

## Syllabus & Overview

### *CS 442: Mobile App Development*

#### **Agenda**

1. Staff
2. Course overview
3. Administrivia
4. Resources
5. Introductions

#### **Staff**

**Prof: Michael Lee**

**[lee@iit.edu](mailto:lee@iit.edu)**

***Hours: Tue/Thu 10-12 (Zoom)***

**TA: Ashik Shaffiullah**

**[asyedshaffiullah@hawk.iit.edu](mailto:asyedshaffiullah@hawk.iit.edu)**

***Hours: TBA***

TA will provide implementation-level support, I will help with concepts/"big ideas" presented in class, to clarify assignment requirements, etc.

#### **Course overview**

***What will this class cover?***

#### **Software architecture / Design patterns**

"Wheels" of software architecture and implementation. Don't (fully) re-invent them!

Can we name any of these?

- Iterator
- Decorator
- Factory
- Singleton
- Observer
- Model-View-Controller (MVC) / MVX

Many patterns presented in the well known software engineering book "Design Patterns", written by the "Gang of Four": Gamma, Helm, Johnson, Vlissides.

### **Existing SDKs & APIs (aka Libraries)**

- NEITHER iOS NOR Android
- we'll be using a cross-platform framework called Flutter (more on this later!)
- tons of pre-existing libraries { widgets, animation, networking, etc. }

### **Tools & Workflows**

- editors / IDEs
- version control
- debuggers
- simulators

### **Common techniques / "Best practices"**

- responsive design
- refactoring
- composition over inheritance
- RESTful APIs

### **... for the development of (mobile?) software applications**

This is our carrot (what's the stick?). The truth is it doesn't really matter what the carrot is --- the journey is what's important in this class.

What makes mobile devices unique? More on this later.

### **(Partial) Topics**

- Application lifecycle
- Reactive programming
- Data management
- Persistence
- Backend integration
- UI design
- Responsive design

### **Class Outcomes**

***i.e., after completing this class ...***

### **Describe & Use common design patterns**

e.g., for separating and managing user interface elements, interactivity, dynamic data, persistence, and concurrency, among other software application concerns

### **Articulate tradeoffs in software architecture design**

For us, this will manifest in the design and architecture of an application development framework, e.g., Flutter.

Why is it designed/implemented the way it is? How is an improvement on what came before? What can be further improved/refined?

### **Proficient in the use of tools & workflows**

Aforementioned IDEs, simulators, etc. If you're already familiar with them, you should be even more practiced!

### **Built multiple mobile applications from scratch**

Making use of a substantial portion of our chosen framework's pre-existing libraries.

### **Administrivia**

*{ Prereqs, Grades, Assessments, Policies, etc. }*

### **Prerequisites**

*Undergrad: { CS 331, CS 351 }*

*Grad: { CS 401, CS 402 }*

This translates into:

- substantial programming experience in and comfort with statically-typed, procedural, object-oriented languages (though not all necessarily the same language) --- functional exp is a plus
- knowledge of the use and runtime complexity of common data structures
- understanding of system-level behavior and its influence on high-level code execution --- esp. having to do with optimization and efficiency
- fundamental understanding of system-level I/O (e.g., files, streams, console-driven user interaction)
- familiarity with callback APIs / asynchronous programming, and basic concurrent programming pitfalls (e.g., race conditions)
- familiarity with common programming tools: IDEs, debuggers, test suites

### **Non-Prerequisites**

*(e.g., mobile app dev experience)*

What you don't need to know!

- mobile app development (show-offs aren't welcome, though I'll give you a chance to tell me what you know later)

- iOS / Android / other mobile app platform
- any sort of GUI development
- web development (though it doesn't hurt)
- e.g., HTML/CSS, JavaScript, Node, React
- advanced asynchronous / event-driven programming

## **Grade Breakdown**

**Assignments** → 70%

**Quizzes** → 30%

( A ≥ 90%; B ≥ 80%; C ≥ 70%;

D ≥ 60%; E < 60% )

## **Assignments**

### **Working software applications**

Each submission will be a working mobile application with a functional GUI.

### **High-level spec & functional reqs**

Generally, we provide you with mockups (e.g., rough sketches) and functional requirements (e.g., user stories, unit tests, etc.), and you flesh out the rest and build the final application.

### **Using (some) existing components**

We will also typically specify what pre-existing components (e.g., widgets, classes, remote APIs) you should / should not use. Towards the end of the semester, most of these limitations will go away!

## **Late policy**

### **7-day late pool**

Each student starts the semester with a 7-day late pool, which can be distributed however they please (one day at a time) across assignments.

