lectures # week-1

Course Introduction & Project Overview (Tuesday, March 31, 2026)

Lecture Recording

Watch on Panopto

Plan & Admin 0:00

First day of TCSS 460 — Client/Server Programming. Course overview, syllabus walkthrough, AI usage philosophy, course website tour, AI Diary assignment introduction, and group project overview. No code today — Thursday will be the first hands-on session with Express and TypeScript.

Discord Setup 1:19

Discord is the primary communication channel for the course. Canvas is for grades and Panopto recordings, but most interaction happens on Discord.

Key Channels 3:38

- **Announcements** — most course announcements will be posted here. Important ones get double-posted to Canvas (which emails you).
- **Ask Charles** — quick questions. If Charles can answer with a "yes" while grocery shopping, he will. Other students can also answer here.
- **Ask a Peer** — questions you'd rather ask classmates. Code sharing is fine — this course is collaborative.
- **Post Resources** — share interesting links, tools, or articles.

Important Setup Steps 2:39

1. Join the Discord server (link on the course site and Canvas)
2. Scroll to the top of the welcome message and react to the agreement
3. If you don't react, you won't get a role and will be kicked after 12 hours
4. Set your display name to your real name — not "Commander Poopy Pants"

> ⚠ **Privacy Note**
>
> Discord is not FERPA-compliant. You can share your own information as much as you want, but don't assume the server protects your privacy. For grade-related or sensitive topics, use email.

Don't Use This Server for Group Work 5:50

Use the course Discord for course-wide communication. When you form your group, create your own Discord server for group-specific work.

Canvas & Course Website 7:38

Canvas holds grades and Panopto recordings. The majority of course content lives on the **course website** — a fully integrated site built with AI assistance.

The course materials were created using generative AI (Claude), then read, edited, and refined by the instructor. Student feedback from previous quarters using similar sites was overwhelmingly positive. Feedback on this quarter's site is welcome and encouraged.

Syllabus Walkthrough 10:57

Instructor & Office Hours 11:28

- No fixed office hours. Available via Discord most of the day during the week.
- On campus Tuesday, Wednesday, Thursday — in-person meetings by request.
- Zoom meetings available anytime by request.

- Protects family time evenings and weekends, but responds early mornings (as early as 6:45 AM, even Sundays).

Course Description 14:50

The official catalog description references CGI, Perl, and technologies from 2006. It hasn't been updated since then. In practice, this course is **project-based full-stack web development** using TypeScript: PostgreSQL database → Prisma ORM → Node.js/Express API → React/Next.js frontend.

Grading Breakdown 20:14

Component	Weight	Details
Group Project	50%	Full-stack project, sprints throughout the quarter
Check-Off Assignments	25%	Weekly individual assignments, peer-verified
Reading Quizzes	20%	Canvas quizzes based on concept readings (Weeks 1–8)
AI Diary & Course Reflection	5%	Weekly AI usage log + end-of-quarter reflection

Check-Off Assignments 20:27

- One per week, designed to take about an hour
- You demo your work to a classmate who checks you off
- Peer gives you a score (0–10) on a shared Google Sheet
- The instructor does not individually grade these — peers handle it

Reading Quizzes 22:08

- Two types of reading material: **concept/theory** (what and why) and **language/framework guides** (how)
- Quizzes are on Canvas, 15 questions, timed at 30 minutes (2 min/question)
- Two attempts allowed, highest grade kept

- Question pools — no guarantee of getting the same questions on each attempt
- Can be done using just the search bar on the course site, but actually reading the material is more valuable

Attendance 27:43

Not required and not mandatory — except for group meeting days. All lectures are recorded on Panopto. If you find you get equal value from recordings and readings, that's fine. The instructor's goal is to bring enough value that showing up is worth it.

Academic Integrity 39:40

This course is highly collaborative. There's not much to "cheat" on. Collaborate, share code, help each other learn. The check-off system is honor-based.

Group Conflict Resolution 40:01

Communicate early and often. Don't let problems linger. Come to the instructor if needed — sometimes just having the instructor pop into a group's Discord channel resolves issues.

AI Usage Philosophy 29:26

AI tools are not optional in this course — they are expected and encouraged. The requirements are intentionally set high enough that you'll need AI to meet them efficiently.

