Journal of Information Technology Education: Innovations in Practice

An Official Publication of the Informing Science Institute
InformingScience.org

JITEiip.org

Volume 19, 2020

# REDESIGNING AN INTRODUCTORY PROGRAMMING COURSE TO FACILITATE EFFECTIVE STUDENT LEARNING: A CASE STUDY

| | | |
|---|---|---|
| Cynthia Corritore* | Creighton University, Omaha, NE, USA. | cindycorritore@creighton.edu |
| Betty Love | University of Nebraska, Omaha, NE, USA | blove@unomaha.edu |

* Corresponding author

## ABSTRACT

| | |
|---|---|
| Aim/Purpose | This study reports the outcome of how a first pilot semester introductory programming course was designed to provide tangible evidence in support of the concept of Student Ownership of Learning (SOL) and how the outcomes of this programming course facilitate effective student learning. |
| Background | Many instructors want to create or redesign their courses to strengthen the relationship between teaching and learning; however, the researchers of this study believe that the concept of Student Ownership of Learning (SOL) connects to student engagement and achievement in the classroom setting. The researchers redesigned the introductory programming course to include valuable teaching methods to increase Student Ownership of Learning and constructive approaches such as making students design an authentic mobile app project as individuals, partners, or within teams. The high quality of students' projects positioned them as consultants to the university IT department. |
| Methodology | This paper employs a case study design to construct a qualitative research method as it relates to the phenomenon of the study's goals and lived experiences of students in the redesigned introductory programming course. The redesigned course was marketed to students as a new course with detailed description and elements that were different from the traditional computer science introductory programming course requirement. The redesigned introductory programming course was offered in two sections: one section with 14 registered students and the other section with 15 registered students. One faculty member instructed both sections of the course. A total of 29 students signed up for the newly redesigned introductory programming course, more |

Accepting Editor Felix O Quayson | Received: February 28, 2020 | Revised: April 19, May 29, July 13, August 3, August 14, 2020 | Accepted: August 14, 2020.
Cite as: Corritore, C., & Love, B. (2020). Redesigning an introductory programming course to facilitate effective student learning: A case study. *Journal of Information Technology Education: Innovations in Practice, 19,* 91-136. https://doi.org/10.28945/4618

| | |
|---|---|
| | than in previous semesters, but two students dropped out within the first two weeks of the redesigned course making a total of 27 students. The redesigned coursework was divided into two parts of the semester. The first part of the semester detailed description and elements of the coursework including a redesigned approach with preparation for class, a quiz, and doing homework in class, which gives students control of decisions whenever possible; and working with each other, either with a partner or in a team. The second part of the semester focuses on students designing a non-trivial working mobile app and presenting their developing mobile app at a significant public competition at the end of the semester. Students developed significantly complex mobile apps and incorporated more complex functionality in their apps. Both Management Information System (MIS) major students and Computer Science major students were in the same course despite the fact that MIS students had never taken a programming course before; however, the Computer Science students had taken at least one course of programming. |
| Contribution | This study provides a practical guide for faculty members in Information Technology programs and other faculty members in non-Computer Science programs to create or redesign an introductory course that increases student engagement and achievement in the classroom based on the concept of Student Ownership of Learning (SOL). This study also deepens the discussion in curriculum and instruction on the value to explore issues that departments or programs should consider when establishing coursework or academic programs. |
| Findings | This study found two goals evidently in support to increase Student Ownership of Learning (SOL). The first goal (Increase their ownership of learning SOL) showed that students found value in the course contents and took control of their learning; therefore, the faculty no longer had to point out how important different programming concepts were. The students recognized their own learning gap and were excited when shown a programming concept that addressed the gap. For example, student comments were met with "boy, we can really use this in our app" instead of comments about how complex they were. The coursework produced a desired outcome for students as they would get the knowledge needed to make the best app that they could. The second goal (Develop a positive attitude toward the course) showed positive results as students developed a more positive attitude towards the course. Student actions in the classroom strongly reflected a positive attitude. Attendance was almost 100% during the semester even though no points for attendance were given. Further evidence of Student Ownership of Learning and self-identity was students' extensive use of the terminology and concept of the course when talking to others, especially during the public competition. Students were also incorporating their learning into their identities. For example, teams became known by their app such as the Game team, the Recipe team, and the Parking team. One team even made team t-shirts. Another exciting reflection of the Student Ownership of Learning which occurred was the learning students did by themselves. |
| Recommendations for Practitioners | Practitioners can share best practices with faculty in different departments, programs, universities, and educational consultants to cultivate the best solution for Student Ownership of Learning based on student engagement and achievement in the classroom setting. |

| | |
|---|---|
| Recommendations for Researchers | Researchers can explore different perspectives with scholars and practitioners in various disciplinary fields of study to create or redesign courses and programs to reflect Student Ownership of Learning (SOL). |
| Impact on Society | Student Ownership of Learning is relevant for faculty and universities to incorporate in the creation or redesigning of coursework in academic programs. Readers can gain an understanding that student engagement and achievement are two important drivers of Student Ownership of Learning (SOL) in the classroom setting. |
| Future Research | Practitioners and researchers could follow-up in the future with a study to provide more understanding and updated research information from different research samples and hypotheses on Student Ownership of Learning (SOL). |
| Keywords | SOL, student engagement, student achievement, introductory programming, redesigning, computer science |

## INTRODUCTION

Faced with poor student performance and high major dropout rates in the Management Information System (MIS) academic program due to a required computer science programming course, faculty responded by designing and implementing an entirely new course that focus on increasing student engagement and performance through the application of the concept of Student Ownership of Learning (SOL). While this is likely an issue in many MIS departments, the problem is not only limited to MIS or even the U.S. In many countries, students who are required to take programming courses often respond negatively, even those in related fields like MIS or engineering (Figueiredo & García-Peñalvo, 2019).

In fact, research studies show that many students commonly have serious difficulties with learning to program (Robins et al., 2003; Hadjerrouit, 2008; Pacheco et al., 2008) and even computer science majors sometimes lack sufficient understanding of fundamental concepts to write simple programs (Eckerdal, 2009). In addition, as required programming courses become more widespread for students at all levels and in many non-technical fields, this problem is likely to grow.

In the UK alone, programming courses are required for students from age 5 and up (Brown et al., 2014). Fifteen EU European countries (Austria, Bulgaria, the Czech Republic, Denmark, Estonia, France, Hungary, Ireland, Lithuania, Malta, Spain, Poland, Portugal, Slovakia, UK 'England') are already offering coding programming in their school curriculum along with plans to develop more (EURACTIV, 2015). At the primary school level, nine EU European countries (Estonia, France, Spain, Slovakia, UK 'England') have already integrated or (Belgium 'Flanders', Finland, Poland, Portugal) are in the process of integrating coding, and at the secondary school level general education, twelve EU European countries (Austria, Bulgaria, Denmark, Estonia, France, Hungary, Lithuania, Malta, Poland, Slovakia, Spain, UK 'England') have already integrated or are in the process of integrating coding (EURACTIV, 2015).

In the United States, 45% of high schools offer programming courses, while 33 states have passed new laws and regulations to make computer science a fundamental part of K-12 education for all students (Computer Science Teachers Association, 2017). Higher education is also seeing increased requirements of programming for non-computer-science majors. In many cases, the aim of such requirements is not necessarily to produce more programmers, but rather to use programming as a vehicle for teaching computational thinking and problem-solving skills.

The fact that none of the students had taken a programming course or written code meant that they had no experience with how programming is traditionally taught. This provided the researchers with an excellent opportunity to incorporate innovative teaching into the design of the new course. To be

clear, this study reports on the design and pilot test of a completely new course and not a redesign of a non-performing course. The new course was developed by the researchers in the MIS faculty in the College of Business for MIS majors with no programming experience and was taught by one of the MIS faculty in-house.

This study's researchers decided to build the new course around a theory of learning that would increase student performance and encourage students to internalize the concepts and skills of programming, problem-solving, and computational thinking. The researchers sought something that would facilitate active learning, which has been recommended for teaching programming to MIS students (Zhang et al., 2019). In this way, the course could provide what the researchers wanted the students to actually get out of the course: rather than becoming expert programmers, students would learn cognitive skills and problem solving that would stay with them through later courses in their major, and ultimately, in their careers in the technology field.

The concept of Student Ownership of Learning provides a means for students to accomplish this thought. The father of Student Ownership of Learning, John Dewey, stated that SOL gives students something to do, not something to learn; and that the doing requires thinking, making connections, and extending knowledge, that result naturally with learning (Dewey, 1916). To the best of this study's researchers' knowledge, Student Ownership of Learning implementation in programming courses has not been explicitly studied in the context of introductory programming courses in higher education.

In this study, the researchers report on how they purposefully redesigned a new introductory programming course for non-programming Management Information System (MIS) majors entirely around the precepts of Student Ownership of Learning. The researchers discuss the underlying research that supports their design and present a snapshot of the first pilot of the new course. The researchers share their experiences and conclusions about the new course based on student feedback and the researcher's observations.

Finally, the researchers identify avenues for research around this approach. While this is a preliminary report on the researchers' efforts, it does provide a roadmap towards how to practically implement Student Ownership of Learning in a programming course for non-MIS majors and the kinds of outcomes that might be expected.

## LITERATURE REVIEW

### *PSYCHOLOGICAL OWNERSHIP*

The concept of psychological ownership is not a new one. Psychologists describe it as a common cognitive-affective state that people experience with a variety of objects. While an exhaustive examination of the body of research is beyond the scope of this study, the researchers described some of the main elements of psychological ownership that are particularly relevant to education.

Psychological ownership can be defined as "that state where an individual feels as though the target of ownership or a piece of that target is theirs" (Pierce et al., 2003, p. 5). It develops quickly and can be felt toward physical entities as well as non-physical entities such as ideas, words, artistic creations, and other people (Isaacs 1933; Pierce et al., 2003). No matter the target, objects perceived as *owned* by a person actually become a part of their extended self (Belk, 1988; Dittmar, 1992). As such, they can come to play a dominant role in an individual's identity. Psychological ownership also has an affective component, thereby, producing a positive uplifting effect (Formanek, 1994). *Owned* objects are perceived more favorably than non-owned objects (Beggan, 1992; Nuttin, 1987). Additionally, feelings of increased self-efficacy and competence related to an owned object have been observed (White, 1959).

Pierce et al. (2003) postulate that three motives drive people's use of psychological ownership. The first two are particularly salient for learning: innate desires for self-efficacy, and effectance motivation for self-identity. They postulate that psychological ownership helps people satisfy these desires. Both self-efficacy and effectance motivation (as coined by Robert White in 1959, is the state of having a causal effect on objects and events in the environment) refer to people's ability to produce a desired result. However, self-efficacy refers to one's competency with the object, while effectance motivation relates more to a feeling of having control over the object. Increased feelings of both self-efficacy and effectance motivation have been associated with having taken ownership of an object (Furby, 1976).

The second desire driving the use of psychological ownership is self-identity (Pierce et al., 2003). In fact, the term *owned* reflects incorporation of the object into one's self-identity. This feeling of ownership is thought to be developed through experiencing a living relationship with the object (James, 1890). Additionally, it develops when actively interacting or associating with an object (Rochberg-Halton, 1980) or with repeated application of control over an object (Formanek, 1994). For this reason, the researchers feel that people *own* their labor and creative efforts. In fact, people feel *ownership* of anything in which they invest their time and physical or mental energy. This is especially true with creative efforts (Durkheim, 1957). In fact, the greater the amounts of control and interaction people have with an object, the more it will be psychologically experienced as a part of their self, i.e. *owned* (Ellwood, 1927; Furby, 1976).

## *Student Ownership of Learning (SOL)*

Teachers have long sought out ways to increase student achievement. Student Ownership of Learning appears that it might be a way to accomplish this. The concept is relatively simple: according to Dewey (1916), the first to propose the model of SOL which refers to incorporating an individual's own questions and ideas that comes from their own experiences, interests, or unique understanding. These are refined and adjusted by the student over time, and eventually lead to new insights. SOL has also been associated with other desirable student behaviors such as increased concept application, taking responsibility, finding personal value, and feelings of control.

Essentially, SOL is a model of learning that blends desirable student behaviors to increase student achievement and level of learning while building on something all students have: their own experiences, knowledge, and view of how the world works. Student Ownership of Learning moves students beyond a superficial demonstration of content understanding towards demonstration of the mastery and true understanding required to apply the content. It requires the student to create goals, evaluate movement toward them, and apply learning in order to reach the goals.

Like any ownership in general, Student Ownership of Learning is postulated to be driven by innate desires for competency with an object (self-efficacy), desire for control over an object (effectance motivation), and ongoing development of self-identify (leading to eventual incorporation of the object into one's identity). The role of these drives is obvious for education. Most teachers would agree that a goal of all courses is to increase student competency with a topic and student self-efficacy.

Seeing students incorporate content and skills learned in a course into their self-identity is an aspiration goal, but one that is not commonly targeted. Student control over their learning, however, is not a typical goal of most courses. So purposefully addressing self-identify and control could be effective ways to increase SOL in a course.

