SER423 Mobile Systems

Syllabus Spring 2017

1. **Instructor**

Tim Lindquist (Tim.Lindquist@asu.edu)
hours: MW 1-3pm
Peralta 230D
office: 480 727 2783;
homepage: http://quay.poly.asu.edu/Mobile
zoom: https://zoom.us/j/5931499831

2. **Catalog Description**

Mobile applications, their architecture, design, and supporting technologies; Mobile device operating systems and frameworks; synchronizing mobile applications, support for inter-application data-exchange; accessing and manipulating special-purpose device capability such as location, orientation, and input modality. Programming practices for securing mobile applications, and optimizing run-time performance. (3 credits)

3. **Prerequisites**

SER 421 (or CSE 445).

a) Web application development,
b) Developing distributed applications in a high-level language, such as Java, with events, serialization, sockets, and threads.
c) Experience developing applications having window-based user-interfaces (GUI); such as, FLTK, Qt for C++, and AWT or Swing for Java.
d) Experience learning new programming languages -- largely on your own. Swift is likely a new language for most students in the class, and will be required for programming exercises and/or on exams. Very little class time is devoted to teaching the Swift.

4. **Course Overview**

This course is the final of the Distributed Web/Mobile focus area for the BS Software Engineering. Various frameworks and platforms are used in the course, but in particular: iOS, and Android. The course is all about developing native mobile apps – Java for Android and Objective-C/Swift for iOS. The course provides several individual mobile app development labs (roughly 5-7 on each platform) that explore using various aspects of the platform. These may or may not be turned in for grades. The course uses a project-based methodology in which we compare the two common platforms and app development languages.

Students are expected to know and use several concepts/technologies: **Serialization**: persisting objects in various formats. **TCP/IP sockets**: You are expected to develop Java and/or Swift programs that make socket connections between a server and clients, and which communicates information via that connection. **Threads**: You will be required to understand the motivation and implementation of multi-threaded applications. **Databases**: Mobile apps rely heavily on app-specific databases – iOS's Core Data, and Sqlite (JDBC) programming for Android. Students are expected to develop apps that access databases on both platforms. Swift and Java: In Swift, we use code blocks (closures) and Optionals with minimal instruction.

The course may be video-cast to multiple locations, and may have an Online section through ASU Online. Students registered to take the course through ASU Online are expected to know and follow the Online course offering policies.

5. **Course Resources** – A computer capable of running the software listed below is (highly) recommended to complete the course. ASU facilities can be used if you can come to any of the ASU campuses. Virtual access is not currently provided to Apple OSX equipment.

a. All students are expected to have access to an Apple MacOS (Sierra) or MacOS X (El Capitan). You will need to have the following software installed on your system:

   a. Xcode. The most recent version at the time the course semester begins. (probably 8.0), available through the Apple App Store for no charge. As of recently, you can develop apps and test them on your personal device without paying to become an Apple developer. Nevertheless, having the iOS Simulator and the Android Emulator are sufficient for completing the course. Xcode does not run on anything other than a MacOSX devices. Course exercises develop Xcode iOS projects (that run with the iOS Simulator) in Swift for this course. You should also install the optional documentation and Xcode command line tools (from terminal: **xcode-select –install**).
   b. Android Studio. Version 2.2. With Android API version 23 (Marshmallow), and 4.0.3 (IceCreamSandwich - 15), Android SDK build and platform tools version 25. The latest of the Gradle plugin.
      See: https://developer.android.com/studio/install.html
      download the installation file, launch it and follow the directions which will guide you through the installation and download of necessary additional downloads.
   c. Android Emulator.
   d. Ant (latest version – 1.9 or later)
   e. Java Development Kit (standard edition version 1.8 or later)

b. The Swift Programming Language (Swift 3) (https://swift.org)
https://itunes.apple.com/us/book/swift-programming-language/id881256329?mt=11

c. The Busy Coder's Guide to Android Development by Mark L Murphy. Not-so recent versions are available free in pdf form through CommonsWare. Using version 4.7 or later should be fine. The instructor is using version 7.4, which at "this time" is the most recent available for free. For the most recent version, see Warescription at: https://commonsware.com/Android/

d. The iPhone Developer's Cookbook: Building Applications with the iPhone SDK, by Erica Sadun; Pearson Education; ISBN 978-0321555458.

e. Professional Android Application Development by Reto Meier, Wrox Programmer to Programmer; ISBN 978-0470344712.

