", it was able to advance the students' knowledge in key areas and provided them with the knowledge that what they are learning is relevant and how it can be applied in real-world scenarios.

Students are given a solid foundation on which to build. However, the Software Industry is looking for more than just a foundation, they are looking for graduates who have a base knowledge, if not experience, with modern software development processes and a core understanding of software engineering principles that can be applied in any language. Having a knowledge of a specific language is nice, but within any company the chosen computer language will change, often on a project-by-project basis, while the fundamental software engineering processes will not.

Not all colleges and universities are able to offer internships to everyone due to many factors, including their physical location or the number of local institutions of higher learning for that area causing competition for the limited number of internship openings.

Suggestions

It is the responsibility of the college to know what the market is expecting and be able to adapt to that changing market. An incoming freshman student will not know what to look for or what they need to improve upon to enter the market, knowing this is a burden that must be placed on the faculty of that university. There are options available:

- Offer an advanced course, under the guidance of a faculty member with intimate knowledge of the current software industry, similar to the one presented here.

- Incorporating industry trends across the curriculum in a consistent manner such that it can be introduced in any course and continued throughout. An example may be to form Scrum Teams in development courses where, at a minimum, students can share their successes and blockers with their team. This, of course, would require Agile training for faculty.
- Including guest speakers from the software industry in classes. This does not need to be done as in-person guests, professionals can join by video conference.
- Inviting industry professionals to work with faculty in a variety of capacities from mentoring to discussing “day in a life” moments at work. This would also give faculty the opportunity to ask questions which they may have been asked either about what it is like in the industry or for concrete examples of newer trends.

Works Cited

Anuszczyk, J. (2018, March 4). Executive Director of Architecture, CVS Health. (D. Chamberlain, Interviewer)

Barker, L. J. (2005). When do Group Projects Widen the Student Experience Gap? *ITiCSE '05 Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, 276-280.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *The Agile Manifesto*. Retrieved December 20, 2018, from The Agile Manifesto: https://agilemanifesto.org/principles.html

Blackburn, C. (2017, August 3). Senior Software Engineer, edX/MIT. (D. Chamberlain, Interviewer)

Bloom, B., Englehart, M., Hill, W., & Krathwohl, D. (1956). *Taxonomy of Educational Objectives: The classification of educational goals. Handbook I: Cognitivie domain.* New York; Toronto: Longmans, Green.

Boran, C. (2017, August 17). Distinguished Architect, Sophos Ltd. (D. Chamberlain, Interviewer)

Bridgewater State University. (2018). *Class Schedule Listening*. Retrieved January 18, 2019, from Infobear: Bridgewater State University : https://infobear.bridgew.edu/BANP/bwckschd.p_get_crse_unsec

Bridgewater State University. (2019). *About Us*. Retrieved 2 16, 2019, from Bridgewater State University 2019 Catalog: https://catalog.bridgew.edu/content.php?catoid=13&navoid=1249

Buck, J., & Shneiderman, B. (1976, September). An internship in information systems: Combining computer science education with realistic problems. *SIGCSE '76 Proceedings of the sixth SIGCSE technical symposium on Computer science education, 8*(3), 80-83.

Carroll, L. (1900). *Through the looking-glass.* Retrieved 11 16, 2018, from The Project Gutenberg: https://www.gutenberg.org/files/12/12-h/12-h.htm

Chatley, R., & Field, T. (2017). Lean learning: applying lean techniques to improve software engineering education. *ICSE-SEET '17 Proceedings of teh 39th International Conference on Software Engineering: Software Engineering and Education Track* (pp. 117-126). Piscataway, NJ: IEEE Press.

Chillas, S., Marks, A., & Galloway, L. (2015, March). Learning to labour: an evaluation of internships and employability in the ICT sector . *New Technology, Work and Employment, 30*(1), 1-15.

Crampton, C. D., & Webber, S. S. (2005). Relationships among geographic dispersion, team processes, and effectiveness in software development work teams. *Journal of Business Research, 58*(6), 758-765.

Dewey, J. (1922). *Democracy and Education: an introduction to the philosophy of education.* New York: The Macmillan Company.

Eyler, J. (2009). The power of experiential education. *Liberal Education, 95*(4), 24-31.

Gardner, P. D., & Motschenbacher, G. (1993). *More Alike Than Different: Early Work Experiences of Co-op and Non Co-op Engineers.* Michigan State University. East Lansing: Michigan Council for Cooperative Education.

Gardner, P., & Bartkus, K. R. (2014). What's in a Name? A Reference Guide to Work-Education Experiences. *Asia-Pacific Journal of Cooperative Education, 15*, 37-54.

Gillies, R. (2016). Cooperative Learning: Review of Research and Practice. *Austrailian Journal of Teacher Education, 41*(3), 39-51.

Haddara, M., & Skanes, H. (2007). A reflection on cooperative education: From experience to experiential learning. *Asia-Pacific Journal of Cooperative Education, 8*(1), 67-76.

Huggins, J. K. (2009). Engaging Computer Science Students Through Cooperative Education. *SIGCSE Bulletin, 41*(4), 90-94.

Johnson, D. W., Johnson, R. T., & Smith, K. (2007, March). The State of Cooperative Learning in Postsecondary and Professional Settings. *Educational Psychology Review, 19*(1), 15-29.

Lester, S., & Costley, C. (2010). Work-based learning at higher education level: Value,practice and critique. *Studies in Higher Education, 35*(5), 561-575.

Little, B., & Brennan, J. (1996). *A Review of Work Based learning in Higher Education.* The Open University.

Liu, Y., xu, J., & Weitz, B. A. (2011). The Role of Emotional Expression and Mentoring in Internship Learning. *Academy of Management Learning & Education, 10*(1), 94-110.

Lohman, L., Austin, E., Borgen, B., & Salo Wolff, S. (2015, September 1). Business Perspectives on Faculty Internships: Not Just for Students Anymore? *Journal for Advancement of Marketing Education, 23*(2).

May, M. A., & Doob, L. W. (1937). Competition and Cooperation. *25*.

Noga, T. J., & Rhoades-Catanach, S. C. (2014, February 1). Faculty Internships: Connecting Tax Practice and Academia. (A. Nellen, Ed.) *The Tax Adviser*.

