Revamping the Classroom
Teaching Mobile App Software Development Using Creative Inquiry

Roy P. Pargas[1] and Barbara J. Speziale[2]
[1]*School of Computing, Clemson University, Clemson, SC 29634, U.S.A.*
[2]*Undergraduate Studies, Clemson University, Clemson, SC 29634, U.S.A.*

Keywords: App Development, Creative Inquiry, iOS, Android, Smartphone, Tablet.

Abstract: Teaching mobile device software development is challenging. Almost everything about it is different from teaching a traditional software development class in which the target computer (on which the software developed by the students is to run) is a laptop or desktop computer. In a mobile device software development course, the target computer is a device (smartphone or tablet) that has a large number of features (Internet access, camera, GPS, gyroscope, media display, etc.) accessible by software. The material that must be covered in such as course is so broad that new approaches to delivering course content must be used. This paper describes the overall method by which we teach such a course. We describe four challenges and explain how we address each. We describe the structure of the course in detail, explain how a class policy of open collaboration and a university program called Creative Inquiry complement the proposed approach. We conclude with student evaluations and examples of apps we have produced over the past several years.

1 INTRODUCTION

Teaching app software development is challenging. Almost everything about it is different from a traditional software development class in which the target computer (on which the software developed by the students is to run) is a laptop or desktop computer. In a traditional object-oriented software development class, the focus is the programming language, how statements are formed, how classes and methods are constructed, and how object-oriented properties, such as inheritance, are brought to life in the program being developed. The focus is almost *never* the computer itself (laptop or desktop) on which the program will run. It is simply assumed that the program can read input files, take input from the computer keyboard or mouse, write output files and display output on the computer monitor.

In a mobile app software development class, the programming language is only the first of several major items that the student has to learn. The student must also learn how to use the interactive development environment (IDE) and the software development kit (SDK) specifically designed for the mobile device. This includes learning about different types of input and output touch elements (for example, buttons, sliders, and text boxes), working with different types of views (for example, table views, web views, views that allow for video), understanding new ways of providing input (such as gestures or speech), learning about new source streams of time sensitive or late-breaking information (such as Rich Site Summary, or RSS, and Twitter feeds) learning about maps, the global positioning system (GPS), accelerometers, gyroscopes, the camera both video and still, and on and on and on. Often apps must also work with app settings or an internal database to store persistent data. More advanced apps may also work with an external database, for example MySQL, that the app accesses through the web using web services written in *PHP: Hypertext Preprocessor* (PHP) or other scripting languages.

Teaching a one-semester course on mobile device software development is *significantly more challenging* than teaching a one-semester course on traditional computer science topics such as an introduction to programming, algorithms and data structures, compiler design and implementation, or operating systems. In courses such as the latter, the topic is well defined and narrow enough for the student to grasp and develop skills for in an evenly-

P. Pargas R. and J. Speziale B..
Revamping the Classroom - Teaching Mobile App Software Development Using Creative Inquiry.
DOI: 10.5220/0004847800710079
In *Proceedings of the 6th International Conference on Computer Supported Education* (CSEDU-2014), pages 71-79
ISBN: 978-989-758-022-2

paced and systematic manner. In a mobile device software development course, the material that must be covered is so broad that new approaches to delivering course content must be developed and employed.

This paper describes the method by which we teach a mobile device software development course. In Section 2, we explain in detail the challenges facing the instructor in this app development course and explain the approach we take in addressing these challenges. In Section 3, we talk about the structure of the course itself. In Section 4, we elaborate on the Creative Inquiry Program at our university and how it supports this course and helps generate ideas for app projects. In Section 5, we describe some of the apps that have emerged as the result of this effort. In Section 6, we summarize the results of student evaluations. In Section 7, we discuss future plans..

2 CHALLENGES / APPROACH

We find four major challenges facing an instructor of a mobile device app development course.

