University of Utah
School of Computing

CS 4530-001 **Syllabus** **August 21st, 2017**

Course Staff and Schedule

Instructor	Dave Heyborne
Lecture Hours	Monday and Wednesday 3:00 PM - 4:20 PM (In M LIB 1120)
Office Hours	Monday and Wednesday 6:00 PM - 8:00 PM (Near M LIB 1120)
Teaching Assistant	Tejas Hirekatur Sreedhar
Help Lab Hours	TBD
Course Website	https://utah.instructure.com/courses/460186

Course Objectives

This course is designed to introduce students to the fundamentals of mobile application development. This includes topics such as how to create objects and data structures in code, how to layout views on devices, how to communicate with servers and services, how to use device sensors, and many other aspects of creating a modern mobile application.

While the focus of the course is not on Android specifically, this course will use the Android operating system and the Kotlin programming language to illustrate the various topics that are covered during lectures. Programming homework assignments will be used to solidify knowledge of the material. A small number of written assignments will be used where appropriate to ensure knowledge of background material. A large final project, chosen from a standard set of projects or an approved project of your own design, will be the culminating focus of the course.

Prerequisites

Programming Experience: You should be very familiar with object-oriented programming in a language like Java, Objective-C, Swift, Kotlin, C++, C#, or other similar languages. You should also be very comfortable making software that involves GUI programming, interacting with the filesystem, and creating hierarchies of well-organized code. If you have not yet successfully created multiple pieces of software that are more complicated than simple command-line applications, you should not take this course.

Problem Solving: This course is designed to illustrate concepts and then have students implement examples of those concepts in a somewhat unstructured manner. While specific requirements will be given for all programming assignments, the implementation details will largely be left to the programmer. You must be able to figure out how to use the tools you are given to create solutions to problems without a lot of guidance. The teaching assistant and instructor are here to help if you get stuck, but will not have time to guide you step-by-step through the solutions to homework problems.

Professionalism: This course expects students to act like professionals. There will be a small number of assignments from which a student's grade is determined, and a limited number of late days allowed on submissions (see Grading, below). You will be expected to turn in work on time or within the provided allowance of late days, and you will be expected to have tested and verified that your programming submissions compile and function as intended before handing them in. Aside from documented emergencies, no exceptions will be made to deadlines. Use source control so that you do not lose your work, manage your time effectively, document your code consistently, and turn work in on time - otherwise, this course will be extremely difficult.

Course Materials

This course does not use a textbook. Instead, students will be expected to read selected pieces of documentation and discussion which will be freely available online and assigned shortly after each lecture. There will also be a lecture video for every lecture presented in class which will cover most of what is discussed at the in-person lecture. Students should not treat the assigned readings or lecture videos as a substitute for coming to class - the in-person lectures are your opportunity to get clarification about course materials and take notes.

Grading

A student's overall grade will be determined from the following components:

Written Assignments	5%
Programming Assignments	60%
Final Project	35%

Individual assignments will be graded as follows:

Core Requirements	75%
Documentation / Code Structure	15%
Visual Appeal / GUI Layout	10%

Grades will be assigned as follows at the end of the semester:

A	94-100%
A-	90-93%
B+	87-89%
B	84-86%
B-	80-83%
C+	77-79%
C	74-76%
C-	70-73%
D+	67-69%
D	64-66%
D-	61-63%
E	0-60%

Percentages will be rounded to the nearest percentage point when assigning grades (0.5% rounds up). There will be no 'curving' or weighting of grades, though adjustments to an individual assignment's value can be made on a course-wide basis at the instructor's discretion.

Late Policy

Students will be allotted 5 late days which can be used penalty-free throughout the semester. All or none of these late days may be used on a given assignment, or spread throughout multiple assignments. A late day may not be subdivided into smaller increments of time - as soon as an assignment is one second late, it will incur a penalty of one late day. As soon as exactly 24 hours pass from that point in time, a second late day penalty will be incurred, and so on. Any assignment received after a student has exhausted their allotment of late days will not be graded and will be assigned a score of 0%. This includes the final project. If you are worried about missing the deadline for an assignment, start working early and hand the assignment in many hours before the actual due date. At the end of the semester, students with late days remaining may convert those late days into extra credit at the rate of 0.5% of extra credit per unused late day.

Academic Honesty

Cooperation among students to better understand course material is highly encouraged, as it is an effective learning tool and essential to real-world development team success. High-level discussion of programming techniques and problem solutions is the best way to help or be helped by your fellow students. The course Canvas page will have a discussion forum available and students are encouraged to use it to work through high-level discussions of problems with other students and the course staff.

Cheating in the context of this course is generally defined as (but not limited to) sharing and copying of code from other students or the Internet. Any code making up your solution should be written and understood by you. Small quantities of template code, such as that provided by Google in Android coding examples, is acceptable. You should still be able to fully explain the function of any and all template code you use. Refer to, but do not copy code from, Android SDK / Kotlin examples.

The University of Utah is extremely strict in it's cheating policies. We will be cross checking your code submissions. We do not distinguish between cheaters who copy others' work and cheaters who allow their work to be copied. Any student caught cheating will automatically be given an E in the course and reported to the University Student Behavior Committee.

Students With Disabilities

The University of Utah seeks to provide equal access to its programs, services, and activities for people with disabilities. Accommodations will gladly be provided for the known disabilities of students in the class. If you will need accommodations during this course, reasonable prior notice needs to be given to the Center for Disability & Access, 162 Olpin Union Building, 801-581-5020 (Voice and TDD). CDS will work with you and the instructor to make arrangements for accommodations.

All written information in this course can be made available in alternative format with prior notification to the Center for Disability & Access.