Pink, D. H. (2006). *A whole new mind: Why right-brainers will rule the future.* New York, NY: Riverhead.

Price, D. A. (1987). The Practicium and its Supervision. (K. J. Eltis, Ed.) *Australian Teacher Education in Review*.

Radermacher, A., & Walia, G. (2013, March). Gaps between indutry expectations and the abilities of graduates. *SIGCSE '13 Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 525-530). Denver: ACM.

Radermacher, A., Walia, G., & Knudson, D. (2014, June 7). Investigating the skill gap between graduating students and industry expectations. *ICSE Companion 2014 Companion Proceedings of the 36th International Conference on Software Engineering*, 291-300.

Reinhard, K., & Pogrzeba, A. (2016). Comparative Cooperative Education: Evaluating Thai Models on Work-Integrated Learning, Using the German Duale Hochschule Baden-Wuerttemberg Model as a Benchmark. *Asia-Pacific Journal of Cooperative Education, 17*(3), 227-247.

Renzulli, J. S., Gentry, M., & Reis, S. M. (2004). A Time and A Place for Authentic High-End Learning. *Educational Leadership, 62*(1), 73-77.

Sanderson, P. (2003, October). Where's (the) Computer Science in Service-Learning? *Journal of Computing Sciences in Colleges, 19*(1), 83-89.

Schambach, T. P., & Dirks, J. (2002). Student Perceptions of Internship Experiences. *Proceedings of the 17th Annual Conference of the International Academy for Information Management*.

Smollins, J.-P. (1999, May). The Making of the History: Ninety Years of Northeastern Co-op". *Northeastern University Magazine, 24*(5).

Tan, J., & Phillips, J. (2005). Incorporating Service Learning into Computer Science Courses. *Journal of Computing Sciences in Colleges, 20*(4), 57-62.

The Joint Task Force on Computing Curricula. (2015). *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.* Technical Report, New York, N.Y.

Vygotsky, L. S. (1962). *Thought and Language.* (E. Hanfmann, G. Vakar, Eds., E. Hanfmann, & G. Vakar, Trans.) Cambridge, MA: MIT Press (Original source published 1934).

Walker, R. E., & Wimmer, R. (2008). The Clinical/Practicum Experience in Professional Preparation: Preliminary findings. *McGill Journal of Education, 43*(2), 157-172.

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education, 12*(3), 197-212.

Appendix A

ABET Accreditation Criteria

Throughout the course, the following accreditation criteria from the Accreditation Board for Engineering and Technology (ABET) will be addressed throughout the course as noted in individual lessons.

The program must enable students to attain, by the time of graduation:

a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline

b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

d) An ability to function effectively on teams to accomplish a common goal

e) An understanding of professional, ethical, legal, security and social issues and responsibilities

f) An ability to communicate effectively with a range of audiences

g) An ability to analyze the local and global impact of computing on individuals, organizations, and society

h) Recognition of the need for and an ability to engage in continuing professional development

i) An ability to use current techniques, skills, and tools necessary for computing practice

j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.

k) An ability to apply design and development principles in the construction of software systems of varying complexity.

Semester Assignment

Create the backend for a game service. The service must allow a game to add and remove users, maintain scores, keep a history of usage (2 days), the user contact information and level, and an inventory of what the user has earned or found. It must allow an administrator to disable and re-enable a user, get a report of high scores, usage information, and time spent in the game. It should also allow an admin to wipe the game of all data.

You do not need to write a game, but you will need to create a program on your computer that can call the game service to test and control it, this is known as your Local Console.

Communication between a game, and in this case the test program, and the game service will be through a RESTful API. Additional data needed by any commands will be sent as a JSON string. Data being returned will be as a JSON string and return status should be correct based on the status of the operation.

The RESTful API will need to support the CRUD operations and follow a standard resource layout. As the course moves along, the API will grow.

The bulk of the work will be written in Python using the Django framework and be handled in various Python files – DO NOT make one large file. The suggested files are:

1. Users.py – manages the user API
2. Admin.py – manages the admin API
3. Inventory.py – manages the inventory/list API
4. Scores.py – manages the scores for each user

You will be part of a team working on this assignment and all files will need to be stored in a git repository in GitHub accessible by your team.

The lessons in class will include examples as to what you could do, you will need to take those examples and extend or customize them for use in the assignment so each lesson will build on the previous week to help create the backend service.

In the end, the entire team will present what they did and need to answer specific questions about the team's game service so each team member will need to know it – this is not a solo project!

Lesson 1: Welcome

Goals

- Cover goals of the course
- Discuss student expectations
- Review semester-long assignment

ABET Alignment

d) An ability to function effectively on teams to accomplish a common goal

h) Recognition of the need for and an ability to engage in continuing professional development

Assessment

Continuous through classroom discussion and homework due following week

Content

Agenda

Time (minutes)	Activity
10	Intro/overview/syllabus review
25	Discussion: What to Expect in the Software Industry
45	Tools: installation – Prepping the System
40	Activity: Communications
20	Interview: Whiteboarding or Collaboration Activities

Time (minutes)	Activity
30	Homework: Intro to Semester Long Assignment

Discussion: What to Expect in the Software Industry

With the class, have a whole group conversation in what their expectations, fears, issues, concerns are about going into the software development industry. Record them on a whiteboard or collaborative document (Wordle, Google Doc, etc.). Expectations:

- Unsure what it will be like
- Heard many different terms (frontend, backend, full stack) not sure where I fit
- Not sure I'm ready
- Not sure what to expect

The goal of this course is to learn more about what is going on in the software industry, work together in small teams to create a project that you can use during an interview as discussion points, learn how to interview and what to expect as a software developer.

Continuing discussion to find out where students are with their understanding of SDLC (Waterfall and Agile), roles (frontend, backend, UI, UX, architect, SQA, SEIT), Staying up-to-date (blogs, books, coding for fun), Tools of the trade (IDEs, Jira-like, git).

- How is software developed?
- Are there specialty areas or concentrations?
- What are terms you may have heard?
- How do teams communicate?
- What types of skills are teams comprised of?