The Core Message 30:08

> "AI is not going to take your jobs. The person who is using AI a lot better and a lot more efficiently than you — absolutely is taking your job."

Whether the AI bubble bursts or not, learning to leverage these tools costs you nothing and prepares you for the future. There is no space in this industry for someone who isn't using AI.

AI as a Force Multiplier 33:39

The instructor describes AI as a "force multiplier" — not reducing the amount of work, but massively increasing what can be produced. Working 2–3x as many hours with a 10x

multiplier means 20–30x the output compared to pre-AI workflows. This includes projects that simply wouldn't have been attempted without AI due to the time investment required.

Known Issues with AI 34:55

- **Environmental concerns** — acknowledged but not a reason to avoid AI in a professional context
- **Ethical issues** — training data provenance is a real concern
- **Hallucinations** — with modern models and good context/intent engineering, rare in practice. The bigger problem is **over-eagerness** — the AI does more than you asked for.
- **Privacy** — be careful with sensitive client or student information. The university uses Microsoft Copilot (on top of ChatGPT) for FERPA-compliant work, but the instructor finds it a poor tool for serious development.

Instructor's Tool Preferences 36:53

Tool	Use Case
Claude (Anthropic) — $100/month Max tier	All serious work. Concise, direct responses.
ChatGPT (OpenAI) — $20/month	Quick non-serious questions (coffee temperatures, etc.). Too wordy for real work.

Prompt Engineering vs. Context Engineering 55:19

This is a key concept for the course. The shift from prompt engineering to context engineering is compared to the shift from a flip phone to a smartphone — you can do the same things, but one requires dramatically less effort per interaction.

Prompt Engineering (The Flip Phone) 55:44

- Going to a web chat (ChatGPT, Claude.ai, Gemini) and writing a detailed prompt with all context every time

- You have to specify: project type, framework version, file structure, data shapes, expected output
- Every new chat window starts from zero — all context is lost
- It works, but it's slow and requires enormous effort per prompt

Context Engineering (The Smartphone) 59:00

- Using agent-based tools (Claude Code, Gemini CLI, GitHub Copilot, Cursor) that sit in your file system
- The tool reads your project files, configuration, and documentation on startup
- You build context once (e.g., in a `CLAUDE.md` file) and every session benefits
- Simple prompts like "let's get started" work because the tool already knows the project

Key distinction: The LLM (the model) is the same whether you use the web chat or the agent tool. The difference is the *tooling* that sits on top — it manages context, reads files, runs commands, and feeds rich prompts to the model on your behalf.

Intent Engineering (The Future) 1:02:52

Beyond context engineering — the AI not only knows your context but understands your *intent*. You can be even less specific because the tool knows what you're trying to accomplish. This is emerging and harder to achieve, but it's where things are heading.

Course Site Tour 42:30

Weeks Page 42:34

Each week has a page telling you everything you need to do: what to read, what's due, what guides to review, and links to the lecture demo repo. Start here if you're unsure what to work on.

Lecture Recaps 47:06

Every lecture gets a recap page generated from the Panopto transcript using Claude. These recaps include:

- Link to the Panopto recording
- Related assignments and code repos
- Timestamped section overview — click a timestamp to jump to that point in the recording
- Transcription corrections and tangent filtering
- Whiteboard photos (if someone reminds the instructor to capture them)

Concept Readings 50:18

Theory/concept material organized by week — networking, HTTP, web APIs, REST, databases, auth, etc. This is what the Canvas quizzes are based on. Week 1 is done; remaining weeks will be published as the quarter progresses (most content is written but still going through editorial review).

Guides 51:39

Practical, hands-on "how-to" material. All TypeScript-specific.

- **TypeScript Fundamentals** — JavaScript for Java developers, TypeScript essentials, objects/arrays, array methods, modules/imports, building and running TypeScript
- **Async Programming** — coming soon
- **Tools** — Node.js setup, VS Code/WebStorm, AI coding agent usage, AI tools setup (Gemini, Copilot, Cursor, Claude Code)
- **Back-End** — Express guides, data access, deployment
- **Front-End** — coming later in the quarter

> **Before Thursday**
>
> If you haven't used JavaScript or TypeScript, read through the TypeScript guides before Thursday's class. Thursday dives straight into Express code.

Check-Off Assignments 1:16:32

Check-Off 1 (due end of Week 2): Clone the lecture demo repo, run it, make some changes, demo to a peer. The check-off Google Sheet is linked from the Canvas assignment (not on the course site, for FERPA compliance).

