Introduction

What is this course about?

“This course explores knowledge, skills and strategies required to **build complex full-stack applications**. Using an iterative development methodology, students will work in **project teams** to design, develop, and test applications and services. **Standard development tools and approaches** will be used to ensure code quality and performance at every step of the development cycle.”

- This is a course about building modern, full-stack applications.
- As much as possible, we’re trying to mirror a commercial development environment (focus on teams, iterative development, “building the right thing”).

CS 346 LAB,LEC,TST 0.50

Application Development

Introduction to full-stack application design and development. **Students will work in project teams** to **design and build** complete, working applications and services using standard tools. Topics include **best-practices** in design, development, testing, and deployment.

Prereq: CS 246; Computer Science students only

* You will work in teams (of four) on a course project.

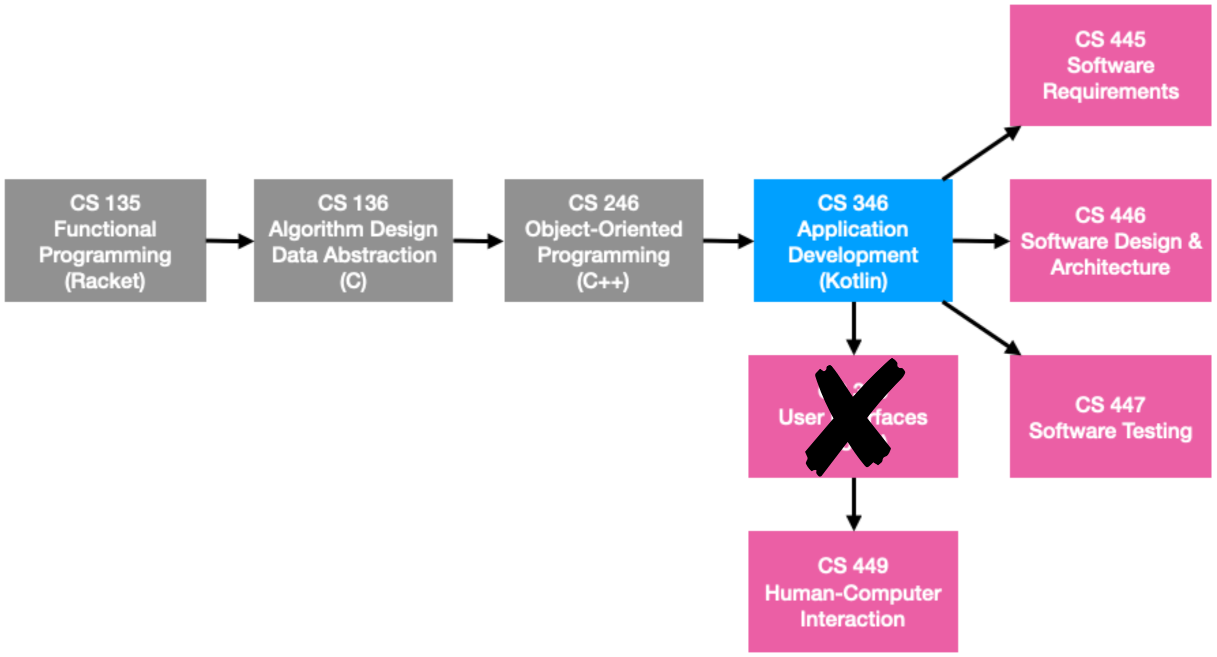
* Lectures will present material that you need to successfully build your project.

* You will walk through requirements definition, design, implementation.

* You will have lots of help!