- What are the biggest communication issues?

Reality: Day-in-a-Life

Discuss topics about what a software development role is like – ie, what might an average day look like.

Every day will be different, somedays will feel like nothing but meetings and other days will be *heads-down* with coding. One can expect there to be some meetings occurring on a daily basis and to work on a team of 3-5 people. The team is there to support you, from the very first day any new employee should expect to be thought of as part of the team and will be invited to all team events (meetings, lunches, planning periods, and more).

More often than not, the teams will have adopted an *Agile* methodology. There are many definitions as to what that actually means, and without getting into text-book definitions, from an operational perspective it means that there is likely a *stand-up meeting* or *daily scrum* early in the morning where everyone talks about what they did the day before, what they think they'll work on today, and bring up any areas of concern or things that they are *blocked* on.

Throughout the day, there should be lots of small conversations and meetings that take place. Somedays it may feel like there wasn't time to do any coding!

Potential Schedule

Hours at every company will differ but generally expect to arrive around 9:00 and not leave until 6:00.

9:00 Arrive

10:00 Daily Scrum

12:00 Lunch (at desk)

1:00 Team meeting for some topic

4:00 Team meeting for another topic

6:00 Leave

Bear in mind that there is usually not a set schedule for arrival and departure times. Most companies have expectations of the quality and overall contributions, not on the number of lines written or the number of tickets completed. Software is *not* a 9-5 job, many times people will log in from home to check on things, do research, or "run builds".

Environment

The work environment has changed in the past 10 years. Many companies have moved from the cubical and walled offices to an open space with desks scattered throughout or low cubical walls housing multiple members of the same team.

In newer office environments, one would likely see *phone rooms* that are usually big enough for 1 or 2 people to have a quick meeting where they can talk without bothering co-workers or take a phone call. Scheduling for other rooms is often done online. There are often kitchen areas with fridges to store lunches and often free soda, coffee, tea, and snacks.

There are often *stand-up* desks available either to become your permanent desk, used in conjunction with your permanent desk, or they are in shared spaces where people will bring a laptop over to it and work from there.

One thing that may come as a surprise is the lack of privacy in an office area. Whether it is an open area or there are small offices housing multiple team members you will be walking behind people and people will be walking behind you. There is very little *surfing the web* other than for that which is work related.

Class Structure

- Topic lecture and discussion (focused on tools, teams, and trends)
- Hands-on (install or tool use)
- Interview Q&A
- Focus on Project
- Guests or Presentations

Tools: installation – Prepping the System

- Create a LastPass account and install the plugin
- Create a google account for use in the classroom
- Create a slack account
- Gather login/usernames and setup for groups

Activity: Communications

Slack, Hangouts, language barriers

Interview: White-Boarding or Collaboration Activities

Talk about interview / white-boarding approach

1. Assumptions / Clarification
2. Brute-force attempt – talk it through
3. Test it – talk it through

(binary tree traversal example -- Given any node in a Binary Sorted Tree, how can you find the next highest number)

Homework

1. Install git:
    a. Verify it does not exist on your system by typing ‘git’
    b. If it isn’t on your system, install from: https://git-scm.com/downloads
    c. Do not install the GUI client
    d. Be sure to test that the client is installed, at a prompt enter ‘git’ and make sure the command is found
2. Create GitHub account:
    a. Create an account at www.github.com
3. Install Python 3.x (latest):
    a. Verify is it not on your system by typing ‘python’
        i. If it is not installed you will see an error that the command is not found, try also ‘python3’ in case your system has multiple versions.
        ii. If it is installed, it should give you some status information followed by a prompt. Type ‘exit()’ at the prompt
    b. If it is not installed, install it from: https://www.python.org/downloads/ and follow the directions
    c. Verify it was installed (repeat step ‘a’ above)
    d. Make a note of how to start Python (is it ‘python’ or ‘python3’?)
4. Install pip:
    a. Verify whether pip is installed by entering ‘pip’, if you have an error about the command not found try ‘pip3’ instead.
    b. If you need to install, install from: https://pip.pypa.io/en/stable/installing/ and be sure to follow all directions
    c. Re-verify whether it is installed or not
5. Install Django:
    a. If you had to install either Python or Pip, then you may already have Django, you can test with: python -c "import django; print(django.__path__)"
    Note: If you used ‘python3’ to start python, then use ‘python3’ in the above command
    b. If you need to install it, follow the instructions at: https://docs.djangoproject.com/en/1.11/topics/install/
6. Research what Agile and XP methodologies are

7. Look up the following words and know their definitions:

Software Architect	Software Developer	Team Lead	Agile
SQA	Scrum	DevOps	Software Engineer
Test Engineer	Product Owner	SDET	
XP (eXtreme Programming)	Kanban	JSON	RESTful
CRUD			

Create a document in your http://drive.google.com account with <u>your</u> definitions -- in other words, <u>DO NOT</u> copy/paste. Share it with me!

Lesson 2

Goals

- Code Management/Versioning Systems
- What is Teamwork in the Workplace
- SDLC

ABET Alignment

c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

d) An ability to function effectively on teams to accomplish a common goal

Assessment

To be done 3rd week, after homework, and observed during discussions about assignment topics.

Content

Agenda

Time (minutes)	Activity
5	Recap of previous week, topics for this week
20	Tools: CMS and IDEs
20	Trends: Agile vs Waterfall
20	Teams: Roles
15	Activity: Break into teams, create channels, groups, etc.