AI Diary 1:18:38

Accept the GitHub Classroom assignment. Every week, answer four reflection questions about your AI usage. Commit weekly — don't fill it all in at the end. Five to ten minutes per week. Not submitted on Canvas — the instructor checks your commit history.

The diary is for self-reflection and also helps the instructor understand how students are using AI to improve future teaching.

Group Project Overview 1:21:55

Full-Stack Architecture 1:25:52

Every group builds a full-stack application:

- **Database** — PostgreSQL for storing ratings and bug reports
- **Web API** — Express/Node.js sitting on top of the database, also proxying to TMDB (The Movie Database)
- **Frontend** — React/Next.js web application (built in weeks 6–10)
- **Auth** — instructor-hosted auth server (already built and running)

Even-numbered groups work with **movies**, odd-numbered groups work with **TV shows** from TMDB.

The Cross-Group Swap 1:26:36

This is the distinctive feature of the project:

- Weeks 1–5: Build your back-end API
- Week 6+: Build a front-end — but **not on your own API**. Two other groups' APIs are assigned to you.
- Your front-end must work with someone else's back-end code
- You submit bug reports through a bug reporting system built into the APIs
- The API team has 24 hours to fix reported bugs

What If You Get a Bad API?

That's real life. You work with what the client gives you. Your grade won't be impacted by a poor API — the instructor has done this for several quarters and accounts for it. The real motivator: nobody wants to be the group that delivers a bad API to other teams.

Documentation Matters 1:30:18

Poor documentation shows up as bug reports. If other teams can't figure out how to use your API from your Swagger docs, they'll report it as a bug — and they should.

Groups 1:22:04

- 4 people per group (some groups of 5 if enrollment requires it)
- Self-selected this quarter (a change from the usual random assignment)
- Sign up in Canvas by Thursday; remaining students will be assigned
- Groups of 3 are allowed if you're the first to request it

Division of Labor 1:34:45

Sprint requirements are intentionally vague — groups decide how to implement features. Perfect 4-way splits of work are unrealistic. Expect uneven workloads sprint to sprint.

If you already have web API or React experience: **teach, don't code**. Mentoring your group members is more valuable than doing it yourself. You'll learn more by teaching.

Schedule 1:39:25

- After Sprint 0 (Week 2), one lecture day per week (Thursdays) and one group meeting day (Tuesdays)
- Group meetings are in-person during class time — the instructor does one-on-ones with each group for demos and check-ins

---

AI Tools for This Course 1:05:01

Four tools are available. The first three are free for students:

Tool	Cost	Notes
Google Gemini (CLI + Code Assist)	Free	Free year of Gemini Pro for students
GitHub Copilot	Free	Free via GitHub Student Developer Pack
Cursor	Free	Free year of Cursor Pro for students
Claude Code	$20/month	No free student tier. The instructor uses this exclusively.

All four support agent mode — reading your project, editing files, running commands. The course guides walk you through setup for each one. You can use multiple tools simultaneously (e.g., Copilot for inline completions + Gemini CLI for terminal agent work).

AI Usage is Expected

The course requirements are intentionally set so that you need AI tools to meet them efficiently. This is by design — learning to leverage AI is a core learning objective.

Tool Update Speed 1:10:47

Claude Code (and similar tools) update 1–2 times per day with new features. The tools evolve faster than the models they sit on top of — which means the LLM may not know about its own tool's latest features. If you're asking about a brand-new feature, tell the agent to research it rather than relying on its training data.

What to Do This Week 42:47

1. **Join Discord** and react to the welcome message
2. **Read the concept readings** — Networking Fundamentals and HTTP & the Web
3. **Take Quiz 1** on Canvas (due Sunday night)
4. **Set up your dev environment** — Node.js, VS Code or WebStorm

5. **Read the TypeScript guides** — especially if you haven't used JS/TS before
6. **Clone the lecture demo repo** — you'll need it for Thursday and for Check-Off 1
7. **Accept the AI Diary** GitHub Classroom assignment and start your Week 1 entry
8. **Sign up for a group** in Canvas by Thursday (or be assigned)

This lecture outline is part of TCSS 460 — Client/Server Programming, School of Engineering and Technology, University of Washington Tacoma.