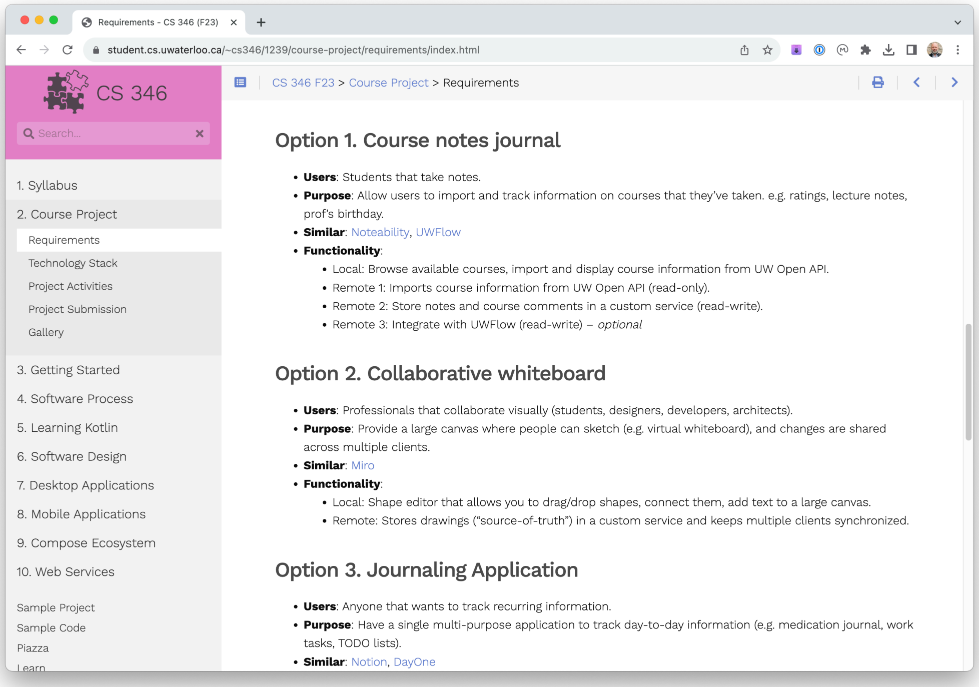
This course was envisioned as sitting somewhere between CS 246 Object Oriented Programming and CS 446 Software Design & Architecture. It's a great choice if you have interest in real-world software development, or software engineering.

It's one of the courses that you can use for the SE specialization.


CS 135 Functional Programming (Racket)
CS 136 Algorithm Design Data Abstraction (C)
CS 246 Object-Oriented Programming (C++)
CS 346 Application Development (Kotlin)
CS 445 Software Requirements
CS 446 Software Design & Architecture
CS 447 Software Testing
User Interfaces
CS 449 Human-Computer Interaction

Course project

- Teams of four (4) people
    - Same lecture and lab sections only!!
- Choose one of the suggested projects, or suggest your own!
- Work through project phases to:
    - Expand on the requirements.
    - Determine architecture and design.
    - Write code, unit tests, release notes.
    - Fully develop all features.



Option 1. Course notes journal

- **Users**: Students that take notes.
- **Purpose**: Allow users to import and track information on courses that they've taken. e.g. ratings, lecture notes, prof's birthday.
- **Similar**: Noteability, UWFlow
- **Functionality**:
    - Local: Browse available courses, import and display course information from UW Open API.
    - Remote 1: Imports course information from UW Open API (read-only).
    - Remote 2: Store notes and course comments in a custom service (read-write).
    - Remote 3: Integrate with UWFlow (read-write) – *optional*

Option 2. Collaborative whiteboard

- **Users**: Professionals that collaborate visually (students, designers, developers, architects).
- **Purpose**: Provide a large canvas where people can sketch (e.g. virtual whiteboard), and changes are shared across multiple clients.
- **Similar**: Miro
- **Functionality**:
    - Local: Shape editor that allows you to drag/drop shapes, connect them, add text to a large canvas.
    - Remote: Stores drawings ("source-of-truth") in a custom service and keeps multiple clients synchronized.