The first challenge arises due to the fact that the content is very new and also constantly changing. The Apple iOS™ and Google Android™ operating systems (OS), for example, are updated at least annually; keeping up with the major changes in the OS and the associated SDKs is difficult. Textbook authors, unfortunately, cannot possibly keep up; on the day that a textbook is published, the OS and the SDK it references are already one version behind the latest stable version. And so the first challenge is finding resources that can help the student learn about the latest version of the OS and SDK in a timely manner.

The second challenge is that the material that has to be covered is very *broad*. In iOS, this includes learning the syntax of a language, Objective-C, whose syntax is unlike any of the other more commonly taught languages such as Java, C++, C# or Python. A student wanting to take this app development course is unlikely to be familiar with Objective-C. (On the Android platform, this challenge is not as great since the language used for Android apps is Java which is known to many students.) On both platforms, however, new course material also includes touch input artefacts such as buttons, sliders, segmented controls, and date pickers as well as input finger movements such as gestures, swipes, and multi-finger touches. The material includes a variety of views such as web views, table views, and views that present video and animation, from and to which students will have to learn how to program transitions.

The course material must also include a variety of components and features of smartphones that are available for software control, components such as the still and video camera, the global positioning system (GPS), the accelerometer, a gyroscope, maps, and a compass. The material also includes new sources of streaming data such as RSS feeds and Twitter feeds. The instructor may also want to cover sending and receiving short message service (commonly called SMS or text) messages, or sending and receiving email. Yet another content topic that the instructor may want to include is graphics for animation, using OpenGL for example.

The third challenge is how to have a discussion of good database design and implementation. An app may require a local database that resides in persistent memory on the mobile device, or may communicate with an external database that resides on a server in the cloud and accessed via web services. In many instances, both an internal and an external database are required. Unfortunately, in many computer science programs, a database design course is an elective and not required. If such is the case at your university as it is at ours, then a minimum of one to two weeks of lecture will be required to introduce very basic database design concepts to allow for an internal and external database implementation in the app.

The fourth and possibly the most difficult challenge is how to make the course relevant, i.e., how to make the course as close to professional app development as possible. Our goal was to avoid having the students develop *toy* apps that have no purpose other than as an exercise in programming. How do we give our students an experience that is close to real world app development? How do we provide students with app ideas that have real-life application? The challenge was to find projects that would give our students this type of experience. We found our answer in *Creative Inquiry*, a university-wide program designed to give undergraduate students experience in designing and developing solutions to research problems posed by faculty members, researchers and staff.

These four challenges require us to *rethink* the manner by which we teach this course. The approach we have chosen (which we detail in Section 3) is a variation of the “flipped” or the “inverted” classroom (Walvoord and Anderson, 1998; DesLauriers, Schelew, and Wiemanm, 2011). This variation consists of a combination of (a) minimized lecture on the part of the instructor, (b) open

collaboration among the students, (c) frequent student presentation of programming assignments, and (d) close mentoring of the semester projects.

In order to minimize instructor lecture, we had to find resources with which students can work outside of class. Examples of these resources are listed in Section 3.2.1. Students are expected to view these video clips or tutorials *before* coming to class. Proponents of flipped classrooms suggest that allowing the students to gain what is referred to as *first-exposure learning* on their own before class and then working on applying their learning during class can in certain situations be more effective than traditional lecture. To be effective, an assessment tool, such as an assignment on the material the student learned, must be included. In our case, the assessment tools are the four assignments described in Section 3.2.3.

Where our approach differs from typical flipped classrooms is in "open collaboration". The purpose of open collaboration is to facilitate the exploration of as many facets of the SDK as possible. Students are encouraged to include different app components (compass, gyroscope, camera, GPS, etc.) in their assignments and to share their knowledge with the rest of the class. This is accomplished by requiring each student to maintain a personal website on which he or she posts URLs of sites that they have found helpful. More importantly, the student posts copies of his or her programming assignments (complete with source code) when they are submitted. Students are free to discuss and share their programs before and after submission. Moreover, after each assignment is evaluated, the instructor selects the two or more best programs and asks the students to give a brief (10-minute) presentation to the rest of the class, focusing on the new app components employed and describing the corresponding source code. Students in effect teach each other what they learned as they developed their assignment. This is further discussed in Sections 3.2.2 and 3.2.3.