Development of SOL specifically is postulated to require the presence of student (1) control, (2) intimate knowing, and (3) investment of self. *Control* here refers to a student's opportunity and responsibility to make decisions about learning tasks and how they will be implemented and completed. An example would be having students choose their own topics for a project. Milner-Bolotin (2001) found that a student's interest in their project topic positively affected feelings of ownership. This makes sense, as SOL predicts that objects that are more known and over which one has more control

are more likely to be perceived as part of self (Pierce et al., 2003), thus increasing Student Ownership of Learning.

The second requirement, *intimate knowing*, refers to the student getting to know the object of ownership from multiple perspectives and in great detail, often better than other students or the teacher. Finally, *investment of self* refers to a student's commitment to devote their time, effort, and energy to follow a course of action with the object with the expectation of a worthwhile result. It is the responsibility of the teacher promoting SOL to provide a learning environment that supports development of these three elements in students.

How can such an environment be created? As mentioned above, and supported by research, students must be given control over objects of ownership, provided ways to develop expertise and confidence in their abilities with the objects (self-efficacy), and have opportunities for repeated, extended, deep interaction with the objects in order to incorporate them into their identity. Again, these objects can be projects, ideas, or important concepts. Not surprisingly, typically SOL has been observed when students are active learners, agents in their own learning, generate new knowledge, and/or relate their own experiences to new knowledge (Rainer & Matthews, 2002). Providing these types of experiences are known to facilitate SOL.

Unfortunately, the researchers of this study were unable to find research studying the application of SOL to a programming course for students with no programming experience. However, research by Dounas-Frazer et al. (2019) is promising. They developed a preliminary model of student ownership of physics projects that they plan to fully develop and report upon in the near future.

As it may be expected, the role of the teacher must also change in order to facilitate increased student ownership of their learning. Acting as a coach to help students set their own learning goals, apply course knowledge, evaluate their own learning progress, and take control of their own learning is critical. A teacher in this role becomes a valued resource for students to use in order to achieve their goals as well as a source to validate the students' decisions, choices, content application, observations, and evaluations (Chan et al., 2014).

## *Constructivist Model of Learning*

Student Ownership of Learning is consistent with the precepts of the Constructivist Model of Learning (Dudley-Marling & Searle, 1995; Hadjerrouit, 2005; Milner-Bolotin, 2001). Constructivism proposes that knowledge or meaning for an object is not innate, but rather is constructed through one's experience with that object in context (Honebein et al., 1993). Understanding is furthered by experience, for example, it is embedded in the interaction of the student with the object. Therefore, constructivism advocates that the teacher should provide experiences involving the object in order to increase SOL. Hadjerrouit (2005), in his pedagogical constructivist model, also recommends the role of teachers be as guides and facilitators of learning rather than simply transmitters of knowledge.

Honebein's (1996) list of the pedagogical goals of constructive learning environments illustrates the overlap of constructivism and SOL. Student ownership is one of the fundamental goals of constructivism-based learning. Constructive learning requires placing responsibility for learning in the students' hands, which ultimately cultivates group and individual autonomy and initiative. All of these would also facilitate Student Ownership of Learning.

A constructive approach requires both authentic activity and a real-world context for the object of ownership (Hadjerrouit, 2005; Honebein, 1996; Honebein et al., 1993). Authentic activities are those that are real or valid in a particular context. They are typically identified as activities that will enhance students' ability to interact successfully over an extended time with their object of ownership. Context refers to the complexity or realism in which the learning takes place. Simplifying a project so that it clearly illustrates one or two important concepts but not realistically complex in context would not be a constructive approach.

Providing realistic levels of complexity is necessary. In fact, doing so can actually make the learning easier as tasks in context cause the application of knowledge to be more intuitive than when applied in a decontextualized or artificial setting (Harel & Papert, 1990). An authentic project provides students with the opportunity to generate and evaluate multiple problem perspectives, recognize when particular skills and knowledge are needed, and retrieve and apply knowledge and skills needed to solve a complex real-world problem. In this way, it is clear that a constructive approach promotes ownership of the object (e.g. project), leading to increased SOL.

## *THE FLIPPED COURSE*

Flipped courses have become a popular way to implement active learning (Freeman et al., 2014). In fact, they have been shown to cultivate a stronger sense of SOL specifically (Mok, 2014). They have also been shown to increase the level of learning, per Blooms Taxonomy, as they can focus on the higher levels: applying, analyzing, evaluating, and creating (Marshall & DeCapua, 2013). In addition, they appear to improve attitude toward a programming course for MIS students (Loftsson & Matthíasdóttir, 2019).

The Flipped Learning Network offers the following definition of flipped (FLIP Learning, 2014):

> Flipped learning is a pedagogical approach in which direct instruction moves from the group learning space to the individual learning space and the resulting group space is transformed into a dynamic, interactive learning environment where the educator guides students as they apply concepts and engage creatively in the subject matter.

Whereas a lecture-based approach has students listening in class and solving problems at home, students in a flipped course have their first exposure to new course concepts outside of the classroom, watching videos or reading. They then use this knowledge during class for engaging in active problem solving and homework that require application of the concepts under study (Bergmann & Sams, 2012; Lage et al., 2000). Hence, the hardest part of learning is done in the classroom, where they have access to their teacher and peers for help.

Proponents of the flipped approach cite many advantages, including:

- Students' control the time and pace of initial learning by consuming basic content knowledge on their own schedule outside of class. This can be done as frequently as desired and in multiple ways, such as reading course material, listening to related podcasts, or watching appropriate videos or screen casts.
- Teachers can focus face-to-face classroom time on relevant problems and tasks to increase content knowledge and application of the knowledge (Gannod et al., 2008).
- Teachers can use class time to check for and ensure student understanding.

Key here is the passing of control to the student, which has been shown to increase SOL. Students recognize this. Thalluri and Penman (2016) found that while SOL increased in flipped learning, students recognized and commented that the flipped approach increased their opportunity for developing their own ownership of the learning.

Research on the effectiveness of flipped instruction has also identified positive effects. For example, students found it easier to take notes and understand content when first exposed to it outside of the classroom via video lectures (Foertsch et al., 2002). Flipped classroom students may also have higher performance and morale (Papadopoulos & Roman, 2010) as well as deeper understanding (Warter-Perez & Dong, 2012). Sharp and Sharp (2017) found in a study of teaching C# using four different modes that Flipped, Blended and Online had greater academic impact than the traditional approach.

They attributed this to the greater degree to which these approaches all fostered greater active learning, student engagement, and self-regulation, that in turn, appeared to increase student academic performance.

However, other researchers have not found a clear advantage of flipped over traditional teaching approaches, although it has been frequently shown not to have a negative impact on learning (Hodge et al., 2014; Johnson & Renner, 2012; Love et al., 2014; Yang, 2017).

# RESEARCH METHODOLOGY: THE NEW COURSE

## *BACKGROUND OF THE PROBLEM*

Creighton University is a medium-sized private Jesuit, Catholic institution in Omaha, Nebraska, the Midwest of the United States. The student populations for the study were MIS majors within the College of Business. Prior to the development of the new course, MIS major students were required to take a general programming class offered by the Computer Science Department. They considered this course to be extremely difficult and not relevant for their major. Computer science and MIS students were in the course together and while the MIS students typically had never taken a programming course, most of the Computer Science students had taken at least one. Their perception of the course as very difficult is not surprising; programming requires an intense, structured, precise method of thinking and problem-solving, design and creation of actual products, and learning at all of the levels of Bloom's cognitive learning taxonomy (Bloom et al., 1956; Xu & Rajlich, 2004). What was surprising was that students did not see the importance of the topic for the MIS major. Without any experience with programming, they couldn't see that many of the things learned in a programming class, such as problem solving and computational thinking, went beyond just coding. These are both central to the MIS major.

Students' negative attitudes toward the course, along with high failure rate (over 60%), led students to avoid the course, and therefore, the MIS major altogether. The number of MIS majors significantly declined over a two-year period. In a field that nationally has low numbers, this was not sustainable. While it was unlikely that this course was not the only reason for the decline in the MIS majors, it was one of the early factors the researchers chose to address. The researchers decided to carefully design and offer a completely new introductory programming course built around the tenets of Student Ownership of Learning in order to increase engagement, improve attitudes, and increase student achievement in the course.

## *METHODOLOGY*

This paper employs a case study design to construct a qualitative research method as it relates to the phenomenon of the study's goals and lived experiences of students in the redesigned introductory programming course. The redesigned course was marketed to students as a new course with detailed description and elements that were different from the traditional computer science introductory programming course requirement. The redesigned introductory programming course was offered at Creighton University in two sections; one section with 14 registered students and the other section with 15 registered students. One faculty member instructed both sections of the course. A total of 29 students signed up for the newly redesigned introductory programming course, more than in previous semesters, but two students dropped out within the first two weeks of the redesigned course making a total of 27 students. The redesigned coursework was divided into two parts of the semester. The first part of the semester detailed description and elements of the coursework including a redesigned approach with preparation for class, a quiz, and doing homework in class, which gives students control of decisions whenever possible; and working with each other, either with a partner or in a team. The second part of the semester focuses on students designing a non-trivial working mobile app and presenting their developing mobile app at a significant public competition at the end of the semester. Students developed significantly complex mobile apps and incorporated more complex

functionality in their apps. Both Management Information System (MIS) major students and Computer Science major students were in the same course despite the fact that MIS students had never taken a programming course before; however, the Computer Science students had taken at least one course of programming.

## *GOALS FOR THE NEW COURSE*

The researchers had two goals for the new course design. Students would:

1. Increase their ownership of learning (SOL).
2. Develop a positive attitude toward the course.

The initial design had to incorporate conditions that increase SOL as well as reduce fear and apprehension towards programming. Increasing SOL would likely help improve student learning and achievement. In this way, the researchers would be helping students to learn not only to program, but perhaps more importantly, to think computationally and problem-solve logically. These skills are essential in MIS, and therefore, valuable in later courses. Developing a more positive attitude toward the course and its content could increase student self-efficacy as well as confidence in the course. It might also reduce the drain of students from the MIS major due to failure rates and the feeling of not wanting to take a course that is perceived as irrelevant.

The researchers decided early on to employ a flipped pedagogy and to integrate a significant, working prototype of a mobile app as the main project and focus of the course in order to address these two goals. Flipping allowed the researchers to move the most difficult part of the course; actually writing code and problem-solving in the classroom where teachers and peers would be available to help. During class, the teacher could help students solve problems, see alternate solutions, and think about approaches to apply the course concepts to their programs. The role of the instructor would be as a facilitator, coach and resource, not simply a giver of knowledge.

### Increasing SOL

The new course had to be designed in a way that facilitated the development of SOL. This meant that student control had to be increased; activities that would provide repeated, extended, and deep interaction with the concepts and products of the course had to be developed. Additionally, this had to be done in the context of employing a flipped approach that fit within the timeline of a traditional course.

The researchers' targeted five conditions that have been associated with increasing SOL (see Table 1). The key driver of all of these was the incorporation of a significant, appropriately complex programming project that would give students extended opportunities for creativity, control, and deep, important interaction with the course materials, content, and ways of thinking. It also would give them a context in which to apply the course materials that illustrated their usefulness and value in a concrete way.

The creation of a mobile app as a project also provided a way to give students significant flexibility in deciding on what they wanted to build, the features they wanted to include, and the design they wanted to use. This gave them a way to easily infuse creativity into their projects, also increasing SOL. Additionally, the capacity for adding features and functionality to mobile apps is large, providing a way to keep the projects complex.

**Table 1: Planned course practices to increase SOL**

| Conditions for SOL | Related Course Elements |
|---|---|
| Control | • Students identify their own project idea.<br>• Teams of two to three students decide what to make, how to make it, and how feature-rich to make it based on their desired grade and their team's desire to win the competition.<br>• Students decide on what instructor demonstrations they want in class related to advance programming elements they want for their apps.<br>• Students choose the amount of complexity to include in their project.<br>• Students develop goals and plans for each milestone. |
| Intimate knowing (extended experience and interaction with the object of ownership) | • Flipped class structure<br>• Long-running, complex project limited only by students' level of skill and ambition<br>• Students find, build, manipulate, and explore functionality they want to incorporate into their apps, adopting some, abandoning others<br>• Positive emotional component - immediate feedback of emotional joy when programming errors are corrected, and code runs correctly. |
| Investment of self (genuine creative effort) | • Brainstorm project ideas.<br>• Develop own design.<br>• Identify desired features.<br>• Develop the app itself.<br>• Plan, create, and deliver a presentation and demonstration to the public. |
| Complex and authentic work | • Mobile apps used as project as students have experience with them in the real world.<br>• Project to complete a fully functioning significant mobile app to be presented to campus and business community.<br>• Mobile app required to have features that went beyond basics taught in the course.<br>• Held to three project milestones for the semester.<br>• Participated in an all-sections app competition with significant prizes open to the public. |
| Self-identify (incorporation of object of ownership into student identity) | • Project that requires direct and clear application of course content.<br>• Work in small teams (2-3) so everyone has to participate.<br>• Allow creativity and personality to guide project idea and design.<br>• Public competition against other teams.<br>• Provide opportunities and give credit for student-student help.<br>• Immersion in project work for the entire second half of the semester. |

### Developing a more positive attitude toward the course

The researchers believed that having a more positive attitude toward the content and the course would increase students' self-efficacy as well as confidence within the course. To provide a familiar and specific context for the programming that would be done in the course, the researchers thought to make the course less formidable and intimidating to the non-programming MIS students. Consequently, the researchers designed the course to focus on mobile app development. All of the students were very familiar with mobile apps, and so this took the unknown aspect of programming off the table. They had all heard about mobile apps being developed by young teenagers who have had successful sales of their app. If young kids could do this, so could they. Picking a specific focus for the course also removed some of the unknown aspects by giving them an exact idea of what they would be doing in the course. So, the researchers renamed the course to, "Introduction to Mobile App Development", thus focusing on the significant of a complex project around the development of a real-world problem to solve.

