

Introduction

CS 346 Application Development

<https://student.cs.uwaterloo.ca/~cs346>

Contents

About the Course	2
Course Staff	3
Course Website	4
Course Syllabus	5
Catalog Description	6
Main Themes	7
What Will You Be Doing?	8
Course Structure	9
Assessment	11
Application Development	12
What is Application Development?	13
Modern Applications	14
Modern Development	18
What's Next?	21
Bibliography	23

About the Course

Course Staff

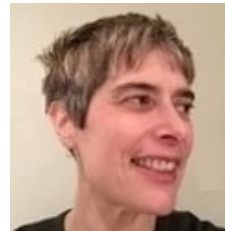
Instructor: Dr. Jeff Avery

- MC 6461 (by appointment)
- jeffery.avery@uwaterloo.ca



ISC: Caroline Kierstead

- MC 4008 (by appointment)
- ctkierst@uwaterloo.ca



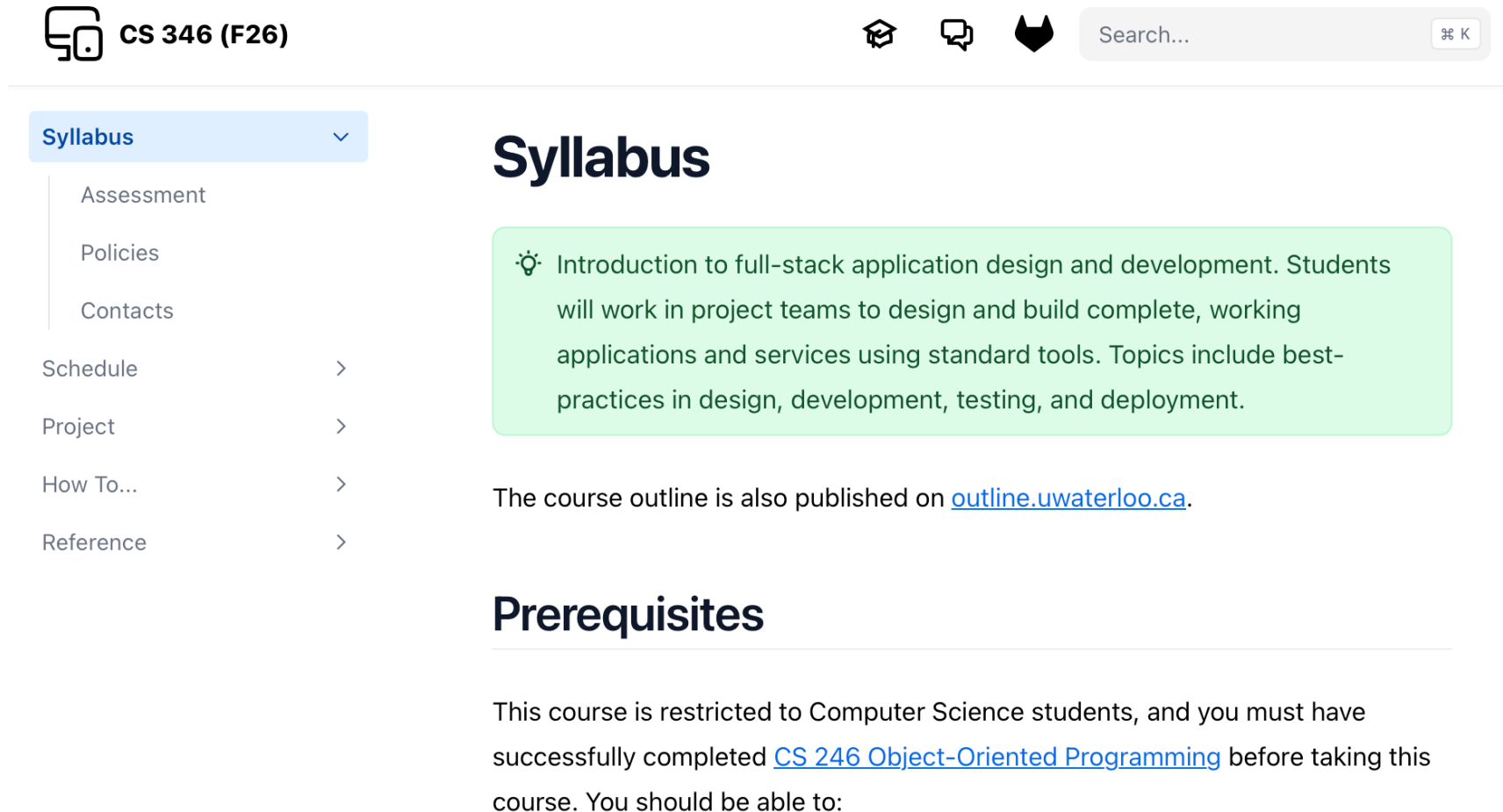
Teaching Assistants

- TBD

See [course website](#) for detailed contact information [1].