Time (minutes)	Activity
40	Activity: Git
30	Interview: Fibonacci Sequence
5	Homework

Tools: CMS and IDEs

Discuss:

- Code Management and Code Management Systems (CMS)
- Difference between git, svn, accurev, etc. (distributed vs hierarchical)
- Branching schemes (gitflow).
- Standardizing on tools (such as IDEs) for development

Trends: Agile vs Waterfall

Discuss:

- SDLC
    - Waterfall
    - Agile
        - Focused on customers
        - Iterative
        - Scrum
            - Time boxed by 1-4 weeks Sprints
        - Kanban
            - Time boxed by Release
- The differences, pros, and cons of each
- Team structure in Agile (scrum master, chicken and pig, celebrations)

Teams: Roles

Give an overview of teams and roles:

- Product Owner
- Product Manager
- Dev Manager
- Architect
- Team Lead
- Developers (Devs)
- SQA

Activity: Git

Form teams, organize Slack. For the purpose of the activity, I will assume the role of product, dev manager and architect. You will not need a team lead, you are all devs. In your teams:

1. Create and organize your Slack channels, groups, etc.
2. Download/install PyCharm (community edition) and write some sample python code to print out some values. Interview
3. Practice with git
    a. *follow-along with instructor*
        i. Everyone should 'clone' the repo (pass the URL via slack)
        ii. Everyone should create a file with their name, and enter their email address and contact info.
        iii. Everyone should commit and push
    b. *follow-along with instructor*
        i. Elect one person to create a file called "readme.md" and add "Team Members: their name".
        ii. Commit and push
        iii. Others pull and edit the file to put their name on it.
        iv. Commit and push – whoops, conflict
        v. Use PyCharm to resolve the conflicts

Interview: Fibonacci Sequence

Work through a problem in teams of 2. Write an algorithm to calculate fib sequence.

The sequence is: fib(0) = 0, fib(1) = 1, fib(2) = 1, fib(3) = 2, fib(4) = 3, fib(5) = 5, fib(6) = 8, …

Essentially: fib(x) = fib(x-1) + fib(x-2)

Homework

In your team, define your coding standards and document your decisions.

Modify PyCharm to meet those standards

One person in the group:

- Create the python project “proj1” and sync it with GitHub
- Same person, create a team and add team members to it

Other members of the team

- clone it
- all in PyCharm:
    - Create a readme.md
        - The file should contain 3 sections:
            - Team members
            - Bios
            - 2-truths/1-lie (2t1l)
        - All members fill out the section about themselves
            - Add your name to the team members section, should be alphabetical
            - Add your bio (should reverse alphabetical)
            - Add two truths about yourself and one lie in the 2t1l section, don’t identify which is which
            - After everyone has entered 2t1l, go guess on each other’s but do not reveal the correct answer to your 2t1l
            - There should be 1 commit for each thing you do (each team member should have 4 commits) and will need to create 4 pull requests. Any team member can review and complete the merge

Follow the Django Tutorial

Lesson 3

Goals

- The TDD Process
- The Development Process (Stories and Tasks)
- The Team Meeting

ABET Alignment

c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

d) An ability to function effectively on teams to accomplish a common goal

i) An ability to use current techniques, skills, and tools necessary for computing practice

Assessment

To be done 4th week, after homework, and observed during discussions about assignment topics.

Content

Agenda

Time (minutes)	Activity
10	Recap of previous week, topics for this week
20	Tools: Agile Management Tools
20	Trends: TDD and various test paradigms
20	Teams: Pairing, Knowledge Silos, brown-bag lunches, team meetings, etc.

Time (minutes)	Activity
20	API design, RESTful thinking, CRUD, JSON
20	Template walkthrough
40	Interview: 9-ball
10	Homework review

Tools: Agile Management Tools

- Tracking systems
  - Stories
  - Bugs
  - Tasks
- Organization
- Workflow
- Whiteboards

Trends: TDD and Various Test Paradigms

- What testing is
- When does one test
- Types of testing
  - Unit
  - Functional
  - Integration

Teams: Pairing and knowledge sharing

- Estimation
- Pairing
- Knowledge Sharing vs Silo
- Cross Functional Teams

API Design

- Discuss what makes a good API
- Define RESTful calls
    - endpoints
    - resources
    - versioning
- CRUD
- JSON payloads

Template walk-thru

A template is being provided as a GitHub project for each team. It contains the base files and project. Students should clone from their team's project and see what is in the files (show example of the GET, POST, and PUT), also show unit test file.

Interview: 9-ball

9-ball problem: You have 9 balls; all balls look exactly alike. Eight of the balls weigh the same, but the 9th is slightly heavier. Using a balancing scale, how would you find the heavier ball? What are the options, what are the pros/cons of the solutions?

Homework

Overview

When starting a new job, more often than not, your project will already be underway. Because it is an ongoing project your first set of tasks will be to get a copy of the source code ("clone" is the git term) and go through the code. In all likelihood, no one will walk you through the code step-by-step to explain how it works, it is your responsibility to study it and try to figure it out.

To help you become familiar with the bode, you will often be assigned small tasks. In this portion of the assignment, you will be cloning existing code (a link will be provided). Once cloned, you will need to look through the code and focus on four files (Users.py, Admin.py, Inventory.py, and Scores.py) along with the unit tests for each file. You are then asked to add additional code to support the missing pieces of the API. Follow the patterns you see in the code you've been reviewing to add the missing APIs and unit tests.

This is a team assignment, each team member should work on separate modules, for example one team member can add the missing Users.py API methods while another adds code to the Scores.py module. Make sure that you create a branch for your changes (that doesn't conflict with anyone else) and frequently `git commit` your code. Then, once you have a portion working remember to `push` it. Once all changes have been made and unit tests are working, submit a pull request for your team to review.

Steps

Clone existing assignment. In the existing assignment, you will find that there are multiple files as part of a Django project. The files contain portions of an API that need to be completed. This should be done by individuals in the team and not just by one person.

Before making any changes, create a branch to work on! An example branch name might be *users_api_additions*. This can be created and checked out in a single step:

```
git checkout –b users_api_additions
```

Other useful git commands:

`git add <filename>`	Add a new file to the local repo
`git commit <filename>`	Update a file in the local repo
`git push origin <branch>`	Put a copy of the branch into the 'origin' repo
`git checkout <branch>`	Checkout another branch (example 'master')
`git rebase <branch>`	Change the base of the current branch to another branch

Most of the git commands can be done through PyCharm, however, there are times when you may need to execute them by hand. Additionally, many IDEs, including PyCharm, offer you the choice of commands to run and not knowing what they do will make it difficult to choose the right command.

NOTE: The APIs do not access anything in a database, that will be in the future. All they should do at this point is produce some output, some canned data to return, log a message, and return a '200' if it works or a '500' if it doesn't.