## *Design of the New Course*

### Time allocation

The researchers divided the semester course into two parts. The first half would focus on the students learning how to program. The second half of the course would focus on mobile app projects. The new course was designed to be a flipped course, with students watching required video demonstrations and explanations along with doing readings before class and then doing programming in class. See the course schedule in Appendix E for details.

During the first half of the course, the students would learn the programming language. Per the flipped method, students would prepare for class by watching video lectures, video demonstrations, and reading their textbook. Then all of the time in the classroom would be exclusively spent on coding and problem solving. The researchers incorporated the use of paired programming for all of the class coding time. The paired-programming paradigm has been associated with better programming outcomes in beginning programmers and has also been shown to improve outcomes with females (Ying et al., 2019). The researchers felt that this approach would encourage students to problem-solve with their peers as well as the teacher, thereby, recognizing that there isn't enough time for the teacher to answer every question.

In the second half of the semester, all class time would be spent on student teams working on their mobile app. While students would not have class preparation or quizzes to do, they would be working twice a week in class and outside of class as needed. This would allow students to really develop a focus on their project and fully immerse themselves in it during class when help is available for the hard problems to solve.

### Course design in detail

In order to provide a realistically complex coding environment, the researchers decided to use a commercial IDE (integrated development environment) that would work on both PCs and Macs. The researchers chose NSB AppStudio, as the University had a license (there is also a free version), it had a graphical side to use for the interface design, and a code behind where students wrote code as event handlers. That way they could focus on coding, not creating interface widgets. The researchers didn't have to talk about HTML and CSS. It is available at https://www.nsbasic.com/. The researchers decided to use JavaScript as it is relatively easy to learn, contains most of the programming concepts of a more serious language, and worked with the IDE (Integrated Development Environment) and most APIs.

The grading distribution for the course would heavily weight the project, as it was the opportunity students had to show the degree to which they had mastered the content and skills of the course (see

Appendix D). Consequently, there was no final exam. The programming concepts that would be covered included variables, arrays, strings, operators, expressions, Booleans, conditionals, loops, functions and scope, interacting with remote databases, and simple API requests. Some of the widgets from the interface side of the IDE (Integrated Development Environment) would also be included. The schedule is available in Appendix E. A midterm would conclude the first half of the semester.

To prepare for class, students would be required to watch 2-4 short (10-20 minute) videos covering new programming concepts and coding demonstrations as well as to do textbook readings. Students would be able to view the videos multiple times with the ability to pause and rerun. These flipped teaching strategies have been shown to increase Student Ownership of Learning as students have control over how fast and often to consume the content (Mok, 2014; Musib, 2014).

If students did not prepare for class, they would not be able to complete the work in the class and might be dismissed. To help ensure that they were prepared for class, every class started with a short (5 minute) knowledge-level quiz over the preparation materials. It would be administered in the course management system (Canvas) and be a simple multiple choice. Students would then spend the rest of the class completing an assignment with their assigned paired partner (see Appendix C for examples of assignments). The decision was to use Code Academy® for the early assignments, as each unit had many interactive activities and programs that allowed students to apply new programming knowledge.

Each assignment included Code Academy lessons as well as optional Extra Credit programs that provided student another way to control their learning. Each paired student would be required to hand in their own programs in order to extend the exposure of each student to the programming concepts under study as well as provide practice with typing code and debugging. The assignments would be designed to give students a structured way in which to apply the concepts as well as practice using previously learned and new constructs.

The project was where SOL would really come into play. The project would provide extensive opportunities for student decisions and control. For example, it would require students to come up with their own idea and design for their app as well as decide on which coding constructs to use to implement their design within the requirements of the project, and what advanced features to incorporate based on the grade they wanted to target. They would also have control over their timelines within three required Milestones, deciding who is to do what, when, and how within their teams.

The requirements of the project would be straight forward; a working mobile app prototype of adequate complexity with 2-3 advanced features chosen by the team (see Appendix A for the project description given to students). The app was called a working prototype because it did not have to be completely, fully functional. The mobile app was not fully functional, but during a demonstration, the scenario path through the app taken by the student presenters made it seem like it was fully functional. This is because students would be following a path that used all the fully functioning features. For example, if there was a dropdown with 10 choices, perhaps only the first two would be functional. In this way, the researchers could focus the students on making the main features work, but not get into the detail of making every feature/element fully functional.

To do that would have added a lot of repetitive, data-entry type of work for which there was not enough time and that did not significantly increase their learning. However, all of the advanced features had to work completely. The advanced features incorporated into the app would be entirely up to the team, but the teacher would provide feedback as to whether they were complex enough or were too complex. Advanced features could be anything technical that had not been explicitly taught in class in detail, such as commercial APIs (e.g. Google Maps, Yelp, Weather, etc.).

In order to incorporate enough complexity into the project, the researchers decided to use teams of three students. The teams would be created in such a way as to balance member coding, creative, and organizational skills on each team. Working in heterogeneous teams would also lend a realistic flavor

to the project. Grading would be equal parts on how well their app worked and its level of difficulty (based on the coding and advanced features they incorporated).

A perfect execution of a simple app would receive a lower grade of B- to C versus a less than perfect execution of an app that contained complex, challenging, and/or very creative elements (B+ to A). The project was divided into three Milestones (see Table 2; see Appendix A for the specifications given to the students). This would reflect how a large project would be organized in the real world and would also serve to keep students on track.

**Table 2: Tasks to be completed in each Milestone**

| Milestone | Tasks |
|---|---|
| 1 | 1. Develop an idea for the app.<br>2. Identify the target audience for the app.<br>3. Determine what the app will do, i.e. functional elements.<br>4. Determine the value-added features (i.e. the bling). |
| 2 | 1. Make revisions made based on instructor feedback from Milestone 1.<br>2. Create storyboards (sketches) of main screens in the app.<br>3. Design algorithms for major functional elements.<br>4. Enumerate top three key user tasks (primary three things users will do with the app).<br>5. Identify five potential problems or difficulties you anticipate in developing the app and how you will address them.<br>6. Create a timeframe of the main activities to be done and who will do each. |
| 3 | 1. Make revisions based on feedback from Milestone 2.<br>2. Complete a minimum of 50% of the app functionality (this includes input and output from a database table).<br>3. Demonstrate with the app how a user would complete the three key user tasks.<br>4. Identify what is left to do and the plan for completion.<br>5. Create an outline of the demonstration presentation. |

The project would culminate with a public presentation and competition. The goal of including a competition was to engender excitement within students as well as to make it clear that their work had value outside of the course. This has been shown to be a strong motivator for learning and a facilitator of SOL (Kearsley & Shneiderman 1998; Ziegel, 2004). It was also another way to give the students more control as it might encourage students to take on more difficult challenges, fueled by their desire to be in the top five in the competition. Student teams would present and demonstrate their app to a panel of judges and an audience.

All course students would be required to attend. Both groups would rate each app in real time using set criteria. Judges could be faculty (with technical backgrounds) and other technical persons in or outside of the University. The audience would be open to anyone in the University as well as the community. The top 10 teams would receive significant awards such as iPads or Apple Watches. These would be donated by the University and local businesses. Other prizes could be purchased using money raised by the students on social media.

The faculty recognizes that competition might have a negative effect on some students. So, while the presentations would be scored, how each team did in the competition would not count toward their course grade. In addition, the teacher could not be a judge.

# FINDING / RESULT

## *THE COURSE PILOT*

### Running the pilot course

There were two sections of the new course offered; one with 14 students and one with 15 students, but two students dropped out within the first two weeks of the redesigned course making a total of 27 students. The faculty worked hard to get the word out that this was a new course, replacing the old programming course requirement, and that students would develop a working mobile app. There was excitement that many students signed up, as previous semesters the researchers had seen few students registered for the Computer Science introductory course.

One faculty member taught both sections. The faculty began the course by giving students a clear and detailed description of the elements of the course that were different from those of traditional course: the flipped approach with preparation for class, a quiz, and doing homework in class; giving students control of decisions whenever possible, and working with another student(s), either a partner or a team, at all times, a focus in the last half of the semester on designing and developing a non-trivial working mobile app, and a significant public competition at the end. This made some of the students apprehensive, but most seemed excited about the possibilities. This was reflected in comments such as:

"It was obvious this course was going to be different from any course I have taken before."

The first two weeks students were a bit hesitant to work with their partner and had to be encouraged to interact with them to plan solutions, approaches, and when they had problems. Also, students who tried to work alone alongside their partner never completed all of the homework assignment during class. The faculty pointed out that the homework was designed to be completed in class with a partner, not just one student. Also, during this time, the faculty adjusted to the new role in the redesigned course. It was strange for the faculty to just wander around the class answering questions as they came up while all the students were working and consulting with each other. When no one had a question, the faculty was at a loss about what to do. The faculty learned to just pick a pair of students to interact with by assessing what they were thinking and doing. The faculty's favorite would be to wander over and say, "tell me about what you are doing right now", or "how did you come to that conclusion", or a favorite to ask "how is that working out for you?" This gave the faculty a great teaching moment if the students were going down the wrong path, or a chance to congratulate them if they were doing the right thing. The faculty often shared the good ideas with the rest of the class.

By the third week students were acting differently in the classroom. They gave the impression of being in charge of the course; they owned it. They approached class in a very deliberate manner. They came into the classroom, took their online quiz, and then went immediately to work with their partner or team with no faculty prompts. They told the faculty that they needed to maximize their homework time. They paused only when called by the faculty to go over any problematic quiz answers. Their engagement was obvious. The classroom was alive, with students interacting with their partners as well as students from other teams through consultation. Students moved around, moved the tables to fit how they wanted to work, and had to be told to leave when class period was over.

The faculty circulated the classroom, answering team and individual questions, helping to debug problems, posing questions about students' work that had been done or ideas they had, and stepping in with guidance when needed. By the end of the semester, student teams were actively consulting with each other about problems they encountered, things they were trying to do that were not working, asking the faculty for advice on directions to take, eavesdropping on faculty interactions with other teams, solving hard problems within their teams, and extending their learning into new areas in order to make their final project superior to that of other teams. It was obvious that teams had taken

ownership of their projects; they always sat with their team, rarely talked to anyone outside of their team (unless they were consulting) and gave their teams names (something the faculty never required).

The culminating competition, called the Argy-Bargy (named by affiliated UK partners), was impressive. Every team had a functional app prototype that they demonstrated on their mobile devices by projecting their phone screen to a large projection screen in an auditorium. The level of complexity of the apps was astounding for students who had never programmed before, and in some cases positioned student as consultants to the University IT staff. See Appendix B for project examples. Finally, it was clear that students in the course had taken ownership of their learning. This was reflected in the comments of a student who, prior to taking the course, had never programmed before and had started the course with the statement "I hate programming." She told the faculty after the course finished, "My team is going to take our app public. I can't wait to start working on it again – it's been three months since I touched it and I really miss it."

# DISCUSSION

## *MEETING THE GOALS*

The goals that were set for designing and implementing a course built on Student Ownership of Learning were met. The first goal, to increase SOL, appeared to be accomplished. It was also clear how students were finding value in the course content and how they had taken control of their learning. This is true of (Pierce et al., 2003) study of psychological ownership. The faculty no longer had to point out how important different programming concepts were.

The students recognized when they had a loophole in their learning and were excited when shown a programming concept that addressed the loophole. For example, loopholes were met with "boy, we can really use this in our app" instead of comments about how complex they were. The course produced a desired result for students; for example, the way students would get the knowledge they needed to make the best app they could. This is true of studies done by (Belk, 1988; Dittmar, 1992).

The faculty's favorite example of the control students took of their time in class was the day that the faculty had to be late to class. The faculty had told the students that they should expect 15 minutes lateness. Upon arrival, the faculty member found them all engaged in coding, troubleshooting, and having team project discussions. Just as they would have been had the faculty member not been late.

They were so engaged that they didn't even notice the faculty arrival in the classroom! Another example was their impressions and observations about the IDE (Integrated Development Environment) they were using to develop their app. While it was a mystery at the beginning of the term, by the end they were consistently voicing strong opinions about what they saw as deficiencies in the tool that they thought should be corrected by the developers. Interestingly, they were on target with their critiques.