Course Website

The course website has everything you need, including these slides.



The screenshot shows the course website for CS 346 (F26). The header includes a logo, the course name, and navigation icons for a graduation cap, a speech bubble, and a cat. A search bar is also present. The left sidebar contains a menu with 'Syllabus' (selected), 'Assessment', 'Policies', 'Contacts', 'Schedule', 'Project', 'How To...', and 'Reference'. The main content area features a 'Syllabus' heading, a green box with a lightbulb icon and text about full-stack application design, and a link to the course outline. Below this is a 'Prerequisites' heading and text stating that the course is restricted to Computer Science students and requires completion of CS 246 Object-Oriented Programming.

CS 346 (F26)

Search...

Syllabus

- Assessment
- Policies
- Contacts
- Schedule >
- Project >
- How To... >
- Reference >

Syllabus

💡 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.

The course outline is also published on outline.uwaterloo.ca.

Prerequisites

This course is restricted to Computer Science students, and you must have successfully completed [CS 246 Object-Oriented Programming](#) before taking this course. You should be able to:

Figure 1: <https://student.cs.uwaterloo.ca/~cs346>

Course Syllabus

Catalog Description

From the course calendar:

“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.” [2]

Main themes:

- Designing and building complex applications.
- Learning best-practices for software development.
- Practice working on a team.