The fourth challenge is making the course relevant. We are fortunate at our university to have what is called a Creative Inquiry (CI) program that encourages faculty to design one to three credit hour courses involving undergraduate students in the faculty member's research. So, for example, an English professor may be working with students in studying the writing of British author Virginia Woolf. A Biology professor may be interested in engaging students in the study of swamp forests. An Entomology professor may have his students researching whether firefly populations are decreasing. These and many other CI courses are opportunities for collaboration with our app development class. The instructor contacts the instructors of these CI courses and discusses the possibility of developing apps supporting these CI efforts. Sometimes a natural collaboration between our app class and the CI course emerges, sometimes it doesn't. When a potential collaboration does emerge, the app class instructor must facilitate the matching of one or two student app developers and the CI course. We devote Section 4 to the Creative Inquiry program that has provided support and a rich source of ideas for this course, helping us respond positively to the fourth challenge above.

3 APP COURSE STRUCTURE

In this section, we describe the organization of our app development course (henceforth referred to simply as *X81*, short for CPSC 481/681, the official number of the course). In *X81*, the semester is divided into two parts. The first half of the semester is spent on four programming assignments of increasing difficulty, with the requirements of each assignment being a superset of the previous. In the second half of the semester, the students work on a major project. The project starts with the students submitting a proposal for a project. After the project is approved, the remainder of the semester is spent on the design and implementation of the software system.

During exam week, students formally present their results in a class mini-conference. With the course instructor serving as session chair, students give 20-minute presentations and demonstrations of their apps, just as they might at a real conference. All are invited and project co-mentors, other students, faculty, staff, and occasionally family members do attend. At the end of exam week, students submit the entire project (source code and documentation) for a final private demonstration to the course instructor and for final evaluation.

Each of these course components is described in detail below, but we start with an explanation of the choice of mobile device platform.

3.1 IOS, Android or Other?

A recent Nielsen poll in the US (Nielsen, 2013) gives Android a 52% to 40% smartphone market share edge over iOS. Another poll (Business Wire, 2013) gives Android a 79% to 13% edge over iOS worldwide in the 2nd Quarter 2013, suggesting that

Android is surging and significantly surpassing iOS. The takeaway from this is that Android and iOS together dominate the smartphone market in 2013.

We are *not* concerned about whether Android beats iOS or vice versa. In *X81* we cover *both* platforms (iOS in the fall semester and Android in the spring). What is important to us is that we cover the *dominant* platforms in this course. We keep track of trends, however, and should another platform (say, Windows) become a significant player in the mobile device world, we will adapt this course accordingly.

3.2 First Half: Building Skills

3.2.1 Gathering and Updating Materials

The first challenge listed in Section 2 describes the difficulty of finding a textbook or other materials that teach the student about the latest version of the mobile device OS and SDK. For the course, we use online materials, including videos, tutorials, sample code, and other online resources.

For example, for iOS, in addition to the comprehensive developer reference site provided by Apple (Apple, 2013), we use the video series provided by Paul Hegarty and Stanford University available on iTunes (Hegarty, 2013). For Android, in addition to the comprehensive developer site provided by Google (Android, 2013), we use sites such as Nick Parlante's (Parlante, 2011) course outline that provides tutorials on topics such as intents, lifecycles, lists, GUIs, and databases.

In addition to these, throughout the semester, we constantly search the web for new and better video or text tutorials that provided instruction on the latest version of the OS and SDK. Students in the class are tasked with seeking and reporting new sites that they find useful in developing their apps. By doing so, the class collectively and continually keeps the set of course resources up-to-date and available to the entire class through the class website maintained by the instructor.