Students were also incorporating their learning into their identities. For example, teams became known by their apps: The Game team, the Recipe team, the Parking team, etc. One team even made team t-shirts. Further evidence of SOL and self-identify was their extensive use of the terminology and concepts of the course when talking to others. This was particularly impressive outside the classroom, such as during the public competition, when works like IDE and API were common.

This reflects that they were reaching the level of unconscious competence, the top level in the development of competence (Broadwell, 1969). At this point, use of the terms had become second nature in the context of their project and were not easily explained or defined to the lay audience as they were being used unconsciously.

Another exciting reflection of the high SOL that occurred was the learning students were doing by themselves. Instead of just consulting the faculty or textbook, they often moved onto the web and

learned where programming information, tutorials, forums and help was available, what sites were good, which were bad, and how to use the information. The genuine need they had for information drove them to learn and refine these skills.

All of these behaviors reflected their increased ownership in their own learning. The outcomes of this were, as the researchers had hoped, good student performance in the course. The researchers saw a reduction in the course failure rate (one out of 27); lowest grade was an F, and high quality of student projects. Students developed significantly complex mobile apps, and in doing so extended what they had learned in the classroom to learn and incorporate more complex functionality in their apps.

Debugging reached new levels as they worked with more complex code. Students were anxious for specific topics to be covered (e.g. how to connect to the database) as they had plans for using this functionality in their app. Often, they wanted a specific topic like this moved up in the schedule as they were ready to incorporate it into their app.

The second goal, developing a more positive attitude towards the course, also showed positive results. Student actions in the classroom strongly reflected a positive attitude. Attendance was almost 100% all semester even though no points for attendance were given. Students completed an Ultimate Question (UQ) survey at the end of the course.

UQ is often used for measuring customer satisfaction in business; it has been found to provide useful feedback for teachers as a course evaluation (Gallo et al., 2015). It is composed of two questions: (1) Would you recommend the course to a friend? and (2) why or why not? All of the students completed the evaluation (n = 27). One hundred percent said they would recommend the course to a friend. The top reasons given were:

- The amount they learned in the course.
- Working in groups and collaborating.
- Availability of individual help from the professor in class.
- The relaxed atmosphere.
- The style of the course.
- Independence.

Other representative answers included:

- I would recommend this course because it gives you the experience that you would need when working through challenges and real-life scenarios.
- This course helped me learn a lot about developing an app which made me feel proud of myself. I love the way that [the instructor] teaches us. It's a really interesting class with hands-on experience. It's amazing at the very end of the semester seeing what I have done in this class and the app I developed.
- It not only challenges you to work hard on your project and assignments, but it also forces you to be creative and think outside the box, which I think is vital when it comes to learning and preparing us for life after college.
- It's a great combination of creativity and applicability. You learn a ton and the pressure commits it to memory. It is unbelievable how much you learn in this class from developing a mobile application.
- To be honest, in the beginning of the class I thought the class would be really hard and dry especially for someone who doesn't have any background with coding. And I was terrified about the project that we have to code an app on our own. But throughout the course, I learned so much about coding. It was hard in the beginning but after a while once I understood the logic, it became very interesting. Overall, I really enjoyed the class which I didn't think I would.

- The independence that teams get in order to rely on one another.

## CONCLUSION

The goal was to design a new programming course for non-programmers from the ground up based on the model of Student Ownership of Learning (SOL). The researchers chose this model as one that would increase student achievement. The researchers also targeted creating a more positive attitude in students to the programming course. The researchers assessed the pilot run of the course appeared to do these things. The methods the researchers implemented in order to increase SOL appeared to have the desired effect of increasing SOL.

Students told the researchers that they felt they learned more in the course than in most courses, something that Thibodeaux et al., had found in their study of SOL (2019b). Was this due to the heavy use of SOL-oriented strategies? In this pilot, the researchers believe that this could be the case. On one team, one of their members was a self-acclaimed programming hater initially; regardless, stayed together with the team, started an LLC, and tried to go public with their app.

The team spoke to several organizations that were interested in buying it as a prototype. This appears to indicate that these students had incorporated their object of ownership in the course; programming learning and their app into their own self-identity.

The faculty continued for two more semesters to help them outside of class to refine and extend their app. They also showed strong mastery of the material, and their desire to take it to the next level likely reflected a high self-efficacy with the content. They had obviously taken control and were driving the process of taking their app to market long after they were required to interact with the app. All of these elements were predicted by SOL and ownership in general and seemed to be evident in many students (Pierce et al., 2003).

Although there was not a direct way to compare students achievement in the pilot with other students in the old course; the level of application of programming concepts and skills in the project apps built by students in the new course was remarkable. It is likely that SOL was increased in these students in the new course, something which has been shown to move students beyond a superficial demonstration of content understanding towards demonstration of the mastery and true understanding required to apply the content.

As the old course did not have any assignments at this high of a level, the researchers suspect that the learning done by most of the students in this course was likely much deeper and of higher level than in the old course.

**Table 3: Final Course Grade Distribution in Pilot New Course**

| Final Course Grade | Number of Students | Comment |
|---|---|---|
| A | 18 | |
| B | 8 | |
| C | 0 | |
| D | 0 | |
| F | 1 | B avg on assignments<br>A on midterm<br>F on Milestone 3<br>F on overall project grade (non-participation) |

The effects of increased SOL were also seen in the pilot. The researchers were very excited about the student performance in the course (see Table 3). Student achievement was high, with just one student failing the course. This was a strong improvement over the previous old course.

The strategies the faculty used to increase SOL appeared to have their intended effect. For example, students commented that the authentic project and the work time in class on the project were their favorite parts of the course as it gave them a reason for learning what they had in the first part of the course, they could work on it with their teams and the faculty all present without trying to get everyone together outside of class (which was a challenge), and that it was fun seeing their own ideas come to fruition.

These comments were consistent with findings of earlier researchers. Thalluri and Penman (2016) saw students identify and comment that the flipped approach increased their opportunity for developing their own ownership of learning. Also, the use of an authentic, complex project allowed the researchers to add many opportunities for student control which is necessary for development of ownership (Honebein et al., 1993; Hadjerrouit, 2005).

The researchers of this study also saw an improvement in attitudes towards the programming course, perhaps due in part to the use of the flipped approach as well as the use of an authentic project (Loftsson & Matthíasdóttir, 2019). While the researchers did not measure attitude change explicitly, the level of energy and excitement in the classroom every day, the eager engagement of the students during class work times, and the extent to which students applied themselves outside of class on the project were more than enough at this course level.

The student course evaluations also were primarily positive, with a few negative comments about the IDE and one of the texts. Opinions were evenly divided on Code Academy. The pilot marked the end of the programming requirement being a roadblock for the MIS major. The department plans to continuously run the newly redesign course in the MIS major curriculum. The researchers have seen the MIS major grow and improved, and the department has even added another elective programming course to the curriculum.

An unexpected component of SOL that the researchers witnessed was the use of peer pressure by the students. Nothing the researchers reviewed in the literature predicted this. In the first part of the course, when a student came unprepared to class, their paired partner let them know in no uncertain terms that this was unacceptable. However, compliance with the class preparation element was almost a non-issue. In the project phase of the course, teams could fire an under-performing team member. While this did not happen, it gave students who otherwise might have ridden the coattails of their teammate's incentive to come to class prepared and be productive team members. Team members also used a free tool named Teammates (https://teammatesv4.appspot.com/) to do a peer evaluation of themselves and their teammates at the end of each Milestone and when the project was complete.

The faculty used these peer evaluations, along with the researchers' observations, to adjust individual team member grades up or down accordingly on each Milestone and for the overall Project with the goal of minimizing freeloading. As a result of the peer pressure and evaluations, the faculty rarely had to talk with students about poor attendance, late assignments, poor preparation, or not participating in the project. Their paired partners and team members kept everyone in line well. This would be a fascinating effect to study further.

The researchers of this study did encounter one unanticipated surprise with the project. The decision for the project to be development of a mobile app raised several unanticipated issues. One was that students planned apps were often too ambitious, something that became apparent during development. Since students were familiar with mobile apps, they inclined to want to develop the quality of

apps that they themselves would use. These were generally beyond the reach of first semester programmers. However, these aspirations were great learning experiences as students settled, then adjusted, their project goals to meet their capabilities and timeframes.

The course itself also presented some challenges. It was definitely more work for the students. Every class required them to be actively involved, prepared at a level to be able to apply new knowledge during class, and to be accountable to their peers for their preparation and competence. Application of concepts is a higher level of learning than knowledge acquisition and requires more effort (Bloom et al., 1956). There was no just showing up for class or spending class time playing on their phones. Students viewed class time as a precious commodity. In many ways, the course was also more work for the faculty.

The faculty had to create the teaching and demonstration videos and organize the elements of the course ahead of time so students could choose between options and alternatives. Additionally, with each team making unique apps and adding different elements to their apps, the scope of questions and programming problems for which the faculty was asked for help was quite broad. However, this gave the faculty many opportunities to model problem-solving approaches. Finally, letting go of the control that teachers traditionally have in the classroom required the faculty to adapt to a new mindset. However, any initial discomfort the faculty had initially was assuaged by the obvious high engagement and SOL of the students.

Just as students became more proficient with this new way of learning as the semester progressed, over time the faculty's skills progressed with the new strategies to maximize student control, encouraged student ownership of their own learning, and worked effectively in a flipped environment. Below the researchers present what was learned in this process in the hope that it will be helpful to others.

- Trust yourself and your instincts. Every class is different, and your teaching methods must be fine-tuned to fit your specific situation.
- Be prepared for student resistance, especially initially as they acclimate to the course.
- Make clear connections to the learning in the first half of the semester (the basics) to the project.
- Be supportive as this process requires more work from the students than they may be accustomed to. The researchers of this study found that keeping the workload reasonable for the first half of the semester kept students from dropping, once the students began the project; their feelings of ownership over-shadowed their concerns about workload.
- Be flexible. Regularly solicit feedback from your students; when students feel a sense of ownership of a course, they may offer suggestions for how the learning environment could better serve them. Seriously consider their suggestions and do not be afraid to make adjustments during the semester.
- Provide frequent feedback and much encouragement. Make expectations very clear. These students are not only learning a new subject, they are also most likely learning a new classroom paradigm.

For teachers, the objective is that students learn what is available to them in a course. While the researchers of this study saw the complexity and quality of the apps increase, they have not yet studied whether this is the effect of SOL or other factors. The next step is to continue to offer the course but begin to quantify the changes that are taking place. A pre- and post-test model to examine the change in students with respect to SOL, ownership, self-identify, attitude, and other related factors would be the next step. The researchers of this study plan to go this route and look at the changes longitudinally.

The researchers of this study are currently experimenting with tools to measure the elements of SOL in the context of a programming course. The plan is to repeat the course, doing a pre and post measurement of SOL using a reliable and valid tool adapted from one developed by Thibodeaux et al. (2019a). The researchers of this study want to closely examine the development of self-efficacy, effectance motivation, and self-identity in Student Ownership of Learning. How ownership in an educational setting happened would be very interesting to look at and could lead to more understanding of a deeper type of learning.

Another important question to study is how to measure the deeper learning that the researchers of this study believe drives Student Ownership of Learning. Research needs to be done to investigate and tease out the factors involved in the nature of the learning that happens in SOL-facilitated courses. While it may be that traditional and SOL-facilitated learning support acquisition of knowledge and comprehension to the same extent, purposefully increasing SOL may cause more learning to happen at the higher levels of concept application, general problem-solving, creative thinking, and collaboration that is retained longer.

## REFERENCES


Beggan, J. K. (1992). On the social nature of nonsocial perceptions: The mere ownership effect. *Journal of Personality and Social Psychology*, *62*(16), 229–237. https://doi.org/10.1037/0022-3514.62.2.229

Belk, R. W. (1988). Possessions and the extended self. *Journal of Consumer Research*, *15*(2), 139–168. https://doi.org/10.1086/209154

Bergmann, J., & Sams, A. (2012). *Flip your classroom: Reach every student in every class every day*. International Society for Technology in Education (ISTE).

Bloom, B. S., Englehard, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). In D. McKay (Ed.), *Taxonomy of educational objectives: The classification of educational goals* (2nd ed.). Addison-Wesley Longman Ltd.