Main Themes

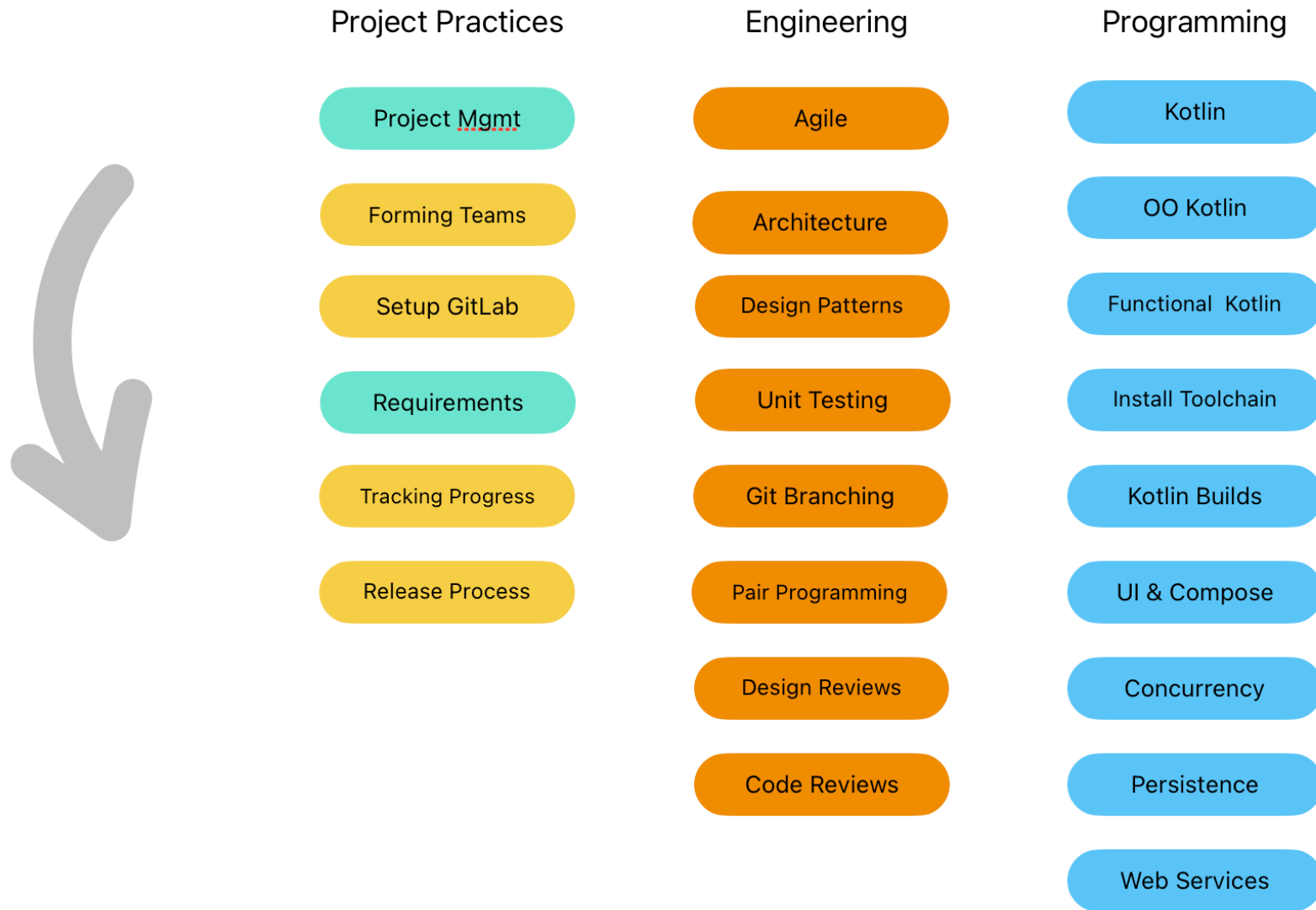


Figure 2: Although this is a CS course, SE and Project Management topics are discussed.

What Will You Be Doing?

You will spend most of your time and effort working on a software project:

1. You will join a project team
 - four people per team, same section
 - you and your team *choose* what to design + implement
2. You will learn our tech stack, toolchain, methods and practices.
 - most material delivered through online lectures.
 - classtime is for working on your project, watching instructor demos, getting help.
 - your instructor and your TA will be available to help you out.
3. You will deliver software releases through the term (four demos, plus a final release).
 - demonstrate features that you have designed and implemented.
 - get constructive feedback from your TA and instructor, and address it.
 - build high-quality software releases that can be installed and used!

Course Structure

1. Introduction

Getting started in the course! Managing projects and product requirements. Agentic workflows.

Dates	Topics	Wed	Fri
Week 01: Sept 7 - 11	Introduction, Process, Requirements	-	-
Week 02: Sept 14 - 18	Coding with AI	-	Check-In 1
Week 03: Sept 21 - 25	-	Demo 1	Demo 1 Test 1

2. General Development

Programming languages, build systems, architecture.

Dates	Topics	Wed	Fri
Week 04: Sept 28 - Oct 2	Kotlin Programming, Kotlin Toolchain	-	Check-In 2
Week 05: Oct 5 - 9	Architecture & Design, Design Patterns	-	Check-In 3
Week 06: Oct 19 - 23	-	Demo 2	Demo 2 Test 2

Figure 3: The [course schedule](#) shows the lecture topics and themes for each week.

Course Structure

Outside-Class (Ongoing)

- Lectures are pre-recorded and posted online.
 - Review required lectures each week.
 - Tests at the end of each section (30 mins, online).
- Meet with your team regularly.
 - Planning, design, coordination.
 - Working on software releases.

In-Class (Three-Week cycles)

- Two weeks of working on your project
 - Wed working session: instructor present to help (optional).
 - Fri demo: demo with your TA & instructor present (graded).
- One week of demos
 - A scheduled 20-minute demo with your TA.
 - Test due at the end of this week.
 - Time to plan the next iteration!

Assessment

Personal (20%)