3.2.2 Individual Student Websites

Students are required to create and maintain individual websites specifically for this class. On each website, students post links to resources that they have found useful in developing their apps. They also post the source code of each programming assignment they submit. This is so that their code is available for download by *other* members of the class. This is part of the class policy of *open collaboration* that is explained further in the next section.

3.2.3 Assignments and Open Collaboration

In this class, student home and class activity consists of (a) viewing video and tutorials providing detailed instruction on app development, (b) submitting four apps satisfying four programming assignments, (c) when asked, giving brief (5-10 minute) impromptu presentations about his or her programming assignment pointing out where in the source specific actions in the app are coded, and (d) sharing ideas about interesting things he or she may have discovered along the way.

The assignments are of increasing difficulty. The first assignment asks the student to build an app with several input controls (for example, buttons, sliders, or text boxes) several output controls (for example, labels, alert boxes, or images such as check marks or smiley faces corresponding to calculated results) and some activity that occurs when the client touches an input control.

An important class policy is: whereas in other programming classes, students are often forbidden from working together and certainly never allowed to copy one another's code, in this class students are not only allowed to copy each other's work but are *encouraged* to do so. The one requirement is that if a student develops his or her app with the explicit or implicit help of another, the recipient must explicitly *acknowledge* the other student's help and give credit to the other both on the recipient's website and in the information page within the app. Giving credit for help received is *mandatory*.

The rationale behind this policy of *open collaboration* is that in a given assignment, different students will go in different directions and use different tools within the SDK. Discussion about their apps in and out of class is strongly encouraged. Copying code is allowed and encouraged. In this manner, students quickly learn multiple features of app development from their peers. The tacit expectation, of course, is that *every* student brings something new to *contribute* to the class collective repository of knowledge.

The second assignment includes all the requirements of the first app and must include multiple views (for example, table views, web views, or views that play audio and video). The third assignment must include a local database created and populated using SQLite3, a commonly used internal database for apps. The fourth assignment has all the requirements of the third assignment, plus

it must involve an external database, preference settings and two additional features such as GPS, the video or still camera, the accelerometer, the gyroscope, or must capture input from some external device such as an automotive on-board diagnostics (OBD2) sensor which can provide the vehicle owner or repairman or app developer with the status of various vehicle subsystems.

By the time Assignment 4 is submitted, the semester is half over.

3.3 Second Half: Major Project

The second half of the semester is spent on the proposal, design, and development of a major project.

3.3.1 Project Proposal

Students may submit an original proposal or may select a proposal from a list of ideas provided by the instructor. The one-page proposal is intended to be a brief overview of the project and is the starting point of a discussion between student(s) and instructor. The result of the discussion is a resubmission of the proposal with more detail and with a fairly complete storyboard sketch. There exists software available for free that facilitate storyboard development producing mock-up views that look very much like actual screenshots from a smartphone. An example is Fluid (Fluid, 2013) that provides a software tool for quickly and easily building mock-ups of apps.

The student has the option of proposing a project that he or she may be interested in. Students are encouraged to submit ideas that relate to and benefit the university and campus life. So, for example, a proposal to develop an app that expedites orders at a local pizza restaurant would not be approved. However, a proposal to (1) help newly arrived international exchange students get around campus using an interactive campus map, (2) complete all university requirements for international students, and (3) communicate with one another and with the international student office through SMS and phone numbers stored in internal and external databases, this type of app proposal would be approved. (A screenshot of this app is found in Section 5).

Most students, however, do not have a pet project in mind. A list of ideas offered by the instructor is the result of interaction with university faculty, researchers, staff and administrators interested in participating in the Creative Inquiry program. Throughout the first half of the semester, the instructor meets with various university personnel (potential co-mentors) and discusses the possibility of collaborating for the purpose of developing an app for the co-mentor and his or her students. The co-mentor and students provide the project content; the *X81* instructor and students provide the technical expertise. The list is the result of this consultation with several faculty and staff and is offered to the *X81* students as pre-approved projects.