Broadwell, M. M. (1969, February 20). Teaching for learning (XVI.). *The Gospel Guardian*, *20*(41), 1–3a. http://www.wordsfitlyspoken.org/gospel_guardian/v20/v20n41p1-3a.html

Brown, N.C.C., Sentence, S., Crick, T. & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE), 14*(2), 1-22. https://doi.org/10.1145/2602484

Chan, P., Graham-Day, K., Ressa, V., Peters, M., & Konrad, M. (2014). Beyond involvement: Promoting student ownership of learning in classrooms. *Formative Instructional Practices*, *50*(2), 105-115. https://doi.org/10.1177/1053451214536039

Computer Science Teachers Association (CSTA). (2017). *CSTA k-12 computer science standards* (revised 2017). http://www.csteachers.org/standards

Dewey, J. (1916). *Democracy and education: An introduction to the philosophy of education.* Macmillan Publishing. https://psycnet.apa.org/record/2003-00119-000

Dittmar, H. (1992). *The social psychology of material possessions: To have is to be*. Harvester Wheatsheaf and St. Martin's Press. http://sro.sussex.ac.uk/id/eprint/66565

Dounas-Frazer, D., Rios, L., & Lewandowski, H. (2019, July). Preliminary model for student ownership of projects. *Physics Education Research (PER) Conference Series*. Provo, UT. https://doi.org/10.1119/perc.2019.pr.Dounas-Frazer

Dudley-Marling, C., & Searle, D. (Eds.) (1995). *Who owns learning? Questions of autonomy, choice, and control.* Heinemann Educational Books. https://eric.ed.gov/?id=ED375388

Durkheim, E. (1957). *Professional ethics and civil morals* (Translated by C. Brookfield). Routledge; Kegan Paul, Ltd. https://doi.org/10.4324/9780429452901

Eckerdal, A. (2009). *Novice programming students: Learning of concepts and practice* [Doctoral Thesis. Sweden: Uppsala University]. https://www.diva-portal.org/smash/get/diva2:173221/fulltext01.pdf

Ellwood, C. (1927). *Cultural evolution: A study of social origins and development*. Century. https://psycnet.apa.org/record/1927-10509-000

EURACTIV (2015, October 16). *Infographic: Coding at school – How do EU countries compare?* https://www.euractiv.com/section/digital/infographic/infographic-coding-at-school-how-do-eu-countries-compare/

Figueiredo, J., & García-Peñalvo, F. J. (2019, October). Teaching and learning strategies of programming for university courses. *Proceedings of the 7th International Conference on Technological Ecosystems for Enhancing Multi-culturality* (pp. 1020–1027). León, Spain: ACM. https://doi.org/10.1145/3362789.3362926

FLIP Learning. (2014, March 12). *Definition of flipped learning*. https://flippedlearning.org/definition-of-flipped-learning/

Foertsch, J., Moses, G., Strikwerda, J., & Litzkow, M. (2002). Reversing the lecture/homework paradigm using eTEACH® web-based streaming video software. *Journal of Engineering Education*, *91*(3), 267-274. https://doi.org/10.1002/j.2168-9830.2002.tb00703.x

Formanek, R. (1994). Why they collect: Collectors reveal their motivations. In S. Pearce (Ed.), *Interpreting objects and collections* (pp. 327–335). Routledge. https://psycnet.apa.org/record/1992-01508-001

Freeman, S., Eddy, S., McDonough, M., Smith, M., Okoroafor, N., Jordt, H., & Wenderoth, M. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, *111*(23), 8410-8415. National Academy of Sciences. https://doi.org/10.1073/pnas.1319030111

Furby, L. (1976). The socialization of possession and ownership among children in three cultural groups: Israeli kibbutz, Israeli city, and American. In Modgil, S. & Modgil, C., Piagetian research: Compilation and commentary (Vol. 8, pp. 95–127). Windsor, England: National Foundation of Educational Research.

Gallo, P., Corritore, C., Wichman, C., & York, A. (2015). Trusting the simplicity of the ultimate question: A customer satisfaction approach to student evaluation of teaching. *Journal of Innovative Education Strategies*, *4*(1), 51–66. http://www.intl-academy.org/wp-content/uploads/2017/04/Paper-4-Trusting-the-Simplicity-of-the-Ultimate-Question-A-Customer-Satisfaction-Approach-to-Student-Evaluations-of-Teaching-Peter-Gallo-JIES-Vol-4-No-1-September-2015.pdf

Gannod, G. C., Burge, J. E., & Helmick, M. T. (2008). Using the inverted classroom to teach software engineering. In *30th ACM/IEEE International Conference on Software Engineering* (pp. 777-786). Leipzig, Germany: ACM/IEEE. https://doi.org/10.1145/1368088.1368198

Hadjerrouit, S. (2005). Designing a pedagogical model for web engineering education: An evolutionary perspective. *Journal of Information Technology Education: Research*, *4*, 115–140. https://doi.org/10.28945/268

Hadjerrouit, S. (2008). Towards a blended learning model for teaching and learning computer programming: A case study. *Informatics in Education*, *7*(2), 181–210. https://doi.org/10.15388/infedu.2008.12

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments*, *1*(1), 1–32. https://doi.org/10.1080/1049482900010102

Hodge, A., Love, B., Grandgenett, N., & Swift, A. W. (2014). A flipped classroom approach: Benefits and challenges of flipping the learning of procedural knowledge. In P. R. Lowenthal, C. S. Yorkand, & J. C. Richardson (Eds.), *Online learning: Common misconceptions, benefits and challenges* (49–59). Nova Science Publishers.

Honebein, P. (1996). Seven goals for the design of constructivist learning environments. In B. Wilson (Ed.), *Constructivist learning environments: Case studies in instructional design* (pp. 11-24). Educational Technology Publications. http://studentcenteredlearning.pbworks.com/f/DesignConstructivistHonebein.pdf

Honebein, P., Duffy. T., & Fishman, B. (1993). Constructivism and the design of learning environments: Context and authentic activities for learning. In T. Duffy, J. Lowyck, & D. Jonassen (Eds.), *Designing environments for constructive learning* (pp. 87–108). Springer-Verlag. https://doi.org/10.1007/978-3-642-78069-1_5

Isaacs, S. (1933). *Social development in young children*. Routledge; Kegan Paul Limited. https://doi.org/10.1111/j.2044-8279.1933.tb02920.x

James, W. (1890). *The principles of psychology (volume 1)*. Henry Holt and Co. https://doi.org/10.1037/10538-000

Johnson, L., & Renner, J. (2012). *Effect of the flipped classroom model on secondary computer applications course: Student and teacher perceptions, questions and student achievement* [Unpublished Doctoral Dissertation]. Louisville, KY: University of Louisville. http://www.academia.edu/download/38862495/Flipped_Classroom.pdf

Kearsley, G., & Shneiderman, B. (1998). Engagement theory: A framework for technology-based teaching and learning. *Educational Technology*, *38*(5), 20–23. https://www.jstor.org/stable/44428478

Lage, M., Platt, G., & Treglia, M. (2000). Inverting the classroom: A gateway to creating an inclusive learning environment. *The Journal of Economic Education*, *31*(1), 30–43. https://doi.org/10.1080/00220480009596759

Loftsson, H., & Matthíasdóttir, Á. (2019). Using flipped classroom and team-based learning in a first-semester programming course: An experience report. *Proceedings of 2019 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 25-31). Australia: IEEE. http://www.ru.is/faculty/hrafn/papers/flipped-classroom-programming-final.pdf

Love, B., Hodge, A., Grandgenett, N., & Swift. A. W. (2014). Student learning and perceptions in a flipped linear algebra course. *International Journal of Mathematical Education in Science and Technology*, *45*(3), 317–324. https://doi.org/10.1080/0020739X.2013.822582

Marshall, H. W., & DeCapua, A. (2013). *Making the transition to classroom success: Culturally responsive teaching for struggling language learners*. University of Michigan Press.

Milner-Bolotin, M. (2001). *The effects of topic choice in project-based instruction on undergraduate physical science students' interest, ownership, and motivation* [Doctoral Dissertation. Austin, TX: University of Texas at Austin]. http://hdl.handle.net/2152/10544

Mok, H. N. (2014). Teaching tip: The flipped classroom. *Journal of Information Systems Education*, *25*(1), 7-11. http://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=3363&context=sis_research

Musib, M. (2014). Student perceptions of the impact of using the flipped classroom approach for an introductory-level multidisciplinary module. *CDTL Brief*, *17*(2), 15-20. https://pdfs.semanticscholar.org/cff8/07b691f9dd92eb1b640f55d981a83088e201.pdf

Nuttin, J. M., Jr. (1987). Affective consequences of mere ownership: The name letter effect in twelve European languages. *European Journal of Social Psychology*, *17*(4), 381–402. https://doi.org/10.1002/ejsp.2420170402

Pacheco, A., Gomes, A., Henriques, J., de Almeida, A., & Mendes, A. (2008). Mathematics and programming: Some studies. *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing* (pp. V.15–1). ACM. https://doi.org/10.1145/1500879.1500963

Papadopoulos, C., & Roman, A. S. (2010). Implementing an inverted classroom model in engineering statics: Initial results. In *Proceedings of the 40th ASEE/IEEE Frontiers in Education Conference* (pp. 15.679.1–27). Louisville, Kentucky: American Society for Engineering Statistics. https://peer.asee.org/16768

Pierce, J., Kostova, T., & Dirks, K. (2003). The state of psychological ownership: Integrating and extending a century of research. *Review of General Psychology*, *7*(1), 1089–2680. https://doi.org/10.1037/1089-2680.7.1.84

Rainer, J. D., & Matthews, M. W. (2002). Ownership of learning in teacher education. *Action in Teacher Education*, *24*(1), 22–30. https://doi.org/10.1080/01626620.2002.10463264

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Rochberg-Halton, E. (1980). *Cultural signs and urban adaptation: The meaning of cherished household possessions* [Doctoral Dissertation. Chicago, IL: University of Chicago]. https://psycnet.apa.org/record/1981-51420-001

Sharp, J., & Sharp, L. (2017). A comparison of student academic performance with traditional, online, and flipped instructional approaches in a C# programming course. *Journal of Information Technology Education: Innovations in Practice*, *16*, 215-231. https://doi.org/10.28945/3795

Thalluri, J., & Penman, J. (2016). To flip a class or not to flip a class: That is the question. *Proceedings of the 2016 Informing Science & IT Education Conference (InSITE 2016)* (pp. 147-157). https://doi.org/10.28945/3414

Thibodeaux, T., Harapnuik, D., & Cummings, C. (2019a). Student perceptions of the influence of choice, ownership, and voice in learning and the learning environment. *International Journal of Teaching and Learning in Higher Education*, *31*(1), 50-62. https://files.eric.ed.gov/fulltext/EJ1206966.pdf

Thibodeaux, T., Harapnuik, D., & Cummings, C. (2019b). Student perceptions of the influence of the COVA learning approach on authentic projects and the learning environment. *International Journal on E-Learning*, *18*(1), 79-101. http://www.academia.edu/download/60488834/Published_20190904-88092-1mp8kny.pdf

Warter-Perez, N., & Dong, J. (2012). Flipping the classroom: How to embed inquiry and design projects into a digital engineering lecture. In *Proceedings of the 2012 ASEE PSW Section Conference*. California: American Society for Engineering Education (ASEE). http://curtbonk.com/pdfs/10B_35_ASEE_PSW_2012_Warter-Perez.pdf

White, R. W. (1959). Motivation reconsidered: The concept of competence. *Psychological Review*, *66*(5), 297–330. https://doi.org/10.1037/h0040934

Xu, S., & Rajlich, V. (2004). Cognitive processes during program debugging. In *Proceeding of 3rd IEEE International Conference on Cognitive Informatics*. British Columbia, Canada: IEEE. https://doi.org/10.1109/COGINF.2004.1327473

Yang, C. C. R. (2017). An investigation of the use of the 'flipped classroom' pedagogy in secondary English language classrooms. *Journal of Information Technology Education: Innovations in Practice*, *16,* 1-20. https://doi.org/10.28945/3635

Ying, K. M., Pezzullo, L. G., Ahmed, M., Crompton, K., Blanchard, J., & Boyer, K. E. (2019). In their own words: Gender differences in student perceptions of pair programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*, (pp. 1053–1059). Minneapolis MN: ACM. https://doi.org/10.1145/3287324.3287380

Zhang, X., Zhang, C., Stafford, T., & Zhang, P. (2019). Teaching introductory programming to IS students: The impact of teaching approaches on learning performance. *Journal of Information Systems Education*, *24*(2), Article 6. http://jise.org/Volume24/n2/JISEv24n2p147.pdf

Ziegel, M. (2004). Preparing teachers for the challenges of technology integration. *Issues in Informing Science and Information Technology*, *1*, 105-113. https://doi.org/10.28945/723

# APPENDICES

## *APPENDIX A: PROJECT DESCRIPTION: GRADING, MILESTONES, PRESENTATION GIVEN TO STUDENTS*

**Project & Milestones Description**

**The project will be evaluated on several aspects. Here is the breakdown. You will be given a grade at each milestone as well as for the final project presentation and your final project.**

| | |
|---|---|
| Milestone 1 | 10% |
| Milestone 2 | 10% |
| Milestone 3 | 10% |
| Final Project Deliverable | 65% |
| Final Project Presentation | 5% |

**Breakdown of Final Project Deliverable (65% of total project grade)**

60% works on device chosen as target (iPhone and/or Android, iPad), and all functionality in place

30% level of difficulty of what you did – brought in new features, new ideas

10% creativity – clever idea, interesting implementation, great design

**Milestones & Individual Participation Summaries**

If you are working with someone else on the project, also due for each milestone is an **Individual Participation Summary,** a Word doc that simply lists what each person did for this leg of the project (include yourself). This information could include the number of meetings attended as well as specific activities carried out (e.g. Mary and I built the initial framework; Joe refined the first two screens, etc.). Turn these in using BL. This is my way of understanding the contribution of each person to the project. I will use this, along with my observations, to adjust individual grades up or down.