Component	What is it?	Weight
Tests	Assigned lectures (4 x 5%)	20%

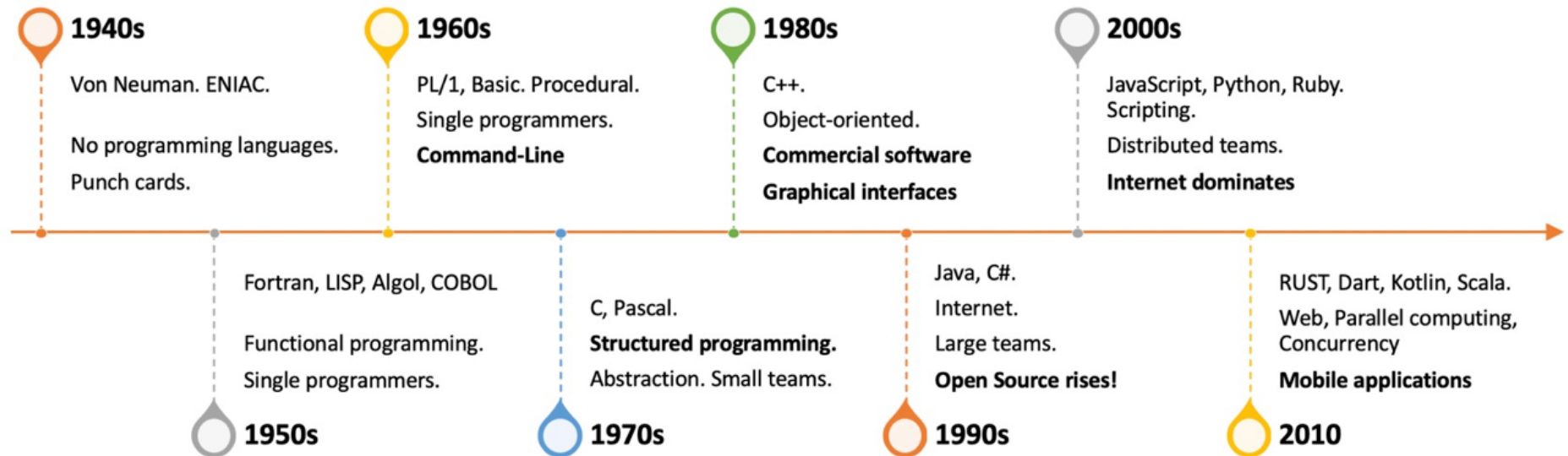
Team (80%)

Component	What is it?	Weight
Demos	Weekly demos (7 x 2%)	14%
Presentation 1	Proposal	10%
Presentation 2	Iteration + software release	10%
Presentation 3	Iteration + software release	10%
Presentation 4	Iteration + software release	10%
Final Submission	User guide + video	26%

NOTE: Attendance is expected for all team deliverables! If you fail to show up, or don't participate, you get a zero for that component.

Application Development

What is Application Development?



How has software development changed?

- **Complexity**: our software does more than software of last decade.
- **Distributed Systems**: Increasing dependency on networking, remote data.
- **Platforms**: textual -> graphical desktop -> web -> mobile.
- **Teams**: Individuals -> Increasingly large & diverse teams.

Modern Applications

Application development is about designing for end-users.

- Software should be useful, and enjoyable to use.
- It can be impactful and potentially used by millions of users.