E.g., a student may choose to submit the first assignment 1 day late and the second assignment 2 days late, at which point they still have 4 late days to apply to later assignment(s). Once a student is out of late days, late assignments will not be accepted for credit. Late days may not be used after the last day of classes.

When an assignment deadline approaches, please let the TA know if you plan on using late days; when you submit your work late, please let the TA know then as well.

## Quizzes

### ***Objective / Concept-focused***

#### ***Unlimited # of attempts before due date***

Quizzes consisting mostly of objective questions (multiple choice, T/F, etc.) will be periodically administered via Blackboard. The function of these quizzes is to help you assess your understanding of conceptual material covered in lecture and assigned readings.

You may take each quiz an unlimited number of times before its associated due date --- the highest score earned on each quiz will be counted.

## Honor Code

### ***(About Copilot/ChatGPT ...)***

Every assignment should be an individual exercise!

You're paying good money and investing valuable time in this class. I fully recognize that you can complete the vast majority of work in this class by "cheating" with some combination of the following:

- Pasting big chunks of code from Stack Overflow / Reddit / Open source projects / your friends
- Using AI coding assistants such as Copilot / ChatGPT to blindly generate most of your code

But this will rob you of hours needed to actually master the material!

What do you want out of this class? (More on this later)

If you just want to pad your GPA, then you may be able to get away with the above, but there are easier classes you can take --- please make space for other students!

If you really want to learn the concepts and get to

Here's the thing: AI coding assistants are great, but *you need to understand the code being generated!* Sometimes you just really need to *type the code* to get it in your muscle (and brain) memory. But there's also no need to memorize all the APIs ...

When I do in-class demos, I will have my AI code assistant turned on, and will try to model "good" coding behavior.

All that said, if we identify what we believe is plagiarism or rampant AI use, we reserve the right to sit you down and ask for a code review. If you can explain everything your code is doing to our satisfaction (and it is not plagiarized), you're good!

## Disability Resources

### [disabilities@iit.edu](mailto:disabilities@iit.edu)

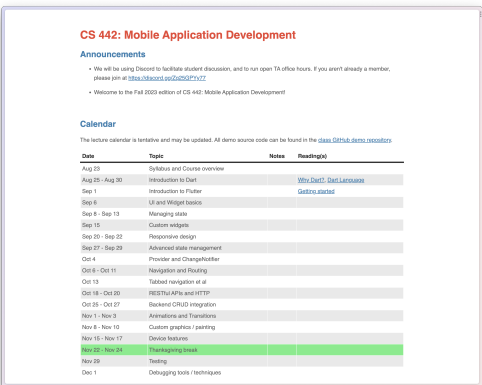
If you have a documented disability that we can accommodate, please contact the Center for Disability Resources (CDR) so they can send me an accommodations letter.

## Resources

### Class website

<https://moss.cs.iit.edu/cs442>

### *Readings before lecture!*



**CS 442: Mobile Application Development**

**Announcements**

- We will be using Discord to facilitate student discussion, and to run open TA office hours. If you aren't already a member, please join at <https://discord.gg/2u25G7m777>
- Welcome to the Fall 2023 edition of CS 442: Mobile Application Development!

**Calendar**

The lecture calendar is tentative and may be updated. All demo source code can be found in the [class GitHub demo repository](#)

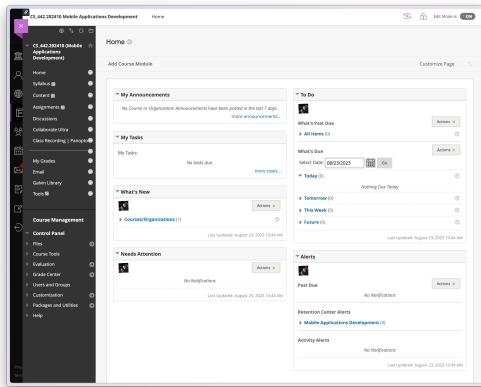
Date	Topic	Notes	Readings
Aug 29	Syllabus and Course overview		
Aug 29 - Aug 30	Introduction to Dart	<a href="#">Why Dart?</a> <a href="#">Dart Language</a>	
Sep 1	Introduction to Flutter	<a href="#">Getting started</a>	
Sep 5	UI and Widget basics		
Sep 8 - Sep 13	Managing state		
Sep 15	Custom widgets		
Sep 20 - Sep 22	Responsive design		
Sep 27 - Sep 29	Advanced state management		
Oct 4	Provider and ChangeNotifier		
Oct 6 - Oct 11	Navigation and Routing		
Oct 13	Tabbed navigation et al		
Oct 16 - Oct 20	RESTful APIs and HTTP		
Oct 23 - Oct 27	Backend CRUD integration		
Nov 1 - Nov 3	Animations and Transitions		
Nov 6 - Nov 10	Custom graphics: painting		
Nov 13 - Nov 17	Device features		
Nov 20 - Nov 24	Thanksgiving break		
Nov 26	Testing		
Dec 1	Debugging tools / techniques		