**Grading (as per syllabus):**

**A category:** All of B category plus is an **innovative, creative project, ideas, and implementations that go beyond the information in the course**. Demonstrates a lot of outside the box thinking, really dazzling.

**B category:** clear, organized, professional, and complete. Includes all aspects of the required materials/points to be made.

**C category:** incomplete, unorganized, ad-lib in nature goes overtime allowed, incomplete. Partial credit based on extent and nature of problems. No new ideas incorporated. Have few if any professional aspects.

**Milestone 1:** Present an overview of your project idea to the class (5 min). This presentation is **informal**, but should be professional, informative, and complete. The purpose is to obtain feedback from myself and your classmates that you can use to refine your project. You must address these points:

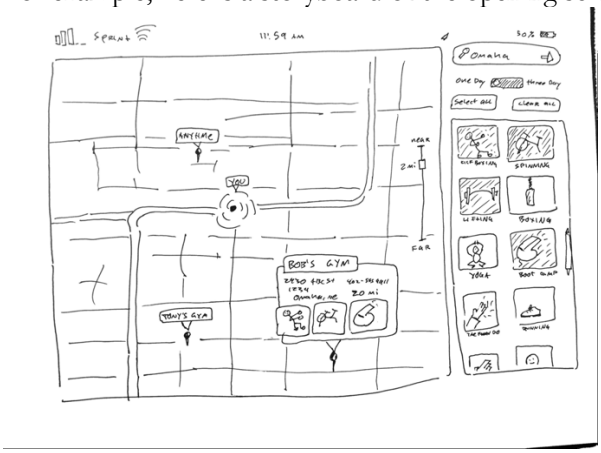
1. Describe the problem you are addressing in detail – what your app will solve.
2. Describe (in detail) the target audience for your app.
3. Provide a high-level description of the app (not a feature list). High-level description means how you would describe it to a friend. For example, "our app helps you to manage a to-do list for each of your classes."
4. Short discussion about any apps out there like this (that's OK if there are) and how yours may differ or be better.
5. Any other items you want to include making your project case **clear and compelling, unique, cool.** This is the part that is going to make your project either an A or a B (or below) project - be creative, think outside the box!
6. Each person hands in your Individual Participation Summary (via BL) - this is simply a Word doc (see above).

**Milestone 2 (10%):** Present an overview of your second milestone. This is an **informal** presentation to me only during class at your desk (5 min). The 'presentation' should be organized, professional, informative, complete, creative, and compelling. The purpose is to obtain feedback from myself that you can use to refine your project. Your presentation must cover the points below (use these as headings).

1. Revisions Made: Note revisions you have made to your project from Milestone 1.
2. Show me algorithms that you will use to develop code. Make an algorithm for each major function or action for which you will be writing code. Remember, algorithms are NOT code, but rough ideas written out in English that code sometimes creeps into just because you know how to code (kind of a short-hand). These are **rough outlines** you can then use to write code. These will be hand-written. Example algorithm to calculate the average of two numbers:

1. Make a function that takes two numbers, adds them, divides by 2, and returns the answer
2. Obtain two numbers from the user – two Alert windows or two prompts?
3. Convert the numbers from text to numbers, if necessary (toString)
4. Call the function and pass it the two numbers
5. Catch the return number from the function
6. Display the answer – alert ("the average of x and y is z")

3. Show me sketches of storyboards of your app – rough sketches of how the screen will look at different points in time – these are very rough – like cocktail napkin sketches. For example, here is a storyboard of the opening screen of an app under design.




4. What are the **three** main things a user will do with your app?
5. List of 5 potential problems or hard parts that you anticipate as you develop your app. Write these down.
6. How you anticipate organizing your app development within your team, and the platform you are developing for (iPhone/Android, Blackberry, iPad).
7. Hand in your Individual Participation Summary (via BL)

**Milestone 3 (10%):** This milestone is an **informal** presentation to me (5 min), and should be professional, informative, complete, creative, and compelling. The purpose is to obtain feedback from myself that you can use to refine your app. At this point, your app should be at least 50% coded. Demonstrate using your emulator if you used one. At a minimum, it should be working on the iPhone emulator (others just extra points). Your presentation must cover these points.

1. Revisions Made: Note revisions you have made to your project from Milestone 2.
2. Present and discuss the parts of the app you have completed as well as those that are in skeleton form.
3. Walk me through an example of how a user would complete one of the three main tasks you identified in Milestone 2 (even if some parts of the app aren't done – just talk me through those).
4. Identify briefly what you have left to do (this may overlap with #2 above) and your plan of how to complete it.
5. Hand in your Individual Participation Summary (via BL)

**Final Project Presentation Day**

You will turn in three deliverables due at the time of your final presentation:

1. Your app (all of the web pages and code, packaged into one zipped file) – export it and put it on a cd, as well as upload the zipped file to BL.
2. Executive Summary – Word doc. Turn it into BL.
3. Peer evaluation (of entire project). There is an Excel form in the Files > Project folder. Evaluate your partner AND YOURSELF for the entire project – BE OBJECTIVE and support your scores with narrative. Turn it in using BL.

**Presentation**

Your final project presentation is **formal** – your timeframe will be given by your instructor. This includes an allowance of a 5-minute question period, and a 5 min. demo time in which the class can use your app on their own devices. That means you must post your app and instructions for how to 'get' it before class (use FB).

Practice your presentation several times. Plan a snappy, to-the-point talk with an organized, focused demo session.

Your presentation will be graded on the following criteria:

· Keeping to your time frame, with demo and Q&A.

· Covering all required elements (see below).

· Quality of the demo.

· Professional quality of the visual presentation itself.

· Professional demeanor and dress of the presenter(s) – if you are working by yourself, get someone else to run your computer while you present.

· How well you answer questions.

**Required Presentation Elements**

a. **Introduction:** Team name, members, statement of problem your app addresses, target audience.

b. **App Presentation:** Present your app, demonstrate the main features, pointing out how you implemented them (i.e. I used an onclick event here to …..). Walk us through at least 2 actions that your user would do with your app. Demonstrate native features you incorporated.

   This is your opportunity to demonstrate your knowledge of the concepts of this course – impress us!

c. **Hands-on Demo:** Make sure the class has installed your app on their device of choice before class. Now walk the entire class through your app in a kind of overview of its main features (keep an eye on time here).

d. **Q&A:** you know I will have at least always have questions!

**Executive Summary**

This is a one-page summary of your project and app. Since it is short, you have to distill down everything you have done and what you have produced – so conciseness and clarity in your writing are key.

Give enough detail that it makes sense, tells a clear story, but does not elaborate to the extent of being repetitive and verbose. Here is the format:

**1. Title Page**

Give the report a title that reflects the nature of the project, with the course name, the app name, your section (early or late) and your name(s) on it.

**2. Executive Summary**
Include a brief description of the aim of the project, what you did, and the main outcomes. Devices the app works on, and how the app can be installed. Include any information needed to install the app to an iPhone, an Android phone, and an iPad. Summarize key issues you had, and point out clever or unique things you did, as well as necessary concepts you used (i.e. DOM, events, errors, etc.). Discuss the native features you used.

## *APPENDIX B: EXAMPLES OF PROJECT APPS*

1. *City Slicker*: a grocer shopping budget helper, with the intent of incorporating local grocery sales.
2. *College Cookbook*: a recipe app that accesses recipes for college students in a database.
3. *Creighton University Athletics;* app that could be used to record baseball statistics during a game by the Athletics Department.
4. *Creighton University Fitness*: app to help students design fitness workouts depending on their needs and personal physique.
5. *Drive Safe*: app that helped students who had had too much to drink to not drive. It kept track of drinks, emergency friend called, Uber calling, blocked sending text messages.
6. *Grade Calculator*: student could enter grades obtained on assignments and test, and their course grade would be calculated
7. *Ideal Gift*: kept track of people and what they wanted; also got information from Amazon when there was a sale on any of these items.
8. *Opportunity Calendar*: calendar that updated activities on campus to students' personal calendar and provided way for student to communicate about events with friends.

## *APPENDIX C: EXAMPLE OF TYPICAL HOMEWORK ASSIGNMENTS*

**Assignment 6: Loops, Functions and Arrays**

For this assignment you will be turning in some of the work you complete with Code Academy. You will also be creating some extra web pages to turn in.

**Process**: Complete the work in CodeAcademy.com for the unit indicated. When done, copy the code, paste it into a Notepad (Windows) or the Text Editor (Mac) doc - save as a .html document. Then upload it to your online web hosting account. (don't forget to upload your homepage also). Add the url to the link on your homepage. When you are all done, create a homepage for this assignment that has links to all of the web pages for this assignment. Turn in the url of this homepage for grading.

**Assignment**

0. First, in your web hosting account, make a folder named ass6 (for assignment 6). All of the work you do for this assignment will go into this folder.

**Part 1 – Tues**

**In Code Academy> JavaScript**

1. Complete Unit 2 Functions > Build Rock, Paper, Scissors and use this for one of your web pages. Name this page rps_game.html.

2. Complete Unit 3 > 'For' Loops in JavaScript> Search Text for your Name and use this for one of your web pages. Name this page namesearch.html.

3. Make a homepage for this assignment that has the assignment name and your name on it in Notepad (Windows) or TextWrangler (Mac) saved as a .html document. Put a link to each of your web pages that you made for this assignment (including the Extra Credit if you did that). Name this webpage index.html.

**Part 2 – Tues**

**In Code Academy>JavaScript**

1. Complete Unit 4 > 'While' Loops in JavaScript> Dragon Slayer! and use this for one of your web pages. Name this page dragonslayer.html.

2. Complete Unit 5 > Control Flow > Choose your Own Adventure 2! and use this for one of your web pages. Name it adventure2.html.

3. Put links to these two web pages on your homepage for this assignment.

**Optional Extra Credit (10 pts)**
1. **Find Max** - Get two numbers from the user. Create a function that takes two arguments (these two numbers) and returns the largest one. Outside of the function, write the largest one to the screen in this format:

The largest of your two numbers, x and y, is y. // where x and y are their two numbers, of course

Name this webpage findmax.html. Put a link to it on your ass6 homepage, and indicate it is Extra Credit.

2. **Backwards** - create an array of numbers 50 - 100, in increments of 5 by using a For loop (e.g.: 50, 55, 60, etc). Then create a function named backwards that takes an array as an argument, prints out the numbers backwards (e.g. 100, then 95, etc). Name this webpage backwards.html. Put a link to it on your ass6 homepage, and indicate it is Extra Credit.

3. **Wheel of Fortune** - Make a new game - the Wheel of Fortune Game. It takes two people. The game asks user1 (input via popup) for a word. Then it asks for the number of letters in the word. This is the word to be guessed by user2 player. Now, the computer displays how many letters the word is, then waits for the first letter guess. User1 inputs a letter. The computer says "wrong, give me another letter, and you have X guesses left" or, if the letter was in the word, the computer says "Yes, i is the fourth letter in the word". The user will have to keep track of the word and the letters guessed (the computer will not). You will need these functions:

> function locateALetter (word, letter) < this takes the current word and letter, and sees if the letter is in the word.
>
> function getALetter < this function gets a letter from the user and saves it to a variable
>
> function getAWord < use to start the game

Use a Switch to output the correct message to the user based on their guess (eg. Letter is x$^{th}$ in the word, or Wrong or you won!).

You may have to add additional logic and code. This is just to get you started.

## *APPENDIX D: GRADING AND DESCRIPTIONS OF COURSE WORK*

**Grading**

| | **Percentage** |
|---|---|
| **Midterm Exam** | 15% |
| **Assignments** | 35% |
| **Project** | 35% |
| **Engagement** | 15% |

Here are the overarching grading criteria for all work in this course. This is in addition to the specific requirements of each item.

| | **A** | **B** | **C** |
|---|---|---|---|
| Content elements (95%) | Contains all elements required by assignment. Shows creativity, works correctly and completely. | Missing less than 10% of elements or functionality required by assignment. Little or no creativity or initiative displayed. | Missing more than 10% of elements or functionality of assignment. |
| Quality (5%) | Assignment well done, organized, shows extra work above and beyond what assignment required, professional, creative. | Assignment fulfills requirements completely with few to no extra work. | Assignment elements and functionality in place but work is messy, shoddy, thrown together and shows little effort, unprofessional. |

**Late Assignments**

Individual assignments will be assessed up to 25% off for each day they are late (weekends and holidays will be counted), up to 2 days late. After two days, NO late individual assignments will be accepted for any reason.