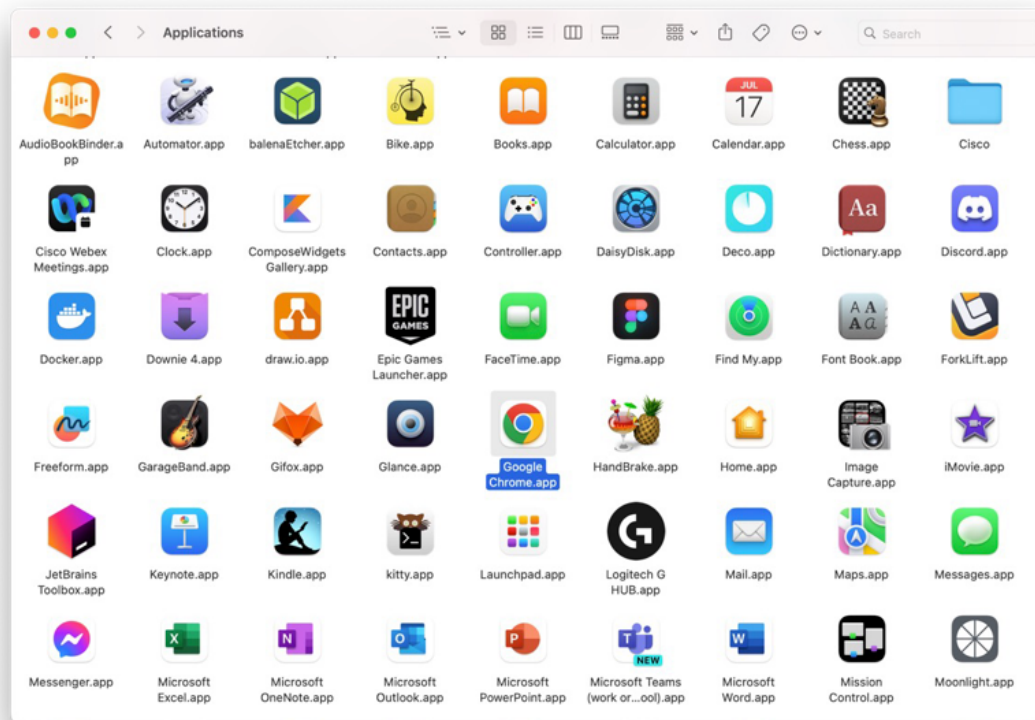


Figure 4: My applications folder, with dozens of applications.

Modern Applications

Applications often run on multiple platforms

- Desktop (Mac, Windows, Linux)
- Mobile (iOS , Android)
- Web (Chrome, Safari, Firefox)

Each platform has distinctive features

- Desktop: windowing, undo-redo, copy-paste.
- Mobile : device rotation, multi-touch gestures.
- Web: deep-linking to content, privacy, browser-compatibility.

Integration with other systems is necessary

- Local & remote data storage.
- Networking & access to remote services.
- Privacy & security features, concern for user data.

Modern Applications

What is a Full-Stack Application?

Historical: “front-end” (UI) and “back-end” (computation).

This aligns with the way that we deploy software:

- Front-end client (local machine)
- Back-end database/server (remote machine)

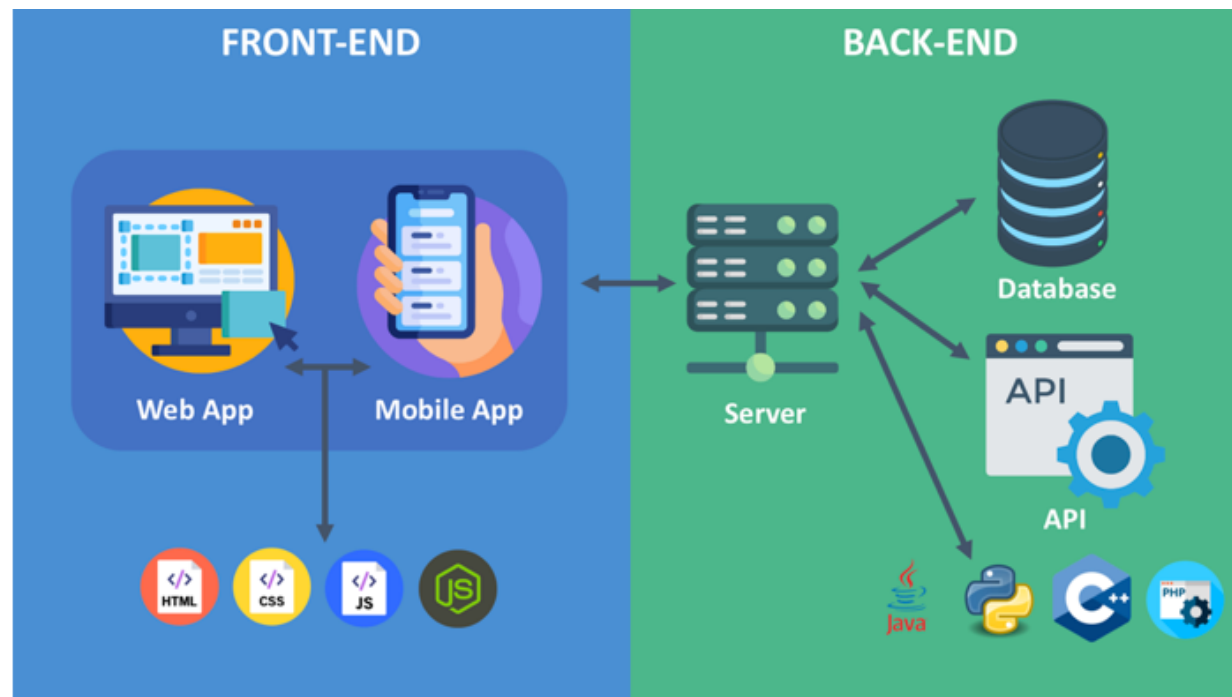


Figure 5: “Full-stack” logic is split between front and back-end.