## Blackboard

### Quizzes

### Grades

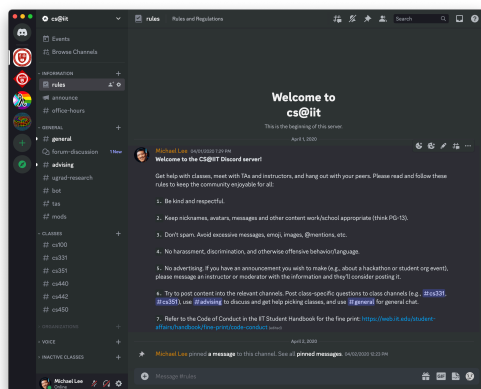
### Lecture recordings



**Discord**

**Peer Q&A**

**TA support**

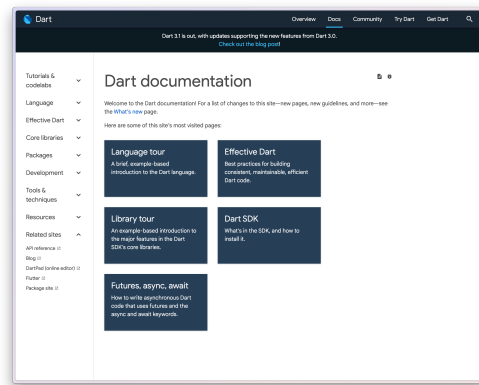
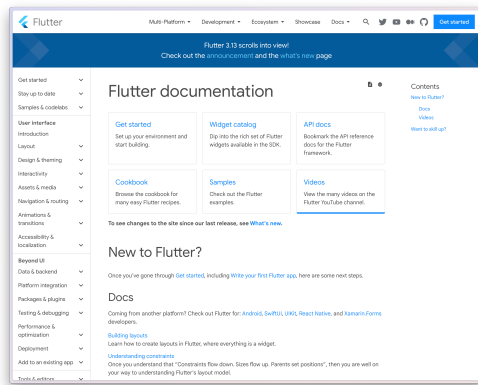


**Flutter & Dart Documentation**

<https://flutter.dev>

<https://dart.dev>

**(Great documentation & plenty of examples!)**



## Introductions

### About me

#### ***Only what's relevant!***

Started developing for iPhone/iOS with first SDK (2008) and worked on enterprise (i.e., in-house) apps for day traders in derivative markets (options, futures, etc.).

Taught the first mobile app dev class at IIT in 2010, and subsequently taught combined iOS and Android versions for a few more years.

Too much work -- effectively a full redesign every year!

Christopher Hield has been teaching an Android version for the past 5 years or so (thanks, Christopher!), but he left for UIC (good luck, Christopher!).

So I had to return to the whiteboard and decide how to teach the class. Most of the concepts that I used to teach are still relevant, but examples are very stale.

iOS vs. Android? Why not both? Flutter = cross-platform development toolkit & run-time environment by Google.

Note: I am *not* an expert (though my prior experience should help), but we will learn and master this together!

### About you

#### ***Why are you taking the class?***

#### ***What do you hope to get out of it?***

#### ***What prior experience do you have?***

Help me understand what our collective experience looks like so we can help each other (and you can help me!)



**Types of apps developed?**

***Console-only***

***Static web (HTML)***

***Dynamic web (client)***

***Web (client & server)***

***Desktop GUI (type?)***

***Native Mobile GUI***

**Why are GUI apps different?**

**They're *Graphical* (Duh)**

***{ Widgets, Layouts, Graphical details, Animations, etc. }***

But this is *not* a graphics design class! Hint: this is a **CS** class. I am also *not* a graphics design expert.

But the good news is that there are tons of pre-existing widgets, layout tools, animations, etc., that we can reuse and tweak.

***Event-driven behavior***

***How do we handle this?***

We need to be able to write code that responds to user input that:

- may happen anywhere in the (reachable) UI
- may happen at anytime

And we want to respond with minimal delay, and without slowing down the system (e.g., by forcing it to repeatedly check for inputs).

Common ways of dealing with this:

- concurrency (e.g., foreground vs. background threads)
- asynchronous programming

***Scaling is complex***

Compared to text-driven UIs, which don't really care about resolution, screen size, etc., GUIs must scale up/down.

*Responsive design* = layouts that "automatically" scale based on screen size / resolution / other shifting UI specifications.

**For next time ...**

***Read through "Introduction to Dart"***