1. Review the existing API and Unit Test code, you should see APIs to:
    a. User.py: Add a new user

   b. Inventory.py: Get the inventory for a user
   c. Scores.py: Update a score for a user
2. You need to follow the established patterns to:
   a. User.py: Get, Update, and Delete a user
   b. Inventory.py: Add a new item for a user, Delete a user's item, Update the item
   c. Scores.py: Add a Score for a user, Delete a user's score, Get a score
   d. **NOTE:** you are going to be storing the data in class level variables for testing purposes only. Sometimes these are called "mocks" because you are mocking the database functionality while you work on and test other things.
3. Some Unit Tests already exist, add any missing one, making sure:
   a. All APIs must be tested
   b. All APIs must be tested for errors as well as success

A Completed Assignment Should:

- Have a full complement of API calls that:
  - Return a Success (200)
  - Failure (500) response code if input/data is missing (GET, of course, has no input/data)
- Unit Tests that call each API and verify that:
  - A Success (200) is returned when the call is valid and data is present
  - A Failure (500) is returned when data or input is missing from the call
  - On a Success, the body/return text is what is expected
- When the unit tests are executed, no errors are detected

Lesson 4

Goals

- The Stack
- *Building* a Product
    - Continuous Delivery (CD)
    - Continuous Integration (CI)

ABET Alignment

h) Recognition of the need for and an ability to engage in continuing professional development

i) An ability to use current techniques, skills, and tools necessary for computing practice

Assessment

To be done 5th week, after homework, and observed during discussions about assignment topics.

Content

Agenda

Time (minutes)	Activity
10	Recap of previous week, topics for this week
20	Tools: Build Systems
20	Trends: Continuous Deployment and Integration systems
30	Teams: Staying up-to-date

Time (minutes)	Activity
30	API validation, return codes, short/long codes
40	Interview: Counting Characters
10	Homework review

Tools: Build Systems

- Version Numbering
- When to deploy
- Building Software
    - Bamboo
    - Jenkins
- CD/CI
    - What they are
    - How they work

Trends: Continuous Deployment and Integration systems

Discuss concepts such as Continuous Deployment and Continuous Integration systems

Cover topics:

- What does it mean to deploy?
- How does it differ based on the application type?
    - Mobile
    - Webpage
    - Downloadable
    - Cloud

Teams: Staying up-to-date info

- The Stack – Even if you aren't working in an area, know the trends
    - Full Stack

    - Front End
    - Back End
- Blogs and Books
    - Ken Beck
    - Robert Martin
    - SOLID Principles
    - 12-factor - https://12factor.net/

API validation, return codes, short/long codes

- Why Validate
- What needs to be validated
- RESTful calls use HTTP, using the return codes
- Publishing the API

Interview: Counting Characters

Create an algorithm to take a string and return a list of all the characters in the string and how many times each one occurs.

Homework

Overview

As discussed in class, most companies will have a system to manage the work in small Agile tasks (stories, bugs, etc.). You have access to Jira, as a team, break down the next few requirements and add them as "Stories" in your Jira Board. Assign them to your team members as you see fit and track the progress as you go. Adding comments to the tickets and have at a stand-up or two during the week to help each other.

API field validation is must-have at most businesses. An API, such as one that runs in the cloud, may need to be left open so it can be called by anyplace. Due to the nature of a RESTful API – meaning, one that harnesses the transport provided by *http* but not the web browser portions, an API can be accessed by many command line tools or even through a browser. This makes it an easy candidate for hacking.

One of the first defenses is to validate that the needed fields are being supplied and in valid combinations. Likewise, ensuring that the values being passed are legal values and of the correct type. If they are not, providing a simple failure code without details will provide information that the attributes are incorrect but without disclosing what is expected. While this does not prevent hacking, it can help hide enough that a novice hacker won't be able to guess what is needed to continue.

Be sure to log the fact that an invalid parameter was received (if there was an error of course). Use the "warn" level as this is not an error in the code, it is merely a warning that something is not correct.

Part of the process of working in teams, is the code review, which is essentially the approval process from a pull request. Something that went unnoticed in earlier code reviews is that it is generally frowned upon to have literals sprinkled through the code. After you submit the pull request from the last set of changes and it has been approved and merged, git pull origin master to get a new copy, create a new branch and replace the return codes with values from ReturnCodes.py file.

As always, once the API validation is complete, keeping the unit tests updated will help ensure a solid API.

Steps

Now that there is an API, it is important to begin field validation. This involves checking all parameters to make sure that they contain the right type of value (if an integer is expected, an integer is received) and if not, an error is logged and a parameter mismatch error is returned. There is a list of possible error codes in a file called ReturnCodes.py. You will notice that multiple codes may use the same value.

1. Update your repo: `git pull origin master`
2. Create a new branch and check it out: `git checkout –b <another branch name>`
3. Validate all fields:
    a. If everything is correct, return a SUCCESS from the ReturnCodes.py file, instead of a 200 (even though SUCCESS is a 200). This will mean you must change existing 200 values to reflect the new name.
    b. If any parameter is incorrect as far as you can tell, return a INVALID_REQUEST
    c. Update Unit Tests to:
        i. Make sure you are testing invalid fields
        ii. All required fields exist
4. Commit your changes using PyCharm

5. Open your pull request in GitHub and send a slack message to your team asking for a review
6. After it has been reviewed, pull a new copy of master: `git pull origin master`
7. Making sure you're on the 'master' branch, create a new branch with a new name
8. Change the return codes and update your tests to verify using the name instead of literal
9. Commit, push, open PR, send out slack message

A Completed Assignment Should:

While the APIs still do not do much other than return canned data, you should now have a good starting point for continued development. At this point you should have validation in your API, unit tests to validate the information coming into the API, and using codes that can be changed and shared with other components.

Lesson 5

Goals

- Team roles
- The ‘make’ system
- Importance of ‘vi’ and regex
- Centralized logging

ABET Alignment

b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

d) An ability to function effectively on teams to accomplish a common goal

f) An ability to communicate effectively with a range of audiences

g) An ability to analyze the local and global impact of computing on individuals, organizations, and society

i) An ability to use current techniques, skills, and tools necessary for computing practice

Assessment

To be done 6th week, after homework, and observed during discussions about assignment topics.