Modern Applications

Example: Things

[Things](#) is an award-winning TODO application for the Apple ecosystem.

Client runs on every device/platform.

- Desktop for serious planning.
- Mobile for checking items.
- Watch for quick reminders.

“Things Cloud” hosts user data.

- Synchronizes data between devices.
- Premium subscription service.

Great example of clean, simple design that was very well-executed. They understand their users.

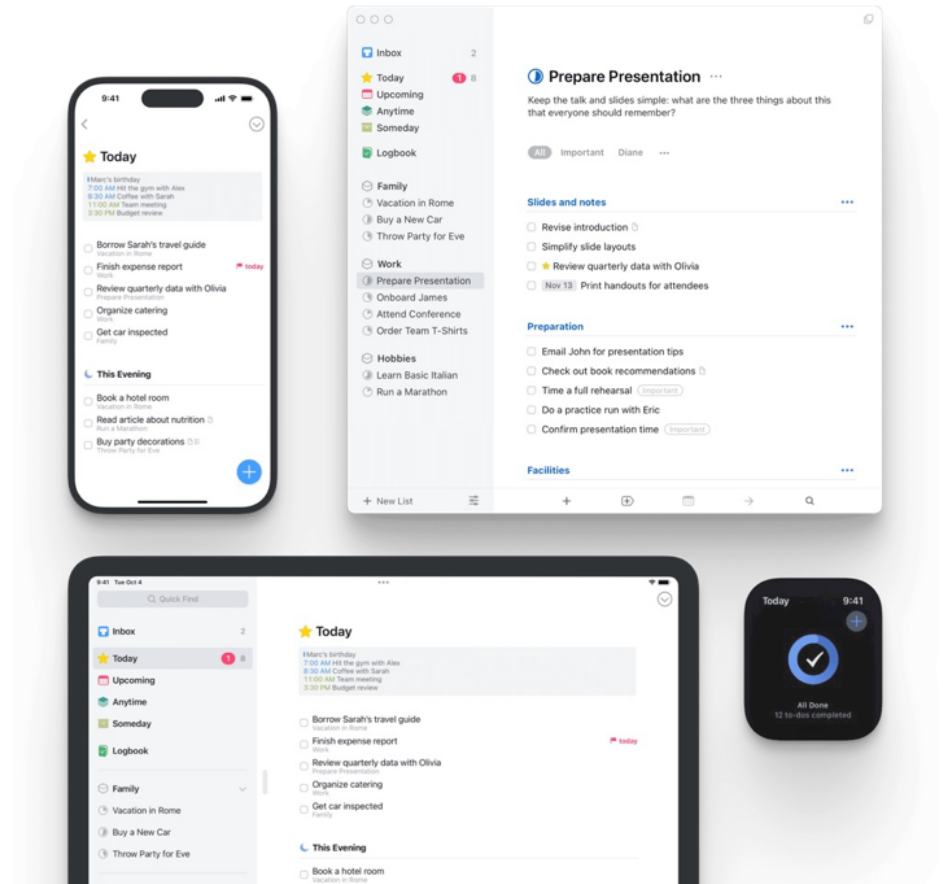


Figure 6: A cross-platform story makes an application more useful and compelling.

Modern Development

Programming language(s) + libraries are required.

- We need to develop for each target platform's native features.
- Additional requirements for graphics, database connectivity, networking, concurrency, security - not provided by language.