3.3.2 Timeline, Database and Code Design

After the proposal is approved, the students develop a project timeline or schedule of tasks and deliverables. The instructor reviews this timeline, critiques it and returns the modified timeline to the students.

After the timeline is approved, database and code module design begins. The app may require both an internal and an external database. Final approval is given only when the instructor is convinced that all of the functionality proposed for the app can be supported by the structure of the internal and external database.

3.3.3 Presentations

Over the next three weeks, each project team gives three short (10-minute) presentations and one full (20-minute) final presentation before the class, reporting on the current status of their work. The instructor and class critique the presentation as well as the content of the project, offering suggestions for improvement or alternative approaches. In this manner, each team has multiple opportunities to refine their presentation skills as they develop the app solution to their problem.

The final presentation occurs during exam week. Co-mentors and students, interested faculty and staff, and family members attend. The instructor serves as session chair and introduces the student speakers, as they would be at a real conference. Every attempt is made to simulate and provide the students with a formal conference experience, including a soda and pizza reception at the end.

3.3.4 Final Demonstration and Submission

After the final presentations, student teams are required to meet with the instructor one last time to give a final, private demonstration of the software and to submit all source code and supporting documentation. The documentation includes a Technical Reference Manual and a User's Manual, the presentation slides, and any other supporting documents. Students have been instructed to develop

the User's Manual using language understandable by a non-technical user of the app. The Technical Reference Manual, on the other hand, is for a computer science student who may want to extend the app in the future.

4 APP IDEAS: CREATIVE INQUIRY

The students are always given the opportunity to propose projects for which they are interested in developing apps. The one requirement in this class is that the app be one that is designed to benefit students, teachers, staff, administrators, or other personnel in an academic environment. The app must help students learn or teachers teach or make life better for someone in the academic community.

Coming up with a list of suitable projects is not easy; they must satisfy several course goals. One course goal is for the project to take advantage of the features of the mobile device. Another course goal is for the project to be generic, i.e., ideally adaptable to any university or institution, not just our university. A third goal is for the project to be upwardly scalable, i.e., capable of being initialized with and holding additional content.

Once a list of potential projects has been identified, matching the right students with the right projects is crucial for success. It is for this reason that we have worked with a university program called Creative Inquiry to help us identify projects and involve other faculty, researchers, staff, and administrators as co-mentors of course projects.

Section 4.1 describes the general Creative Inquiry program. Section 4.2 explains how Creative Inquiry has helped provide app ideas and projects for *X81*.

4.1 What Is Creative Inquiry?

Clemson University's Creative Inquiry model for undergraduate research provides students with team-based, collaborative research experiences that address real-world problems and prepare students for jobs in the changing economy. Graduates have stated that they were better prepared for jobs or graduate school, and more attractive to employers. Students in Creative Inquiry have developed business plans for start-up companies and participated in patent applications, in addition to presenting and publishing their work.

The model develops teams of students to address topics identified by the faculty mentors, the students, or that are spurred by external influences. Each project is embedded within one or more academic courses. Projects may be multi-disciplinary. Teams work on a given project for two or more semesters. Some projects have a natural end point; others continue to evolve for many years. For example, a performing arts team dedicated two years to developing and producing an original play. Several English and Social Sciences teams have collaborated on books. An award-winning interdisciplinary team - with students from engineering, business, and the humanities - has worked for several years to boost economic development and the standard of living in a Haitian village. Their signature project was designing and installing a water system for the village.

Each academic year, approximately 3,500 Clemson undergraduates participate in Creative Inquiry projects. Departments, such as food science and geology, use Creative Inquiry to attract and retain Others departments, such as industrial engineering and bioengineering, use it to in concert with their senior design and freshman courses. Students in all departments praise their experiences as valuable in terms of the research accomplished and the opportunities to gain insights into potential careers.

Creative Inquiry is sustained by high levels of student and faculty interest, substantial funding from the university, and a demonstrated record of accomplishments within both the academic and business worlds. The program has grown to the point that private donors and industries are sponsoring student research teams. A key feature of the program is its flexibility. Projects from all disciplines are supported. Students are encouraged to develop ideas for projects and then identify faculty mentors to refine the ideas and mentor the work.

