CmpSci 491IP, iOS Programming

Fall 2020, MW 9:05 - 10:20, Online

Chip Weems
CMPS 342, 545-3163
weems@cs.umass.edu

Office Hours: M 10:30 - 11:45
and by appointment

Course web page: https://people.cs.umass.edu/~weems/homepage/index.html

Book: Beginning iPhone Development with Swift 5, Wallace Wang, Apress. Optional Reference: Pro iPhone Development with Swift 5, Wallace Wang, Apress. Both available as e-books.
Apple Human Interface Guide:
https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/

Seminar Motivation

In this seminar we explore how to develop apps for Apple's iOS. Mobile platforms present interesting challenges such as limited power, memory, and screen space. They also offer some features such as touch-screen, camera, accelerometer, location awareness, camera, sound, etc., and of course extreme portability, that make possible some innovative applications.

Far more important than the technology, however, is the approach taken to designing an app. Successful apps serve the needs of their users. The app store is full of little-used apps that failed to do so. So before we start coding, there are some key questions to answer: Who is the user? What do they need? How, where, and when will they use the app? What will it provide that isn't already available? If it relies on a data source, how will that be maintained so that the app remains useful? Once we have a good picture of the user and their needs, we can write a product definition statement that guides the design of the app.

Apps also fail because they are unreliable. To avoid that, it is important to design an architecture for the app, and a development plan that builds and tests the app, piece by piece, with a clear schedule of milestones in the development process. Which components are at the core of the app, and need to be brought to a stable state earliest, so that other components can go forward? What are the dependent relationships between the components, so that their creation can be ordered most effectively?

Another source of failure is cumbersome interface design. How does the user model the process that the app provides? App developers can be prone to creating interfaces that fit their own models, or that are easiest to map into the APIs of the environment. How can that be avoided?

Lastly, apps often fail for lack of a business model. Does the app have a sufficient user base? Is the app feasible, given the resources of the development team? How will the app and developers respond to wild success? What is the plan for future releases to sustain interest in the app?

As you can see, the goal is to learn how to successfully build an app that is actually useful, which involves much more than just writing Swift code for iOS.

Learning Process

To learn how to answer these questions, we will go through the process of developing an app of your own choice, as if you are forming a startup company. You can work individually or as a small team (2 or 3), noting that teams will be expected to develop more extensive apps than individuals. There will be homework to prepare for designing the app and planning your startup, and design reviews, using the class as a focus group to provide feedback. Once development begins, there will be a set of demo days, where I will play the roll of your venture capital source, and you will need to show that you're on schedule with your milestone plan.

This is not a traditional course, where the professor lectures to you. Beyond the project work, you will also be preparing presentations in which you (working in a small team), guide the rest of the class through developing an app that uses some part of the iOS APIs. In the first half of the semester, you will do this for one of the basic APIs covered in the book, extending their sample app to something more interesting, or possibly combining it with other of the APIs.

In the second half of the semester, you will choose from among the vast range of possibilities in the APIs to introduce the class to something beyond the basics - perhaps something that one or more members of your presentation team will be using in their apps. There is no better way to really learn how something works than by having to teach it to others.

That means the course content isn't predefined - it is designed to adapt to your interests. You should view it as an opportunity to explore a wide range of topics under the broad umbrella of iOS applications.

Initially, I'll do a few lectures to lay some groundwork and provide time for you to prepare your first presentations. At least a week before your presentation, you should schedule a meeting with me to plan what you will do. You have an hour or more, and I expect you to use it all. Your presentations can begin with walking the class through examples from the book(s), but part of the grading will be in how much you take us beyond that.

Required Technology

iOS development requires the use of Apple's Xcode environment. Xcode only runs under MacOS. MacOS generally needs to run on Apple Macintosh hardware. In prior years, some students have been successful in getting Xcode to run on Hackintosh substitutes (PCs with a custom build of MacOS), but some have also thought they were successful, only to discover too late that some feature they need is inoperable. Normally, those students have been able to finish their project by using the Macs in the U-Space, but because of the COVID restrictions, that won't be an option this year. The safest strategy is thus to have an actual Mac.

We will be using iOS 13, even though iOS 14 is due out soon. Many of the concepts will carry over. Starting with beta software or making a transition in mid-semester is a recipe for problems. You should turn off auto-update on the machine you'll use for development, or you can wake up to a nasty surprise some morning and find that nothing is working.

You should be able to access to a machine that can run XCode and the iPhone simulator during class, because we'll be actively going through examples, testing out ideas, sharing what we discover, and sometimes helping each other with solving problems.

This semester, classes will be conducted via Zoom and using Moodle, and because they are designed to enable us to interact directly, they will be synchronous. So you will also need internet access that enables the use of those technologies.

Class Recordings: I have listed the course as not being recorded because some parts of it will not be. Your design and project presentations, and the feedback from classmates will not be recorded to reduce the pressure on the presenters and allow comments to be be made more freely. My introductory lectures will be recorded, and with the permission of groups presenting other APIs, their presentations will also be recorded. Recordings will be available through Moodle.