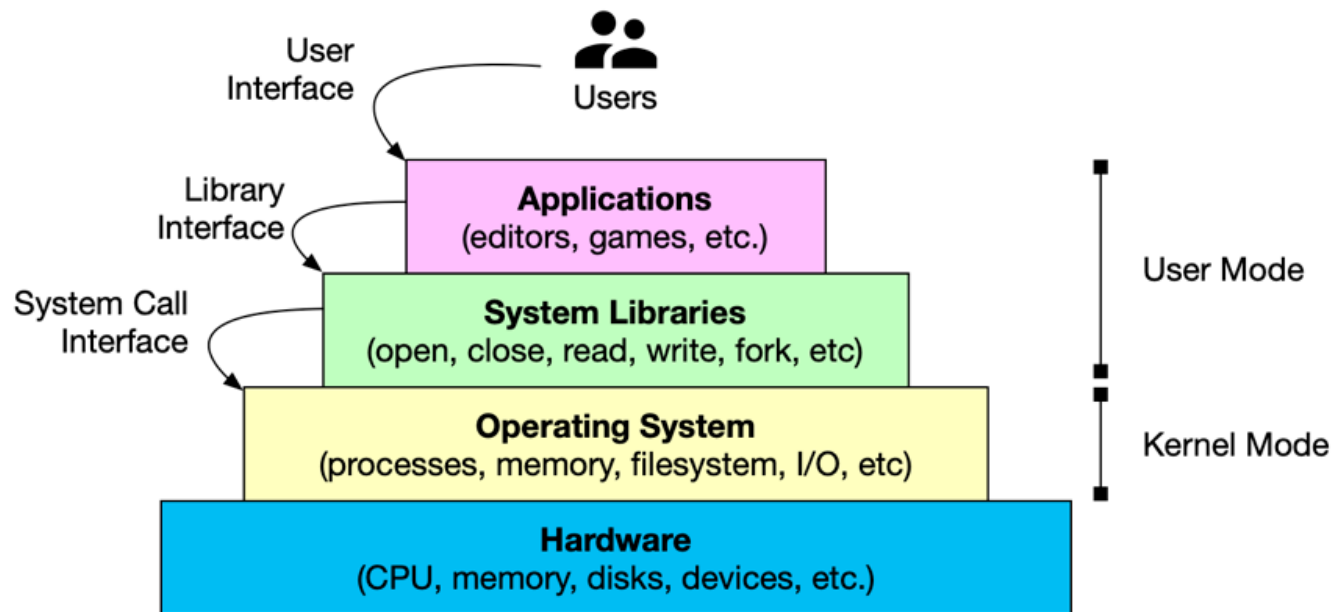


Figure 7: User libraries leverage underlying [OS functionality](#).

Modern Development

We need modern language features:

- Automatic memory management.
- Strong, static type systems.
- Functional & OO paradigms supported.
- High portability & support for multiple platforms.

We also need an ecosystem:

- Tooling and library support for missing functionality.
- Community support, including books, forums, training materials.

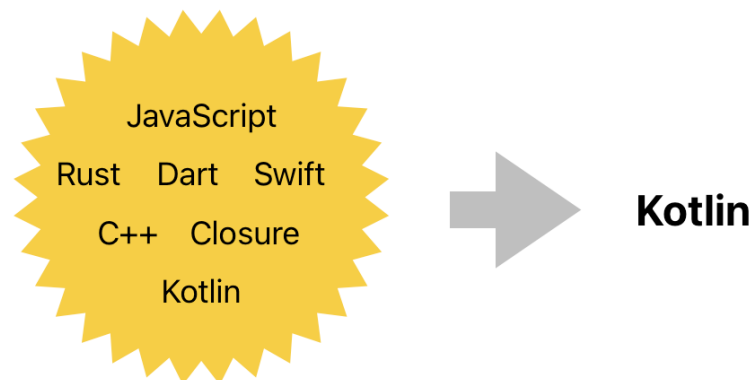


Figure 8: There are many choices for a tech stack. Kotlin is popular, and offers great tooling and multi-platform support.

Modern Development

“Do I still need to learn to code when GenAI can do it for me?”

I'm sure many of you have used GenAI on your courses, for assignments.

However, coding was never the hardest part of software development.

- Software projects tends to struggle because of human-issues: understanding requirements, communicating, making difficult decisions with customers.

Architecture and design become increasingly important as you develop larger and more complex systems.

- Is your AI solution scalable and maintainable? How extensible is it?
- If you rely exclusively on AI, what do you do when it fails to work?



You are allowed to use AI in this course, but you need to use it carefully.
We will discuss AI workflows in week 02!

What's Next?

There's a [weekly agenda](#) that lists what you should be doing each week.

- **Week 01**: lectures are recorded and being held in person.
- **Week 02+**: you are expected to watch online lectures before class.

Week 01 - Introduction

Your goals this week should be to meet people & form project teams. Once that's done, you can move on to setting up your GitLab project.

Lectures

- [Introduction](#)
- [Project Management](#)
- [Software Projects](#)
- [Requirements](#)

How To...

- [Form a team ↗](#)
- [Setup GitLab ↗](#)
- [Organize GitLab ↗](#)

Due This Week

- Nothing is due.

Figure 9: See the [weekly agenda](#) for required lectures and additional in-class topics.