4.2 How Co-mentoring Supports *X81*

X81 is classified as a Creative Inquiry course. As such, it enjoys financial support from the university; this support is used to purchase mobile devices to be used in the class, or pay hourly wages to students who have taken the course before and who can serve as teaching assistants. More importantly, though, other members of the university recognize and support the Creative Inquiry model and are more willing to participate in co-mentoring a Creative Inquiry project.

A typical project involves one or two faculty members plus one or more of their students, *i.e.* the

X81 instructor plus one or two *X81* students. For example, one project last year involved an English professor and avid researcher in the works of British author Virginia Woolf. The English professor had eight students working with her in gathering and organizing material associated with Virginia Woolf. These included maps where Woolf lived and worked, pictures of British houses she lived in and London gardens she frequented, articles written about Woolf, essays critiquing her work, lists of websites where one can gather more Woolf information, and even copies of ledgers of daily household expenses that Woolf and her husband maintained.

Two *X81* students undertook the technical development of an app that organized the material gathered by the English professor and her students. The final version of the app presented the content on an Apple iPad in an efficient, easy-to-use, and intuitive manner. The English professor and the *X81* instructor served as co-mentors of the entire group with one guiding the software development process and the other guiding the organization of the content. The results of the effort were presented in an international conference on Virginia Woolf last spring (Sparks and Pargas, 2013).

5 RESULTS

The proof of a pie is in the eating. The proof of a course on mobile app software development is in the apps that are produced. Over the past two years, students have successfully produced over 30 iOS and Android apps.

Figure 1: Three sample apps produced by X81 students.

Three examples are shown in Figure 1. In the upper left, *iCUExchange* is designed to help international students prepare for their study at this university. The app was designed by three international exchange students and provides helpful information before and after the international students arrive on campus, including important deadlines, contact information, tips on getting around the campus, finding classrooms and getting to know other people. In the upper right, the *Virginia Woolf* app is designed for people who want to be able to access a mobile collection of photos, documents, and general information about the British author. The app serves as a knowledgeable portable tour guide. The bottom app, *R3-ROS Robot Remote*, provides an iOS user with an interface to control robots (represented by cubes) within a virtual world. The app uses the gyroscope and motion sensors of iOS devices to control the robots.

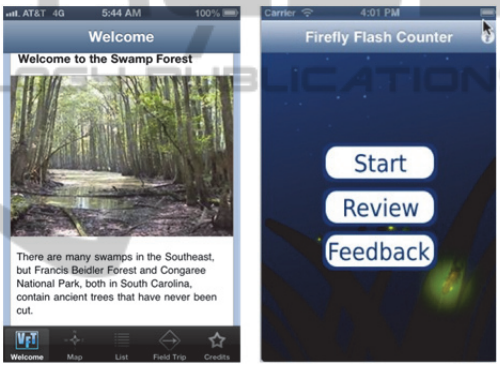

Figure 2: Two apps available at the Apple app store.

In some cases, development of apps start out in the classroom but continue after the semester is over and eventually are made available to the general public. Two examples are shown in Figure 2. *Swamp Forest*, shown on the left, provides a virtual field trip through the Francis Beidler Forest and Congaree National Park in South Carolina, USA. This and two sister apps, *Cove Forest* and *Salt Marsh*, are all available at the Apple app store. The Firefly Flash Counter, shown on the right, is also available at the Apple app store and is used by citizen scientists around the USA to count fireflies as part of the Vanishing Firefly Project 2013 (Baruch, 2013). In total, six iOS apps originating from this course are now available at the Apple app store and one Android app will be available on the Google play store in spring 2014.

