Course Introduction

Modern Application Development Patterns (Web + Mobile + Cloud)

Jason Kuruzovich

2026-01-13

Welcome to ITWS-4500

Advanced Web Systems Development

Today's Agenda

Learning Objectives

1. Understand the course structure, expectations, and assessments
2. Learn a **pattern-first** way to think about modern applications
3. Compare common patterns across **web and mobile** clients
4. Preview backend, data, and deployment patterns you'll use all semester
5. Start thinking about project ideas and teams

About Your Instructor

Jason Kuruzovich

Associate Professor
Lally School of Management

- Research: AI/ML applications, digital platforms, entrepreneurship
- Industry: Enterprise software, startups, consulting
- Teaching: Web systems, data science, analytics

Contact: - Email: kuruzj@rpi.edu - Office: Pitt 2206 - Hours: Tuesday 9-11 AM or by appointment - Book: bit.ly/jason-rpi

Course Overview

Why This Course?

> Build **production-quality** applications as **integrated systems**.

. . .

This is not only about writing code. We'll focus on:

- **Architecture** — how systems are structured and why
- **Patterns** — reusable solutions to common problems
- **Trade-offs** — choosing "good defaults" and knowing when to change them
- **Integration** — making components work together
- **Operations** — deploying, monitoring, and evolving systems

Big Idea: Patterns Outlast Tools

Tools and frameworks change quickly.

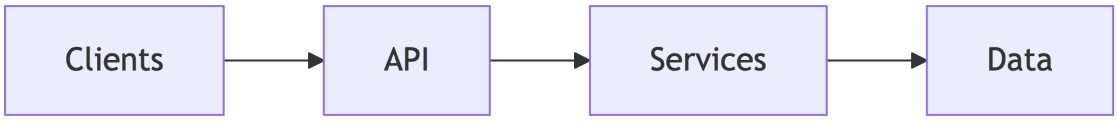
. . .

Patterns stick around:

- Client server communication
- Authentication and sessions
- Data modeling and persistence
- Scaling and reliability
- Dev workflows and deployment

A Pattern Map

The Full-Stack View



. . .

We'll revisit this diagram all semester and add details as your systems grow.

Client Patterns: Web vs Mobile

Web Applications

Common pattern:

- Browser UI (React, Vue, etc.)
- Communicates over **HTTP/HTTPS**
- Mostly **stateless** requests

Typical concerns:

- Routing
- Authentication
- State management
- Performance (latency, bundle size)

. . .

Mobile Applications

Common pattern:

- Native or cross-platform app
- Often the **same backend** as the web app
- More constraints (battery, offline use, flaky networks)

Key takeaway: Web and mobile usually share the same backend.

Backend Patterns

API-Centered Design

Most systems revolve around APIs:

- REST (common default)
- GraphQL (when it fits)

Typical responsibilities:

- Authentication & authorization
- Business logic
- Validation & error handling

. . .

Service Organization

You'll commonly see:

- **Monolith** — everything in one deployable
- **Modular monolith** — one deployable, clean internal boundaries
- **Microservices** — multiple deployables, more coordination

Rule of thumb: Start simple; add distribution when you have clear reasons.

Data Patterns

Databases (Common Choices)

- Relational: PostgreSQL, MySQL
- Document: MongoDB

. . .

What Matters More Than the Brand

- Data modeling (entities, relationships, constraints)
- Indexing and query patterns
- Security (least privilege, injection prevention)
- Migrations and change management

Data Access Pattern

Application → Data access layer → Database

Infrastructure & Deployment Patterns

From Laptop to Cloud

A common progression: 1. Local development 2. Containers (Docker) 3. Cloud deployment

. . .

Infrastructure as Code

Instead of manual setup:

- Configuration files
- Repeatable environments
- Version-controlled deployments

If it can't be reproduced, it's not done.

AI-Assisted Development

The New Reality

Modern developers use AI to:

- Generate boilerplate
- Explore unfamiliar stacks
- Debug and refactor faster

. . .

The Catch

> AI is most useful when you understand the underlying patterns.

This course will help you **ask better questions** and evaluate AI output critically.

How This Course Flows

Patterns → Tools → Systems

We will move from:

- Concepts and patterns
- To implementation and integration
- To deployment and operations

. . .

Coming Up

- Web & API architectures
- Databases and persistence
- Docker and deployment
- Multi-service systems

Expectations

- You do not need to know every tool in advance
- You *do* need to think clearly about design choices
- Focus on the **why** behind decisions

. . .

Your job this semester: build something real, learn patterns, explain trade-offs.

Logistics

- **Office Hours:** Tuesday 9–11 AM, Pitt 2206
- **Email:** kuruzj@rpi.edu
- **Appointments:** bit.ly/jason-rpi

. . .

Next Class

Docker and Infrastructure as Code