**Assignments**

1. **Midterm Exam.** This exam is a combination of what you know and how you apply it. It will involve answering questions and writing code.
2. **Assignments.** Assignments that build on the content and work done in class will be assigned intermittently. These are opportunities for you to bring several skills together and show what you can do with the problem scenario posed in the assignment. The grading criteria are shows earlier in this syllabus (see above). Don't forget that creativity can count to raise your grade, it needs to work to get an A, and there are MANY ways to do just about

anything in development! It is a good idea to work with someone else when you are writing code – but each of you needs to do your own work. Don't cut and paste each other's code, or you will also have to split the points for the assignment.

3. **Project.** The course has a significant project, which will involve creating an app that not only works but works well and is complex. You will work on this for several weeks, towards the end of the semester. It will require all of the skills you have learned in the class, and will involve a formal presentation. While the design of your app will count wrt creativity, you will be primarily graded on whether it works (see grading criteria above). This project may be individual or with teams – ask your professor which one will be done in your class.

   Specific guidelines and grading criteria will be provided in a Project Documentation doc. Basically, the overall project grade takes into account the quality and completeness of the project as well as how well the concepts of the course are applied, classmate and my input as to effectiveness in the team, peer evaluations (your contribution), and presentation of the project. There may be Milestones that are graded as the project moves forward. If the project is done with teams, y**our individual grade for your team project (all parts) will be a mix of the project grade and your effort/participation in the project. This will be determined by intermittent peer evaluations from your team members and my professional observations. Note that a team member may be fired by the team for non-performance. I must be involved in this decision as arbitrator.**

4. **Engagement.** Course engagement is vital; that means your 'attendance' and active participation is necessary. I will use my judgment to evaluate your involvement and quality of participation in the course. I will use these criteria to guide my focus on your engagement.

| | | | |
|---|---|---|---|
| Positives | Always (3) | Occasionally (2) | Rarely (1) |
| 1. Enters into class discussions and activities with quality input. | | | |
| 2. Offers questions or comments during activities and on Facebook. | | | |
| 3. Interacts with professor and classmates outside of scheduled activities to obtain clarification, enrichment, both in person and on Facebook. | | | |
| 4. Interacts with professor and classmates to clarify ideas, offer questions or comments both in person and on Facebook. | | | |
| 5. Engages in the group activities and communication tools as needed. | | | |
| 6. Finds and shares additional information about a topic both in class and on Facebook. | | | |
| 7. Contributes to class in meaningful ways; with comments, questions, helping others, answering questions on Facebook. | | | |
| 8. Class and Facebook contributions of high quality and relevant. | | | |
| 9. Adds terms and defines terms in Facebook Doc. | | | |

| | | | |
|---|---|---|---|
| 10. Completes work (in-class, assignments) consistently on time. | | | |
| Negatives | Always (-3) | Occasionally (-2) | Rarely (-1) |
| 1. Skips class. | | | |
| 2. Interacts intermittently, infrequently in-person and on Facebook. | | | |
| 3. Exhibits disruptive behavior. | | | |
| 4. Never logs in to course information system or Facebook. | | | |
| 5. Rarely posts in Facebook group or helps classmates. | | | |
| Scale:<br>A: 25-30 + my judgment B: 20-25 + my judgment C: 19 and below + my judgment | | | |

## *APPENDIX E: COURSE SCHEDULE*

| | | |
|---|---|---|
| **Aug** | | |
| Week 1 | | |
| **8/22 Thurs** | **Do before Class**<br>1. iPhone and iPad owners - bring your device to class and bring your Device ID (directions here on how to get it from your device).<br>3. Install Google Chrome, Mozilla Firefox, and the add-in Firebug for Chrome and for Firefox (your Mac or PC).<br>4. Get a Google account if you don't have one already (your Gmail account is a Google account).<br>5. Get an account on CodeAcademy.com - use your name for the account so I can tell it is yours (you will be handing in some homework using this tool).<br>6. Join class Facebook space (anyone who is a member can approve membership requests from other class students). | **In Class**<br>1. Welcome to BIA 375 and more with ZeFrank frog<br>2. Format of the class: read, listen, practice before class. Then apply concepts in harder work in class, finish after class for homework as needed.<br>FB for questions, announcements.<br>WebEx for online seminars, meetings.<br>4. Experience Survey - HTML, CSS, JS, FTP, DB |
| | **New Assignments** (look under Syllabus on Menu for listing of all due dates)<br><br>1. Assignment 1: Facebook Introductions<br>2. Assignment 2: Google Website<br>3. Assignment 3: App Project Brainstorming | |

| | | |
|---|---|---|
| Week 2 | | |
| **8/27 Tues** | **Do before Class**<br>1. nothing today | **In Class**<br>No formal class today - work with your team and on your own on Assignments 1-3 |
| **8/29 Thurs** | **Do before Class**<br>1. nothing today | **In Class**<br>No formal class today - work with your team and on your own on Assignments 1-3. |
| | **New Assignments**<br><br>No new assignments | |
| Week 3 | | |
| **9/3 Tues** | **Do before Class***<br>1. nothing today (build on what you started last week - project brainstorming, HTML homework) | **In Class**<br>1. Some of Last year's projects - scan QR codes with your phone to run them (30 min)<br><br>2. Project Brainstorming - finish up (15 min). Pick main ones - list what you need technically (eg. Database)<br><br>3. Work on HTML homework - finish up, work on problems. |
| **9/5 Thurs** | **Do before Class**<br>*you will be assessed and graded on these at the beginning of class - see PreClass 1 Assignment on right (we'll do this In Class)<br><br>1. Readings<br><br>a. Introduction to CSS Style Sheets (Links to an external site.)(Note: there are three ways to structure and use CSS with a webpage - we are using the separate stylesheet file method).<br>b. How to add CSS to a Google Sites webpage (Links to an external site.) | **In Class**<br>1. PreClass 1 Assessment (graded)<br>2. Project Brainstorming Presentations (informal - no ppts needed; class provides feedback). Auction next Tues - team who came up with idea has first chance.<br>3. Ass 4: CSS and Styling (Code Academy) - work on this in class<br>**** if you finished this already**, go to the Extra Credit at the end of this assignment and finish that in class today. |

| | | |
|---|---|---|
| | 2. Lecture Vodcasts -<br><br>a. Web CSS: HTML Attributes and Introduction to Styles (Links to an external site.) ***note** - HTML attributes are being phased out and replaced by styles<br>b. Web CSS: DIVs, IDs, and Classes (Links to an external site.) | |
| | **New Assignments**<br><br>1. Finish Ass 4: CSS and Styling | |
| Week 4 | | |
| **9/10 Tues** | **Do before Class***<br>*you will be assessed and graded on these at the beginning of class - see PreClass 1 Assignment on right (we'll do this In Class)<br><br>1) CodeAcademyJavaScript Unit 1: Introduction to JavaScript> Getting Started with Programming - complete these sections:<br><br>1. Getting to Know You<br>2. Why learn programming?<br><br>2) Readings - WebMonkey tutorial by Thau (Links to an external site.)<br><br>a. Intro http://www.webmonkey.com /2010/02/JavaScript_tutorial/<br>b. Lesson 1 http://www.webmonkey.com /2010/02/JavaScript_Tutorial_-_Lesson_1 | **In Class**<br>1. PreClass 1 Assessment (graded)<br><br>2. JS Pointers<br><br>3. Problem Solving<br><br>4. Assignment 5 Part 1: Intro to JavaScript (JS) |
| **9/12 Thurs** | **Do before Class**<br><br>1) Readings - WebMonkey tutorial by Thau (Links to an external site.)<br><br>a. Lesson 2 (up to 7. Link events) - http://www.webmonkey.com/ 2010/02/JavaScript_Tutorial_-_Lesson_2 | **In Class**<br>1. PreClass 1 Assessment(graded)<br>2. JS Pointers<br>3. Problem Solving<br>4. Assignment 5 Part 2: Intro to JavaScript (JS) |
| | **New Assignments** | |

| Week 5 | | |
|---|---|---|
| **9/17 Tues** | **Do before Class***<br>*you will be assessed and graded on these at the beginning of class - see PreClass 1 Assignment on right (we'll do this In Class)<br><br>1) CodeAcademy<br><br>a. JavaScript Unit 2: Functions<br>> complete section Introduction to Functions in JS<br>b. JavaScript Unit 3: For Loops<br>> complete section Introduction to 'For' Loops in JS<br><br>2) Readings<br><br>a. Functions - http://www.quirksmode.org/js/function.html (Links to an external site.)<br>b. For Loops - http://www.quirksmode.org/js/state.html#for (Links to an external site.)<br><br>3) Install Filezilla Client on your computer for use in class Tues (FTP program for uploading/downloading files) -<br><br>https://filezilla-project.org/download.php?show_all=1 (Links to an external site.)<br><br>4) Resources<br><br>a. JS Loops http://www.echoecho.com/JavaScript9.htm (Links to an external site.) | **In Class**<br>1. PreClass 1 Assessment(graded)<br><br>2. JS Pointers<br><br>3. Filezilla - setup and use with otis accounts, change pw<br><br>4. Assignment 6: Functions and Loops Part 1 |
| **9/19 Thurs** | **Do before Class**<br><br>1) CodeAcademy<br><br>a. JavaScript Unit 4: While Loops in JavaScript<br>> complete section Introduction to 'While' loops in JS<br>b. JavaScript Unit 5: Control Flow<br>> complete section More on Control Flow in JS<br><br>2) Readings -<br><br>a. While loop http://www.tutorialspoint.com/javascript/javascript_while_loop.htm (Links to an external site.)<br>b. Boolean Logic (and/or/not): http://www.quirksmode.org/js/boolean.html (Links to an external site.) | **In Class**<br>1. PreClass 1 Assessment (graded)<br>2. JS Pointers<br>3. Problem Solving<br>4. Assignment 6 Part 2: Functions and Loops |

| | | |
|---|---|---|
| | c. Switch https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/switch (Links to an external site.) | |
| | **New Assignments** | |
| Week 6 | | |
| **9/24 Tues** | **Do before Class*** <br><br> Breather Day - no preps - read over what we are doing in class - be ready! You will be randomly assigned with a partner to one of these to be completed entirely in class. | **In Class** <br> 1. Work on Assignment 7 - Breather Day assignment. **Due by the end of class.** |
| **9/26 Thurs** | **Do before Class** <br><br> 1) CodeAcademy <br><br> a. JavaScript Unit 5: Data Structures > complete section Arrays and Objects in JS <br> b. JavaScript Unit 6: Objects I > complete section Introduction to Objects I <br><br> 2) Readings - <br><br> a. JavaScript Objects movie (7 min) - great one but need good bandwidth <br> b. JavaScript Basics - http://www.htmlgoodies.com/primers/jsp/article.php/3600451/Javascript-Basics-Part-8.htm (Links to an external site.) <br> c. Optional: if you want to hear another person describe JS objects another way: Elegant Code - (read up to Constructors and stop) http://elegantcode.com/2010/11/12/basic-javascript-part-2-objects/ (Links to an external site.) | **In Class** <br> 1. PreClass Assessment <br> 2. JS Pointers <br> 3. Assignment 8 |
| | **New Assignments** | |
| Week 7 | | |
| **10/1** | **Do before Class** <br><br> 1) Readings <br><br> a. JavaScript Events - http://www.elated.com/articles/events-and-event-handlers/Asses | **In Class** <br><br> 1. Assessment (format TBA on FB; <br><br> Assessment link on top of |

| | | |
|---|---|---|
| | b. The DOM model - http://www.elated.com/articles/javascript-dom-intro/ (Links to an external site.)<br>c. More about DOM: read all three http://dom-tutorials.appspot.com/static/index.html (Links to an external site.) (do Tutorial 1: Lessons 1 and 2; Tutorial 2: Lesson 1; Tutorial 3: Lesson 1)<br><br>2) Videos<br><br>a. What's the DOM? (5 min) - plays in BL<br><br>The rest of these videos are .mov format - if they don't play in BL for you, download them and play them on your computer (PC or Mac) - you need QuickTime Player (free) to play them.<br><br>b. Nodes (in DOM) (3 min)<br>c. Working with the DOM -Getting Elements (11 min)<br>d. JS Events Introduction (8 min)<br>e. JS Click-Load Event (7 min)<br>f. JS Focus-Blur Event (2 min)<br>g. JS Timers (6 min)<br><br>3) Resources and References (for future use)<br><br>a. * List of JavaScript Events http://www.web-source.net/javascript_tutorial/javascript_events_reference.htm#.UkXZhGiYYt0 (Links to an external site.)<br>b. Another list of JS Events with examples http://www.koderguru.com/tutorials/javascript/javascriptevents.php (Links to an external site.)<br>c. JS and DOM Reference - short tutorials with examples http://www.htmldog.com/guides/javascript/ (Links to an external site.)<br>d. Details about DOM http://www.w3schools.com/htmldom/dom_intro.asp (look at menu on left side)<br>e. Essentials of DOM and JS in 10 min http://www.youtube.com/watch?v=URF2sVQWuxU | this schedule - opens 2 min before class starts)<br><br>2. Assignment 9: Events and DOM |
| | **New Assignments** | |
| **10/3** | **Do before Class** | **In Class**<br>1) Assessment - log into AC |