Content

Agenda

Time (minutes)	Activity
10	Recap of previous week, topics for this week

Time (minutes)	Activity
20	Tools: Make, Vi, Regex
30	Trends: Centralized Logging, Dynamic log levels, error reporting
20	Singletons, logs and data, when to use recursion
30	Vi – go for it!
40	Interview: Deep Copy
10	Homework

Tools: Make, Vi, Regex

Because Make and Vi are on most systems now, they are becoming more common (again).

Why Make? What is vi, what about vim? Why do we care about regex?

Trends: Centralized Logging, Dynamic log levels, error reporting

- Centralized Logging through a Log Server
- Debugging Production Deployments
    - Dynamic Log Levels
    - Error Reporting

Singletons, logs and data, when to use recursion

- Review Classes and Objects
    - What is a singleton?
- How is data logged
- What are log levels

- How/when do we add log statements
- Why recursion?

Vi – go for it!

'vi' tutorial

- Insert and Append
- Deleting characters, words, lines
- Replacing
- Repetition
- Searching
- Saving

Define Regular Expressions (regex) and the 'search' in vi

Interview: Deep Copy

Explore the difference between deep and shallow copies. How would you write a 'deep copy' method?

Homework

Overview

Another important component to a system, particularly one that runs in the cloud, is the ability to control it. Sometimes this is done with scripts on a cloud system, but this can also be done through software connecting to it. It is usually more dangerous to have local software control it as, if hacked, anyone could take control. This is something that can be addressed with encryption and authentication, however we are in the early stages and not going to concern ourselves at this point in time.

In the Console module are the basic processing to control the application. There will need to be changes made in there to access the APIs that were added before.

We also need to enhance logging, without restarting the system. The current log level (DEBUG) is set in the System.py file which is not what would be expected for a production system and should be set to INFO. We also hook in the API in the admin console that can control it, it was added during the early stages of development but never made to work.

Note that while it won't be continually mentioned in the instructions, it is important to added requirements to Jira and break them down as needed. For example, adding/updating unit tests isn't its own story, it is a subtask of just about any story and can be treated as such.

Steps

Modify the System.py, which is a singleton, and stores shared variables, it should have a log level variable that is set to "DEBUG" by default. If a user calls the PUT on the admin/logger API with a value of 1, set the log level to "DEBUG" and if they use a value of 0 set it to "INFO". A GET on the admin/logger API will return a 0 or 1 specifying INFO or DEBUG.

In the repo is a Console.py that can be executed from the command line and will allow you to control the system. For example:

```
python Console.py users add Dave

python Console.py users del Dave

python Console.py admin set loglevel 1

python Console.py admin get loglevel
```

Begin adding commands to it to *add*, *get*, *update*, *delete* (CRUD) users, scores, inventory, and set the log level. For example, create a command to "list" all users, then add 2 users and have it list them. Do both appear?

A Completed Assignment Should:

The system should now have a console to query information about the running system and have a log level that can be changed from the command line. We can now get a list of users, inventory list, scores, and the current log level.

Lesson 6

Goals

- Full Stack Developer
- Mobile App vs Native App
- Talking to the Db

ABET Alignment

a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline

c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

h) Recognition of the need for and an ability to engage in continuing professional development

i) An ability to use current techniques, skills, and tools necessary for computing practice

Assessment

To be done 7th week, after homework, and observed during discussions about assignment topics.

Content

Agenda

Time (minutes)	Activity
10	Recap of previous week, topics for this week

Time (minutes)	Activity
10	Tools: Db Visualizer
15	Trends: Native vs Mobile Apps
30	Teams: Full Stack and Front End through Back End
15	React and Node
40	UX – design and present
30	Interview: Photo Archiver
10	Homework

Tools: Db Visualizer

- What tools are there to visualize the database?
- When would we use these vs just dumping the data?
- Does it interfere with the MVC?

Trends: Native vs Mobile Apps

- What are the advantages/disadvantages to Native apps?
- What is a Mobile App or a Web App in relationship to a Native App?

Teams: Full Stack

- What is a Full Stack developer?
- How does one get experience?

React and Node

- Research: What is React or Node.js?

UX – Design

- Where are your eyes drawn?
- What is the flow

Interview: Photo Archiver

Your task is to create a design for a product that will receive photos (up to 3 at a time) from multiple users. The photos will be uploaded from a mobile device, which may drop out of service and come back.

(This is more of an activity to flesh out requirements, not to get to the design phase.)

Homework

Overview

Up until now, everything has been done in memory, meaning non-persistent storage has been used for the data. In early development, this is a common practice and easier than resetting the database each time something is being added. However, at some point it needs to be switched over to be stored in a database or some file system.

For this project, Django is used as the MVC (Model, View, Control) system. Within the Django configuration we can define what type of database is used, as well as how to access it. Django then manages the calls to the database and the SQL around it. A change made to the Model will be able to be applied through migration steps which can automatically be generated by Django and then applied to a database through a single command.

So far, only a few pieces of data were stored and they are in the 'admin' API, name of the admin is stored, along with the creation date of the service. This information is retrieved with the GET on the admin API.

You will need to start storing your data, look in the model and follow the pattern in the Admin.py file as to how to store and retrieve data.

You may have noticed that if you add a user with the Local Console, then restart your service, the user must be added again. In the long run, this is not the desired process. The next step, therefore, is to move the database from memory and to persistent media. In this case, the local disk.

We are already using SQLite3 as the in-memory database, we simply want to move that to a file. MySQL and SQLite are very similar however SQLite can store the entire database in a single file, this does limit some of the SQL functions but for our purpose, we won't notice.

Steps

1. Start by examining the existing code. The admin.py file contains some instructions that store and retrieve data from the database. Check to see how those work.
2. Check in the model to see what the fields are and notice how simple the actual code is compared to the Db design in the model.
3. Locate whichever model you are working on (users, admin, scores, inventory) and see what the fields are.
4. Follow the pattern laid out in the admin.py file and store your data using similar calls.
5. Remove any 'mock' code you added to remember things such as the username.
6. Make sure all unit tests are working.
7. Someone on your team (whoever has the ticket) will need to locate the Django configuration file for your project and change the location of the database from "memory" to a file.
8. Make sure all changes have been committed, pushed, merged (through the proper PR process) and that you have the latest code and run all unit tests.

