

**CS 398: Application Development**

## **Week 03 Lecture: Kotlin 2 & 3**

Collections; Object-Oriented Kotlin

# Schedule Change

## Concerns

- Excessive lecture content in Weeks 5-6
- This content should be delivered in-part before, not during, Sprint 1.

## Suggestion

- Reduce sprints from 4 to 3
- Gain 1 week before Sprint 1 to learn this material
- Gain 1 week at the end of the course to work on your final report.

**Sprint demos are still worth 60% overall, just split across 3 demos (4x15% vs 3x20%)**

Week 4	Jan 24 - 28	Design patterns; UX prototyping	Analysis & Design	Design review (15%)
Week 5	Jan 31 - Feb 4	<b>Projects; Build systems; Application Toolkits: JavaFX; Android</b>	Sprint 1 Kickoff	
Week 6	Feb 7 - 11	<b>Unit testing; Software Design (Ousterhout); Refactoring</b>	Sprint 1 Completion	<b>Sprint 1 Demo (15%)</b>

# Current Week (Revised)

Design review is Fri Jan 28th.

- Everyone will have a 15 min slot booked in-class.
- 10 min presentation - <https://student.cs.uwaterloo.ca/~cs398/b-templates/>
- Will include design patterns, and a screen mockup (next week)

Week 3 ◀	Jan 17 - 21	Architectural characteristics; Architectural patterns; Diagrams; Kotlin 2 & 3 (OO Kotlin)	Analysis & Design	Architecture
Week 4	Jan 24 - 28	Design patterns; UX prototyping	Analysis & Design	Design review (15%)
Week 5	Jan 31 - Feb 4	<b>Projects; Build systems; Application Toolkits: JavaFX; Android</b>	(Infrastructure)	Infrastructure
Week 6	Feb 7 - 11	<b>Software Design (Ousterhout)</b>	Sprint 1 Kickoff	
Week 7	Feb 14 - 18	<b>Unit testing; Refactoring</b>	Sprint 1 Completion	<b>Sprint 1 Demo (20%)</b>

in-person



# Q & A

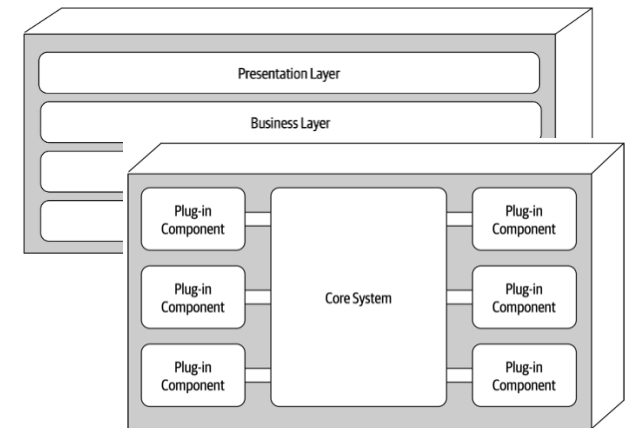
# How many NFRs should I have?

- Remember: MOST NFRs will not apply to your system!
- At most, you will find 1-3 that you think are relevant.
  - **Monolithic** application? Usability, Localization, Accessibility, Performance. ← Today
  - **Distributed** application? Performance, Scalability. ← Sprint 3

The process we're working through is applicable to ANY software system that you build. If you did this with a different application, you would highlight different characteristics.

# What should I diagram?

- **System diagram** (critical)
  - Pick an architecture diagram.
  - Add labels and descriptions to it based on what you're building.
    - Add UI, database/file storage.
    - Add key classes (once you know what to add).
- **Class diagrams** (optional)
  - Do you have 1-2 classes that are critical? Document them!
- **Activity diagram** (optional)
  - Do you have 1-2 key activities that would benefit from being documented?



You will continue adding to your system diagram next week, when we talk about design.





# Activities

# TODO Today

## Planning

1. Create project plan 




## Requirements

1. Pick users, (optional) create personas 
2. Interview people that fall into your role 
3. Identify requirements, (affinity diagram) 
4. Document requirements in GitLab 

### Reminders:

1. Quiz is due before the end of today!
2. We will do check-ins this morning (5 mins per team)

## Analysis & Design

1. Determine technical impact 
2. Choose architectural style 
3. System diagram 

### Next Week

4. Design Patterns
5. UI Mockup (Low-fidelity Prototype)