Grading: Presentations: 20% Homework: 40% Project: 40% No exams

Grade Scale:
A: 93%, A-: 89%, B+: 85%, B: 80%, B-: 77%, C+: 74%, C: 70%, C-: 67%, D+ 63%, D: 60%

Presentations: Two group presentations of APIs, one in the first half of the semester on a basic API from the book, one in the second half, on a more advanced API. 10% each. Graded on thoroughness of coverage, quality of preparation/presentation, engagement of class participation, coverage of allocated time. Presentation slides will be submitted as a pdf.

Homework: Nine assignments overall. Two assignments will involve getting used to Xcode, Swift, and basic APIs, while keeping a journal of issues encountered and how they were resolved. These will be demonstrated in class. The journals will be submitted. Six will be preparation for the project. The ninth assignment is a final version of those six, combined into a complete business plan, with changes incorporated in response to feedback. Each homework is 4%, except that the last one will be 8%. They will all be submitted as pdfs on Moodle.

Project: There will be five components of the project. A presentation of your plan (6%), a first milestone demo (7%), a second milestone demo (7%), a final demo of the complete project (10%), and an overall assessment of the project (based on complexity, use of iOS features, achievement of goals). For the plan and demos you will not need to submit anything, although if you use slides, you are welcome to submit those. For the final project assessment, there will be a brief written report of what you learned, problems you overcame, and who did what.

Late Policy: The API and project presentations, and the demos, are scheduled for particular days, and subsequent days are given over to presentations by others. Because most the six project preparation assignments and the project demos combine into a progression, falling behind inevitably snowballs into a trouble finishing the project. Because the project preparation assignments are drafts that will be revised, it is better to submit what you have ready for those (there will be a 30% late penalty, for those, which is probably more than I will deduct for deficiencies – I'm generous with partial credit when I have something to actually grade.

Project Teams:

When you are in a team, then your grade is partially dependent on your performance as a team. Some ways that you can help to ensure that your team does well:

Be realistic with your partner in discussing your abilities and time commitments when you divide up the project work. If possible, get together and work collaboratively, taking time to brainstorm ideas for how you will go beyond the requirements. Paired programming is a recognized software engineering technique in industry, and is known to yield higher productivity, especially on small, intensive assignments. In many cases, you'll learn more and learn it more quickly through working together directly.

Make sure you have each other's schedules, phone numbers, e-mail addresses, etc. Establish a clear policy of when it is OK to contact one another. Let each other know when you are going to be unavailable with enough advance warning for good contingency planning.

Communicate! Communicate! Communicate! Tell each other what you are thinking. Don't keep thoughts inside. Remember to praise each other for jobs well done. If you feel the need to criticize, use statements that start with "I think…." or "I feel…" Avoid criticism that starts with an accusatory "You did…," or "You always…" Be very clear about who is doing what - in the final project summary, I will ask you to jointly summarize what each of you contributed.

Think of yourselves as a team. Develop some team spirit. Name your team. Develop a unique style for your presentations and demos (e.g., team colors, a logo). Get to know each other as individuals. Discuss hobbies, career goals. If you each take the time to care about one another, then you'll do what it takes to excel as a team.

University Accommodation Statement

The University of Massachusetts Amherst is committed to making reasonable, effective and appropriate accommodations to meet the needs of students with disabilities and help create a barrier-free campus. If you have a disability and require accommodations, please register with Disability Services to have an accommodation letter sent to your faculty. Information on services and materials for registering are also available on the University of Massachusetts Amherst Disability Services page

Course Inclusiveness Statement

No matter who you are or how you define yourself you are welcome in this class. Each person here is a human being deserving of dignity and respect. My goal is to help you learn the subject matter in a way that you will find useful, and to help you have an enjoyable and empowering experience in doing so. It is important to keep in mind that we are all coming to this class with different backgrounds. For many, this is a first app development course, while others have some prior experience. We are all here to learn together! There are no dumb questions! From time to time, I may enlist some students to help others in class. If I ask you to help, remember that we all have different modes of learning, and there is no stigma to be associated with needing assistance. Please reach out to me if you have any concerns.

University Academic Honesty Statement
Since the integrity of the academic enterprise of any institution of higher education requires honesty in scholarship and research, academic honesty is required of all students at the University of Massachusetts Amherst. Academic dishonesty is prohibited in all programs of the University. Academic dishonesty includes but is not limited to: cheating, fabrication, plagiarism, and facilitating dishonesty. Appropriate sanctions may be imposed on any student who has committed an act of academic dishonesty. Instructors should take reasonable steps to address academic misconduct. Any person who has reason to believe that a student has committed academic dishonesty should bring such information to the attention of the appropriate course instructor as soon as possible. Instances of academic dishonesty not related to a specific course should be brought to the attention of the appropriate department Head or Chair. Since students are expected to be familiar with this policy and the commonly accepted standards of academic integrity, ignorance of such standards is not normally sufficient evidence of lack of intent (http://www.umass.edu/dean_students/codeofconduct/acadhonesty/).