6. **Student Learning Outcomes**

a. Students are able to: Select an appropriate mobile-device operating system and configuration to meet the requirements of a mobile application, knowing the tradeoffs involved such as performance, security, and available framework support.

b. Students have knowledge of programming practices that are specific to developing efficient applications for mobile devices, and knowledge of host and target development tools including emulators and simulators, as well as device runtime environments.

c. Students have knowledge of and experience in developing applications with extensive user-interfaces in the context of limited display and various user input capabilities.

d. Students are able to: Analyze, design and realize applications that synchronize mobile databases with information stores on other devices.

e. Students are able to: Develop applications that communicate and synchronize among mobile and non-mobile devices, such as game applications.

7. **Topics**

a. Platforms: configurations, runtime environments and limitations. Mobile App build process.

b. Development environments; tools and implementation characteristics;

c. Development Paradigms; model, view, controller (MVC), data sources, outlets, and callbacks.

d. User-interface; multi-view, multi-touch interfaces and associated tool-kits; animation and multi-media.

e. Facilities for communicating with other on-platform applications;

f. Facilities for communicating with off-platform services; through the Web and other direct inter-connection frameworks such as sockets or RPC.

g. Mobile device language runtime environments; benchmarking their performance, and understanding their configuration. Build steps and the artifacts produces and consumed at each step.

8. **Assignments**:

The course includes several programming-based projects that explore aspects of distributed, web and mobile applications. At the discretion of the instructor, these may be submitted for a grade. Laboratory projects require that students use different languages and development frameworks. Each project normally spans about 1 week of the 15-week semester, and the course includes roughly 9 individual lab development projects. Students are required to provide oral and written communications describing their application development. App development exercises are augmented by quizzes (worksheets), and exams. Exams are primarily short answer, and include coding examples as well as app development concepts.

9. **Grading**:

1. Laboratory projects and quizzes/worksheets are graded to account for 40% of the course grade.
2. A midterm exam accounts for 30% of the course grade, and
3. A final exam accounts for 30% of the course grade.

10. **Academic Integrity Policy**

Discussions among students on class material and laboratories are encouraged, but all students in this course must follow the AIP and must turn in their own work. That means that every source-code class making up a laboratory app that you submit in this class must have the your copyright, and it must have been developed by you. In addition to claiming ownership of the code you deliver in this class, you must grant the Instructor and ASU the right to build, evaluate and demonstrate your code. Any student found in violation of this policy and/or ASU's AIP may be given a failing grade for the course. Example ways you can violate ASU's AIP is to represent someone else's work as your own (whether you bought it or not) or if someone else represents your work as their own. Its fine to discuss concepts with others, and to help others locate and correct errors in their work. But, all graded work (exams, programming assignments, as well as any written exercises or quizzes) in this class must represent your own individual work. Grading may include executing student solutions using software that compares the structure and content of other student submissions, or other available solutions. Any cases of suspected violation of ASU's AIP will be referred directly to the college office according to established policy. By your registration in this class, you are assumed to have read, understood and agreed to this policy.

ASU's AIP: https://provost.asu.edu/academic-integrity

One ramification of this policy is that every student must assure that neither an electronic nor hard copy of their work is accessible to another student. If you share a computing system with someone else, you must know how to use access control to protect your files. If someone else steals your work (with or without your knowledge,) you may both get failing grade for the course.