Undoubtedly, instructors at many universities have developed app development courses with approaches similar to this course (e.g., Muppala,

2011). We feel, however, that this course is different from others with its policy of open collaboration and with the availability of the Creative Inquiry program to provide support and real project ideas. Open collaboration is a powerful and effective way to cover large numbers of disparate topics related to app development and Creative Inquiry is a rich source for real-life projects.

6 STUDENT EVALUATION

This three-credit course has been well received by students. Students may count it as one of two computer science electives and therefore the course must compete for student enrolment with over 15 other elective computer science courses. It consistently attracts between 15 and 18 students each semester, a very good and manageable size for a project course.

Multiple end-of-semester evaluations show that over 85% agree or strongly agree (ASA) that the course added significant value to their resumes and transcripts, over 90% ASA that the open collaboration policy helped them learn the material more easily, and among students who chose a Creative Inquiry project, over 85% ASA that working on an actual research project with a co-mentor and students from other disciplines was a valuable educational experience. Moreover, over 90% would recommend this course to other students and over 85% reported that they were satisfied with the learning that they received.

On the negative side, over 95% felt that the amount of work required by the course was significantly greater than other three credit hour computer sciences courses and over 85% spent on average over 10 hours per week on this one course.

A bit surprisingly, relatively few students (less than 10%) felt that the absence of a textbook hindered their learning of the material. Moreover, over 95% ASA that much of their learning came from discussions with their classmates and over 75% ASA that they were successful in finding answers to technical questions on blog sites.

7 CONCLUSIONS AND FUTURE WORK

Teaching mobile device software development indeed is challenging. The rapidly changing software tools, the amount of material that must be covered, the need of students for database design skills, and the constant search for real-world applications pose significant problems for the instructor of such a class. Traditional lecture by the instructor no longer suffices. New approaches such as open collaboration among students, frequent presentation of coding tips by students, and continuous communication between instructor and students and among students themselves, are necessary in order for student projects to be successful. And support by the university for undergraduate research projects in programs such as Creative Inquiry, both in terms of academic credit as well as monetary support, helps generate the all important research ideas that produce realistic and beneficial app projects.

ACKNOWLEDGEMENTS

The authors are grateful to the students who have taken and participated in this app development course, to faculty who have served as co-mentors, and for the continued support received from the Clemson University Creative Inquiry Program.

REFERENCES

Apple, Inc., https://developer.apple.com, 2013.

Baruch Public Service, *Vanishing Firefly Project*, http://www.clemson.edu/public/rec/baruch/firefly_project/, 2013.

Business Wire, *Apple Cedes Market Share*, http://www.businesswire.com/news/home/20130807005280/en/Apple-Cedes-Market-Share-Smartphone-Operating-System, August 6, 2013.

DesLauriers L, Schelew E, and Wieman C. *Improved Learning in a Large-enrollment Physics Class*. *Science* 332: 862-864, 2011.

Fluid, https://fluidui.com, 2013.

Google, Inc., https://developer.android.com, 2013.

Hegarty, Paul, *Coding Together: Developing Apps for iPhone and iPad (Winter 2013)*,
https://itunes.apple.com/us/course/coding-together-developing/id593208016.

J. K. Muppala, Teaching *Embedded Software Concepts Using Android*, Proc. 2011 Workshop on Embedded Systems Education (WESE 2011), EMSOFT 2011, Taipei, Taiwan, Oct. 13, 2011.

Nielsen, *Who's Winning the U.S. Smartphone Market*,http://www.nielsen.com/us/en/newswire/2013/whos-winning-the-u-s-smartphone-market-.html, August 6, 2013.

Parlante, Nick, *CS193a Android Programming* http://www.stanford.edu/class/cs193a/, 2011.

Sparks, Elisa and Pargas, Roy, *Hands on the iPad: Crafting a Visual Guide to Place in Woolf*, The 23rd Annual Conference, Virginia Woolf and the Common (Wealth) Reader, Vancouver, BC, Canada, June 6-9, *2013*.

Walvoord B. E., and Anderson V. J., *Effective grading: A tool for learning and assessment*. San Francisco: Jossey-Bass, 1998.