A Completed Assignment Should:

- Store information in a live database on your local laptop.
- Be storing all data in a single file

Lesson 7

Goals

- Scaling
- Micro-services VS monolithic
- Cloud Architects
- Open Source and Licensing

ABET Alignment

a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline

b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

e) An understanding of professional, ethical, legal, security and social issues and responsibilities

j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.

Assessment

To be done 8th week, after homework, and observed during discussions about assignment topics.

Content

Agenda

Time (minutes)	Activity
10	Recap of previous week, topics for this week
20	Tools: OpenSource and Licensing
30	Trends: Micro-Services and Scalability
20	Teams: The Cloud Architect and Cloud Engineer
30	Lesson: Cloud
40	Interview: Equal Odds
10	Homework

Tools: OpenSource and Licensing

- What is an OpenSource tool?
- How do you incorporate it into your work?
- What are the license types?

Trends: Micro-Services and Scalability

- Monolithic vs Micro-services
- Scaling up/down; Scaling out/in
    - When do you Scale?
    - Why?
    - How do you know when you may need to?

Teams: The Cloud Architect and Cloud Engineer

- What is it like to work in the cloud?

- How do you design for the cloud differently than designing for downloadable applications?
- What about web apps?

Lesson: Cloud

Walk through AWS

Interview: Equal Odds

You are going to be given a file with a list of items in it. How would you go about picking a single random item from the list? You:

- May read the list only once
- Do not know how many items are in the list
- Do not have enough memory to load the entire list into memory

Homework

Overview

Most companies have set up their development environment so they can be run on a workstation or laptop. When they move into the cloud, there are often two to three different product environments, the common ones are:

1. Dev – an environment that is running the "latest and greatest" software. The database will exist but may become corrupt or reset as the latest branches are deployed which can happen from a few times per day to with every check-in. However, it is a good place to test your software, especially if there are other components involved.
2. Staging – The stage area is usually set up to be just like the production environment, but with tests data. It is less likely to become corrupt, but due to SQA efforts, it can happen and will usually be restored to a usable state. The purpose of this environment is to really test things out and let them "burn in". Sometimes this environment may be called "test" instead of "stage".
3. Pre-deployment – sometimes this is called stage, but this is an area to test the deployment before it hits production. This is almost always running the production version OR the version that is about to go into production. It is the last step before a deployment hits production.
4. Production – this is the real thing. This is the environment that live customers are using. The data in the databases will be real and at any time real customers could be using it. Logs are often being retrieved for analysis either by a human or an automated tool and, in theory, no live debugging occurs in this environment.

Now that your code is working on your local machine and you have a database to keep things persistent, it's time to move your code to the cloud. In our fake company, assume that you are going to be setting up the Dev environment and that the production engineers or devops are the ones managing the other environments.

In our company, the easiest way to deploy this will be to create an Ubuntu EC2 instance and configure it to have git, Python, Django, and Sqlite3. Then clone your repo and start up the

service, it won't be automated but it should work. Practice making changes from your laptop, pushing them to GitHub, merging them in, and upgrade your service. Everyone on the team needs to be able to do this, you never know who will need to do the next upgrade!

Steps

1. Configure AWS and poke around!
2. Create an instance and follow the online tutorials to connect to it
    a. If you have a Mac, to connect simply start up the terminal and 'ssh' to it (you will need to install the keys)
    b. If you have a PC, start up PowerShell, install the keys, and 'ssh' to your machine. If you do not have PowerShell you may need to install PuTTY
3. Install the needed products (git, Python, Django, Sqlite) and test that they work
4. Clone your repo
5. Start up the service
6. Make sure it is working and start testing the upgrade process
    a. Make a change, commit and push it.
    b. Issue a Pull Request and get it approved (do not break anything)
    c. Connect to your EC2 instance and shut down the service if it is running
    d. Pull an update from the main repo
    e. Restart your service and confirm your change is there

Now that it is running in the cloud, can your Local Console access it? Locate the IP Address of your instance, and while your service is running in the cloud, locate where 'localhost' is specified and replace it with the IP Address and see if that works.

Using AWS, create a single EC2 instance for your team. Log into the instance and make sure everyone can create keys to access the machine.

Configure the instance making sure Python is installed as well as sqlite3. Install git and clone your repo.

Now that the software and components are on the system, can you start it up? Change the local console on your laptop to point to your instance and see if your commands will work. You may need to explore the security groups to allow your laptop to access your machine.

A Completed Assignment Should:

At this point your backend server should be running in the cloud using a local database. It should be accessible to the Local Console running on your machine.

Additionally, you should be able to update and deploy a new version.

Lesson 8

Goals

- Virtualization
- Database Architects
- Big Data.

ABET Alignment

a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline

b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

d) An ability to function effectively on teams to accomplish a common goal

Assessment

To be done next week, after homework, and observed during discussions about assignment topics.

Content

Agenda

Time (minutes)	Activity
10	Recap of previous week, topics for this week
30	Tools: Virtualization
30	Trends: Virtualization

Time (minutes)	Activity
10	Teams: DB Architect
30	Activity: Installing a VM
40	Interview: Roman Numeral Conversion
10	Homework

Tools: Virtualization Tools

- Tools for virtualization include management and container based systems as well as hypervisor systems:
    - Docker
    - Vagrant
    - VMWare
    - Virtual Box
    - Parallels
- Why?

Trends: Big Data

- What is “Big Data”?
- Document vs Relational Database Systems
- What is data warehousing?

Teams: DB Architect

- Why do companies use a Db Architect?