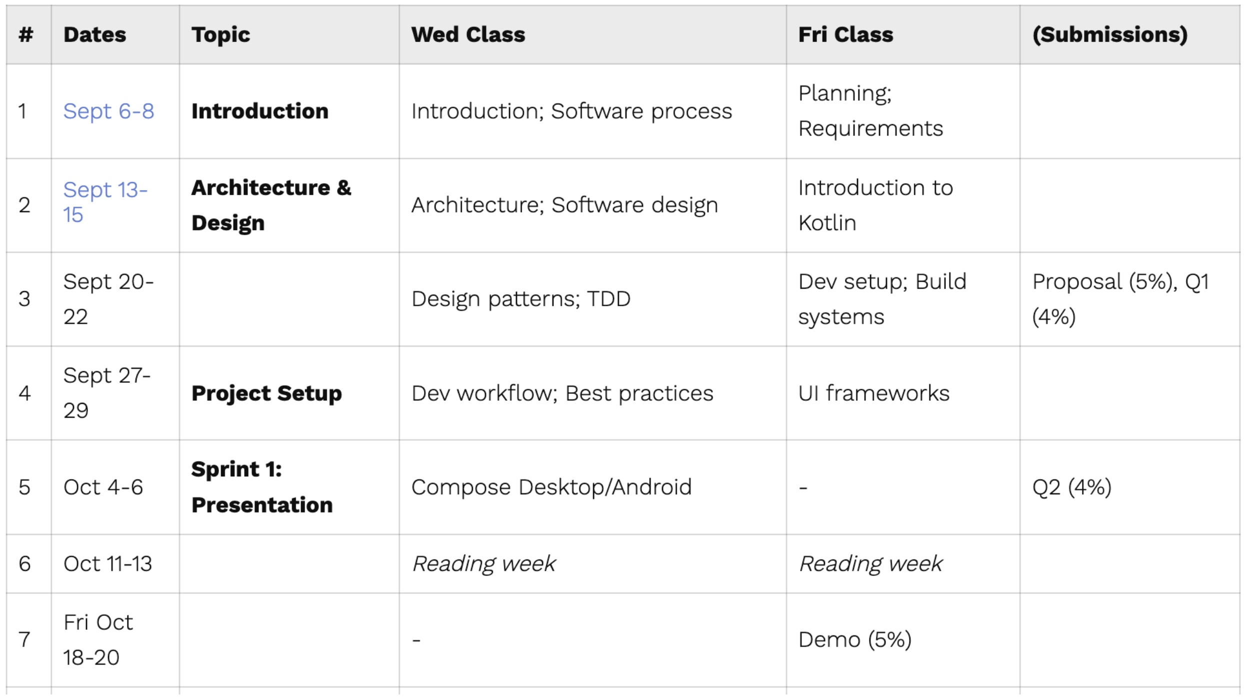
Option 3. Journaling Application

- **Users**: Anyone that wants to track recurring information.
- **Purpose**: Have a single multi-purpose application to track day-to-day information (e.g. medication journal, work tasks, TODO lists).
- **Similar**: Notion, DayOne

Basic requirements are provided, but you have lots of flexibility in the design, and you are expected to add more than just the required features! ★

Structure

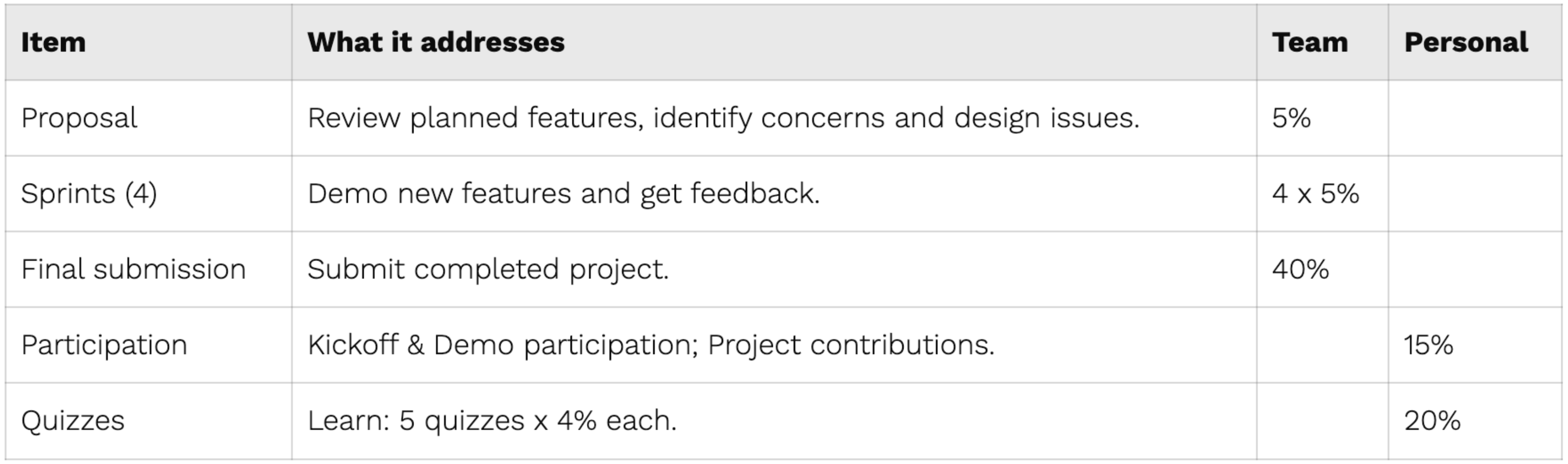
- Weeks 1-5
  - Lectures, demos
- Weeks 6+
  - In-class working sessions
  - Everyone should attend!
  - Get help from the instructor
  - Extra lectures
- Milestones
  - Two-week sprints, demos
  - Final project report