| | | |
|---|---|---|
| | 1) ** log into your Application Craft account - and look around. Your login is your CU email (netID@creighton.edu) - pw is my last name with 99 on the end - no caps or space. | and pick one interesting item to tell class about.<br><br>2) Assign. 10 - Intro to AC<br><br>3) Change your AC pw (have to do this in class today) |
| | **New Assignments**<br><br>****** MIDTERM REVIEW** | |
| Week 8 - Oct 7 - 11 | | |
| **10/8** | **Do before Class**<br><br>1) Readings (read AFTER you watch the videos)<br><br>> Standard Mobile Widgets http://www.applicationcraft.com/developers/documentation/product-guide/mobile-apps-sites/other-mobile-widgets (Links to an external site.)<br><br>> Global Data Store (goes with Switching between Apps and Embedding Apps videos)<br><br>http://www.applicationcraft.com/developers/documentation/scripting-apis/client-api/global-data-pool-functions/ (Links to an external site.)<br>> Switch App<br>http://www.applicationcraft.com/developers/documentation/scripting-apis/client-api/app-functions/switchapp/ (Links to an external site.)<br>>parentApp<br>http://www.applicationcraft.com/developers/documentation/scripting-apis/client-api/app-functions/parentapp/ (Links to an external site.)<br>>getGlobalData (it's counterparts, setGlobalData and clearGlobalData are linked at bottom of the page)<br>http://www.applicationcraft.com/developers/documentation/scripting-apis/client-api/global- | **In Class**<br>1. Assessment<br><br>2. download these files for midterm and put on zymic in "midterm" folder that you create. You'll use these on the midterm.<br>I used BL to mail links to the zipped Midterm Exam files and the Midterm Review files:<br><br>http://www.cindycorritore.com/375/midterm13/midterm.zip (Links to an external site.)<br>cindycorritore.com/375/MTReview.zip<br>(Links to an external site.)<br><br>3. Assignment 11 |

| | | |
|---|---|---|
| | data-pool-functions/getglobaldata/ (Links to an external site.)<br><br>2) Videos<br><br>Watch all of the "An Intro to UI Design" videos (there are six; total of about 15<br>min. worth) - might want to take notes.<br><br>> Overview<br>> Pages and Page Navigation<br>> Repeating Widgets across Multiple Pages<br>> Switching Between Apps Seamlessly and the Global Store<br>> Embedding One App Inside Another<br>> Time Saving Tricks and Tips<br><br>3) Resources and References (for future use)<br><br>AC Product Guide - user manual for all of AC http://www.applicationcraft.com/developers/documentation/product-guide (Links to an external site.) | |
| | **New Assignments** | |
| **10/10** | **Do before Class**<br><br>**Study for the Midterm -** practice writing html, css, and JS, putting it on pages, uploading to zymic using FileZilla. Go over notes - few questions on tech language and concepts from class.<br><br>**** You will be using Notepad or TextWrangler (Mac), FileZilla (for FTP), and your Zymic.com website. Your work MUST be uploaded in order to be graded.**<br><br>**(I posted the review again here)**<br><br>**----------**<br><br>**Right before you come to class to take the midterm:**<br><br>**1. Have Notepad or TextWranlger, FileZilla, and a web browser open.**<br><br>**2. Make a folder named "midterm" in your Zymic account (using FileZilla)**<br><br>**3. Download and unzip this file - put it in a folder named biamidterm on your own computer (where you can find it). You can do all of your midterm work from that folder. So have it open before taking the test.** | **In Class**<br>**Midterm -**<br>Open book, computer,notes, online, etc.<br><br>**** You will be using Notepad or TextWrangler (Mac), FileZilla (for FTP), and your Zymic.com website. Your work MUST be uploaded in order to be graded.** |

| | | |
|---|---|---|
| | **4. Using FileZilla, upload the three unzipped files (briards.html, styles, css, and dog.jpeg) into your Zymic account (be sure you can see them using the url so you know they are in the right place) and browse to that webpage in your browser.**<br>**http://www.cindycorritore.com/375/midterm13/midterm.zip (Links to an external site.)**<br>**5. Open the briards.html file on your computer in Notepad or TextWrangler.**<br><br>**6. Open Blueline > Homework.** | |
| | **New Assignments**<br><br>1. Write your reflection for the first half of the semester. I have details here about how to write a reflection, etc. | |

| Week 9 | | |
|---|---|---|
| **10/22** | **Do before Class**<br><br>1) **Readings**<br><br>> Using the Firefox built-in Developer Tools for Debugging JavaScript https://hacks.mozilla.org/2013/09/re-introducing-the-firefox-developer-tools-part-1-the-web-console-and-the-javascript-debugger/ (Links to an external site.)<br><br>>JavaScript Try-Catch statement http://www.impressivewebs.com/javascript-try-catch/ (Links to an external site.)<br><br>2) **Videos**<br><br>a. **Watch all of the "Client-Side JavaScript" videos EXCEPT the last one (Ajax)** - there are five to watch (about 10 min. worth) - might want to take notes.<br><br>> App & Widget Events<br>> The Code Explorer and Editor<br>> The Application Craft Client API<br>> Debugging your Apps - read the first reading above before viewing | **In Class**<br>1. Pre-Class Assessment<br><br>2. Assignment 12 Part 1 (due Weds midnight) - we'll do Part 2 Thurs. |

| | | |
|---|---|---|
| | > Customizing the Error Handling - read the second reading above before viewing<br><br>b. **Watch all of the "Adaptive Layout" videos EXCEPT the last one (Adaptive Rules)** - there are six to watch (about 15 min. worth) - might want to take notes.<br><br>> An Overview of Responsive Design<br>> The Importance of Containers<br>> The Sizes Property Explained<br>> Layout Modes<br>> Widget Alignment<br>> Using Tables in Adaptive Layout<br><br>3) Resources and References - good ones<br><br>> Client API (for Application Craft) - note these sections in particular that list and describe the functions available for different objects in AC:<br>> App functions<br>> Page functions<br>> Widget functions<br><br>> AC Developer Cheat Sheet - one page description of all AC objects and their events | |
| | **New Assignments** | |
| **10/24** | **Do before Class**<br><br>1) **Readings**<br>none<br><br>2) **Videos**<br><br>none<br><br>3) Resources and References - good ones | **In Class**<br>1. Pre-Class Demo (Dr Corritore)<br><br>2. Assignment 12 Part 2<br><br>3. Database Review Packet |
| | **New Assignments** | |
| Week 10 | | |
| **10/29** | **Do before Class**<br><br>1) **Readings** | **In Class**<br>1. Pre-Class Assessment |

| | | |
|---|---|---|
| | > Database Review Packet<br><br>> Create a Database using phpMyAdmin (Links to an external site.) - read the first 3 pages - up to Open and Close a Connection. phpMyAdmin is a program that runs in a web browser written in the programming language php. It is used to access and interact with databases that live on database servers.<br><br>> phpMyAdmin - this is the ide we will be using to interface with our mySQLdatabase: http://www.phpmyadmin.net/home_page/index.php (Links to an external site.) - click on Demo to open the Demo phpMyAdmin - login in with Username: root and no password. You can see on the left all of the databases that have been created on the Demo database server.<br><br>This will start phpMyAdmin that is connected to their many demo databases. Look around, try stuff out. Then watch this video (http://www.youtube.com/watch?v=n7c5zMk8cx4 (Links to an external site.)<br><br>) using the Demo database to work along with the video (instead of the one the video uses).<br><br>2) Videos<br><br>3) Resources and References - good ones | 2.Assignment 13 Part 1 (due Weds midnight) - we'll do Part 2 Thurs. |
| | **New Assignments**<br><br>**1. Project Milestone 1 (due next Tues in class)** | |
| **11/1** | **Do before Class**<br><br>1) **Readings** | **In Class**<br>1. Pre-Class Assessment<br>2. Assignment 13 Part B |

| | | |
|---|---|---|
| | 2) **Videos**<br><br>a. in AC, watch the first four videos in the section Integration. *<br>> An Overview<br>> Setting up a Connection<br>> Querying External Databases<br>> Data Views<br>> Configuring Data Section Properties and Expression Editor<br><br>b. in AC, watch this video from the section Widget Data:<br>> Populating List Widgets with the Population Dialog<br><br>* here is the AC program that are used in these videos. There are two versions: one is done with "minimal" JavaScript, and the other uses 'maximum' JavaScript. To import them into your AC account, log in. Then pick the icon Import - navigate to one of the files (on your computer), pick it, take all the defaults. Note: the Next button on the screens is at the bottom - you have to pull the little white window up to see it.<br>Minimum JavaScript<br>Maximum JavaScript<br><br>3) Resources and References - good ones | |
| | **New Assignments** | |
| Week 11 | | |
| **11/5** | **Do before Class**<br><br>1) **Readings**<br><br>none<br><br>2) Videos<br><br>a. None<br><br>3) Resources and References - good ones | **In Class**<br>1. Milestone Presentations to Class<br><br>2. Review Project<br><br>3. Review Milestone 2 |
| | **New Assignments**<br><br>**1. Project Milestone 2** | |

| | | |
|---|---|---|
| **11/7** | **Do before Class**<br><br>1) **Readings**<br><br>> CRUD http://en.wikipedia.org/wiki/<br><br>(Links to an external site.)<br><br>2) Videos<br><br>a. AC Database - * import this AC file- it is the app used in the videos. We will use it in class.<br>b. in AC, these are under the Video: Server-Side JavaScript - first three movies*<br><br>> A High-Level Overview of Server - Side JavaScript Using AC ** Note: our gator Databases are called Server-Side Databases (they are on a server other than AC's).<br>> An Overview on Coding Up Your Server-Side JavaScript Calls in Application Craft<br>> A Detailed Look at Coding Up Client/Server Side JavaScript Calls in AC<br><br>c. in AC, these are under the Video Integration section.<br><br>> Automated and JavaScript Drill-Down<br>> Containers Explained (this is about data containers)<br><br>3) Resources and References - good ones | **In Class**<br>1. Pre-Class Assessment<br>2. Assignment 14 (finish in class) |
| | **New Assignments** | |
| Week 12 | | |
| **11/12** | **Do before Class**<br><br>Nothing required | **In Class**<br>1. Milestone 2 Due in Class<br><br>2. Milestone 3 review<br><br>3. Work on Project |

| | | |
|---|---|---|
| | **New Assignments**<br>**1. Project Milestone 3** | |
| **11/14** | **Do before Class**<br>Nothing required. | **In Class**<br>1. Work on Project |
| | **New Assignments** | |
| Week 13 | | |
| **11/19** | **Do before Class**<br>Nothing required. | **In Class**<br>1. Work on Project |
| | **New Assignments** | |
| **11/21** | **Do before Class**<br>Nothing required. | **In Class**<br>1. Milestone 3 Due in Class<br>2. Final Project review<br>3. Work on Project |
| | **New Assignments** | |
| Week 14 | | |
| **11/26** | **Do before Class**<br>Nothing required. | **In Class**<br>1. Work on Project |
| | **New Assignments** | |
| **11/28** | **Thanksgiving!** | |
| | **New Assignments**<br>**1. Final Project and Presentation next week** !!! | |
| Week 15 | | |
| **12/3** | **Do before Class** | **In Class**<br>1. Review Presentation for Thurs |

| | | |
|---|---|---|
| | Nothing required. | 2. Finish up Final Project |
| | **New Assignments** | |
| **12/5** | **ARGY-BARGY**<br>**PROJECT PRESENTATIONS**<br><br>Time TBA - both classes present at same time | |
| | **New Assignments** | |
| FINALS WEEK | | |
| **12/9 Mon-day**<br><br>**2PM** | **Do before Class**<br><br>** check BL - you have several assignments due by 2PM today (evaluations, reflection), my custom Course Evaluation.<br><br>** Remember, you must fill out the Creighton College of Business Course Evaluation (you received an email with a link to it) in order for your grade in the course to be released. | |
| | **New Assignments** | |

## BIOGRAPHIES

**Dr. Cynthia Corritore** is a Full rank Professor of Information Systems and Technology in the Heider College of Business, Department of Business Intelligence and Analytics at Creighton University in Omaha, Nebraska. Her Ph.D. degree is in Computer Science from the University of Nebraska - Lincoln, where she specialized in Artificial Intelligence and Human Computer Interaction. Her research interests center around examining the effect of technology on humans. Her publications range from modeling online trust in e-commerce as well as innovative methods of teaching. She has published research in these areas, but most recently has been focusing on teaching pedagogies with innovative technologies.

**Dr. Betty Love** is a Full rank Professor in the Mathematics Department at the University of Nebraska at Omaha. Her Ph.D. degree is in Operations Research from Southern Methodist University where she specialized in the design and implementation of parallel algorithms for network optimization problems. Her research interests have focused on the interface between mathematics and computer science at various levels. Currently, she is the Principal Investigator on a grant from the National Science Foundation to create and evaluate a novel general-education course called Introduction to Mathematical and Computational Thinking that uses programming to teach mathematical concepts.