Lesson: Installing a VM

Walkthrough using Virtual Box and Parallels

Interview: Roman Numeral Conversion

Roman Numerals: Construct an algorithm to convert a Decimal (integer) value into Roman Numerals (string).

```
String convert (int number) {

}
```

The rules are as follows:

1. I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000
2. The pattern shows the decimal numbers starting with only 1s or 5s and alternating
3. Letters next to each other are added
4. The I, X, C can be used for subtraction by placing them in front of the next two values but it cannot go beyond the 2nd value, for example IV means 4, IX means 9, IL is invalid as the I was used beyond the I's next 5 or 1. Therefore a 49 is: XLIX which is a 10 from 50 (40) plus 1 from 10 (9).
5. M cannot be used for subtraction because there is nothing higher than it
6. II means 2, not 0 as there was no concept of 0

I	1	V	5	X	10	XV	15	XX	20 …	XL	40 …	XCV	95
II	2	VI	6	XI	11	XVI	16	XXI	21 …	XLI	41 …	XCVI	96
III	3	VII	7	XII	12	XVII	17	XXII	22 …	XLII	42 …	XCVII	97
IV	4	VIII	8	XIII	13	XVIII	18	XXIII	23 …	XLIII	43 …	XCVIII	98
		IX	9	XIV	14	XIX	19	XXIV	24 …	XLIV	44 …	XCIX	99

Homework

Overview

Companies that have customer data must take great care to not lose the data. No one wants to go play a game only to find that their account was accidentally lost! One common scheme to combat this is to create a master/slave system where there is one database (master) that is used for reading and writing by the services, and another (slave) that is a shadow copy of the master database. The slave database is not used by the service but will keep itself in-sync with the master behind the scenes. With both databases on different servers, if one fails it is easy to promote the current slave to be a new master and start a new slave.

Cloud providers do more than just host instances in the cloud, they also provide services. One of the AWS services is a Relational Database System (RDS). Their RDS will keep the master/slave databases for you and should one fail it will perform the rollover automatically.

In this story, your team will need to create a single RDS instance and make it accessible by your EC2 instances. You will then reconfigure Django on your instances to use the RDS instance. Restart one of the services to configure the database on the RDS instance followed by a restart of the remaining EC2 instances. Once all instances are up and running, all instances will be effectively sharing the data, meaning that a user created by one team member on their EC2 instance will be usable by another team member on their EC2 instance.

Steps

This should be done as a team:

1. Connect to AWS and launch an RDS instance
2. Configure the security group to allow access by all of the EC2 instances for your team

3. Stop all of the services for the team and select one to be the one to configure the database
4. Start that instance up and let it configure the database – run the unit tests to be sure it is working correctly, run the Local Console against it to ensure that the database is empty and that it is no longer connected to the database on the instance
5. Reconfigure the remaining EC2 instances and restart their services
6. Have one team member create/add a user and have other team members list the users to make sure the database is shared

---

A Completed Assignment Should:

You now have a fully functioning, but not feature rich, backend service running on multiple instances using a shared database. Every member of the team should have taken part in each phase of the development process and be able to talk about it in detail. Consider projects that you may have in the back of your mind whether it is a mobile app to ask quiz questions or an action game. Keeping track of user data, scores, etc. will need a backend server similar to the foundation you've created.

One often asked question at an interview is to go into detail about something that you worked on as a class project or a solo undertaking. How much detail could you get into regarding this project?

Lesson 9

Goals

- StackOverflow and ExpertsExchange
- A Computer Scientist vs a Software Developer vs a Software Engineer
- Building a network

ABET Alignment

a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline

e) An understanding of professional, ethical, legal, security and social issues and responsibilities

h) Recognition of the need for and an ability to engage in continuing professional development

i) An ability to use current techniques, skills, and tools necessary for computing practice

j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.

Assessment

To be done next week, after homework, and observed during discussions about assignment topics.

Content

Agenda

Time (minutes)	Activity
10	Recap of previous week, topics for this week
30	Trends: Searching the Internet for Solutions
20	Teams: Engineer or Developer or Scientist
20	Activity: Research Job Openings
20	Professional Development
40	Interview: Internships, Connecting, Social Networking
10	Homework

Trends: Searching the Internet for Solutions

- Evaluating solutions
- Copyright
- Copy/paste

Teams: Engineer or Developer or Scientist

Discuss: Computer Scientists are often considered a generalist. A developer is considered someone who works primarily in a single language without branching out or moving around in the stack. An engineer is someone who tries to understand *how* it works and knows constructs, often, more-so than specializing in a single language

Activity: Researching Job Openings

How do you go about finding and evaluating job openings and knowing what you should and shouldn't apply to? What terms are you unfamiliar with?

Professional Development

What do you do in your "down time"? Conferences, videos, programming for fun?

Interview: Internships, Connecting, Social Networking

Preparing for the interview, connecting with the interviewer.

Linked-In isn't Facebook

Homework

Overview

When a product has been tested and ready for release there are a number of final steps that need to be made. So far you have worked on a project that runs in the cloud, runs using a database that can be shared by other instances. This also means that a game client should be able to add a user by contacting one instance, then access it using another instance, and update a score on yet another instance. By doing so, instead of having a single instance handling many requests by many clients, the *load* can be distributed across the servers.

This process of distributing the load is called *Load Balancing*, and in AWS language it is called an *Elastic Load Balancer* or *ELB*. You will need to create a single ELB for your team and add all of the instances to the ELB. After doing so, you should change the Security Groups on each instance so only the ELB can access the web port. Then, simply alter the Local Console to access the ELB's public address and test.

As a team, decide what other features could be added and create new Jira tickets for them. Prioritize them on how important they are and assign them appropriately. Any changes that are made must be deployed to all EC2 instances, you must keep the same version running on all instances. To test changes, remember that you can still run it locally.

Steps

1. Create an ELB – only one per team
2. Attach the team instances to the ELB
3. Adjust security groups to allow only the ELB to access the instance's web port

4. Set the ELB health check to access /v1/admin/health, this is an API which will return a SUCCESS status code without any data, the ELB will use this to determine if the EC2 instance is running the software correctly
5. Create additional tickets for additional functionality
6. Prioritize your new tickets
7. Begin work on the additional tickets in priority order

A Completed Assignment Should:

This service is now scaled out to handle thousands of requests, it is also fault tolerant such that if an EC2 instance fails the ELB will automatically remove it from the system and route future requests to other instances.

At this point the service won't scale automatically but it is very easy to remove or add additional EC2 instances and connect them to the ELB.

Finally, and more importantly, additional work is being done to bring the service to the next level.