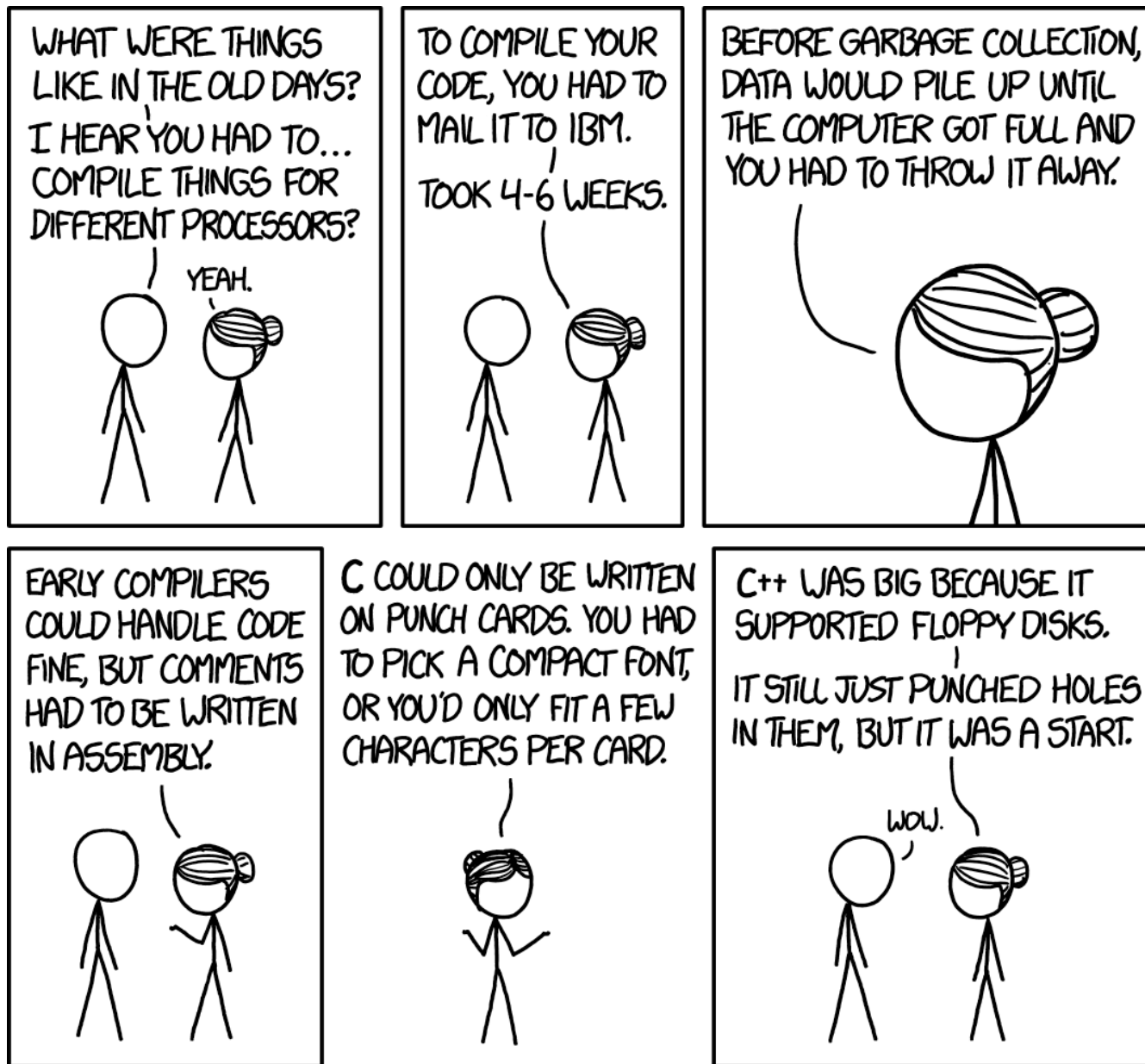


Figure 10: <https://xkcd.com/1755>

Bibliography

- [1] J. Avery, “CS 346 Application Development.” [Online]. Available: <https://student.cs.uwaterloo.ca/~cs346>
- [2] J. Avery, “CS 346 Course Calendar Entry.” [Online]. Available: <https://uwaterloo.ca/academic-calendar/undergraduate-studies/catalog#/courses/rkSgQ5NmYh>