#	Dates	Topic	Wed Class	Fri Class	(Submissions)
1	Sept 6-8	**Introduction**	Introduction; Software process	Planning; Requirements	
2	Sept 13-15	**Architecture & Design**	Architecture; Software design	Introduction to Kotlin	
3	Sept 20-22		Design patterns; TDD	Dev setup; Build systems	Proposal (5%), Q1 (4%)
4	Sept 27-29	**Project Setup**	Dev workflow; Best practices	UI frameworks	
5	Oct 4-6	**Sprint 1: Presentation**	Compose Desktop/Android	-	Q2 (4%)
6	Oct 11-13		*Reading week*	*Reading week*	
7	Fri Oct 18-20		-	Demo (5%)	

https://student.cs.uwaterloo.ca/~cs346/1239/syllabus/schedule/index.html

Assessment

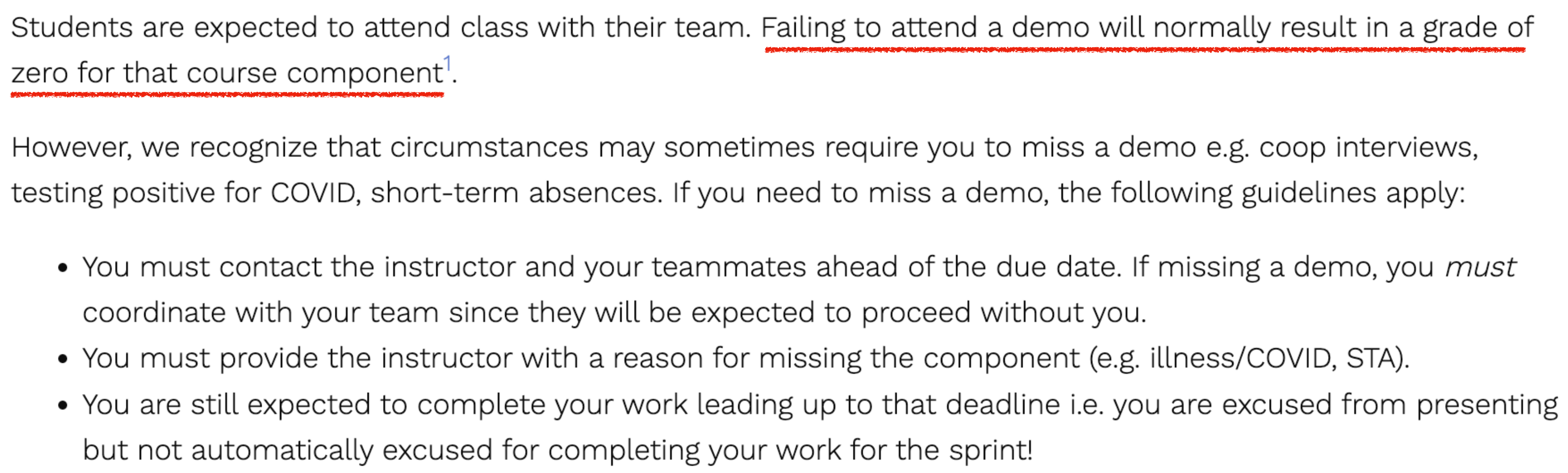


Item	What it addresses	Team	Personal
Proposal	Review planned features, identify concerns and design issues.	5%	
Sprints (4)	Demo new features and get feedback.	4 x 5%	
Final submission	Submit completed project.	40%	
Participation	Kickoff & Demo participation; Project contributions.		15%
Quizzes	Learn: 5 quizzes x 4% each.		20%

Mix of project team and personal assessments.

- Everyone on the team *usually* receives the same team grade for their project work (you can be docked marks if you miss demos - see course policies).
- Personal contributions are quizzes, and a grade that your team assigns you at the end of the course (you all get to evaluate each other, anonymously).

Absence policies



Students are expected to attend class with their team. Failing to attend a demo will normally result in a grade of zero for that course component[1].

However, we recognize that circumstances may sometimes require you to miss a demo e.g. coop interviews, testing positive for COVID, short-term absences. If you need to miss a demo, the following guidelines apply:

- You must contact the instructor and your teammates ahead of the due date. If missing a demo, you *must* coordinate with your team since they will be expected to proceed without you.
- You must provide the instructor with a reason for missing the component (e.g. illness/COVID, STA).
- You are still expected to complete your work leading up to that deadline i.e. you are excused from presenting but not automatically excused for completing your work for the sprint!

Normally you can miss, at most, one demo or other deliverable during the term without penalty.

https://student.cs.uwaterloo.ca/~cs346/1239/syllabus/course-policies