SE/COM S 3190 - Construction Of User Interfaces
Final Project

Published On: April 16, 2025, 12:00 AM CST
Due Date: May 11, 2025, 11:59 PM CST

Contents

Prepared by: ***Pranava Sai Maganti***

This is a team project. Please review the rubric carefully, as points will be awarded for effective communication and collaboration. Ensure tasks are evenly distributed between both members. Plan accordingly and complete the project the right way.

If you need help, try office hours—the schedule is on ***Canvas***. Lots of help is also available through the Piazza discussions. Please start the assignment as soon as possible to get your questions answered right away.

1 Overview

This document outlines the detailed requirements, expectations, and grading rubric for the Final Project in SE/COM S 3190 – Construction of User Interfaces. The Final Project serves as a comprehensive demonstration of each student team's learning, growth, and practical application of course content throughout the semester. It reflects not only your technical competence in full-stack development but also your ability to collaborate effectively, communicate ideas, and deliver a polished and meaningful solution to a real-world or fictional problem.

The Final Project is structured to replicate a professional development cycle—starting with project proposals, followed by planning and modular implementation, and ending with final documentation, and presentation. The project is carried out in teams of two and incorporates both individual accountability and collaborative execution.

Teams are expected to demonstrate strong fundamentals in front-end and back-end development, employ thoughtful design strategies for UI/UX, build a scalable system architecture, and document their technical process through detailed planning artifacts and reports.

1.1 Purpose of the Final Project

- To demonstrate mastery of web development skills including React, Node.js, Express, and database integration (MongoDB or MySQL)
- To design and implement a production-ready Single Page Application (SPA) or Multi-Page Application (MPA)
- To simulate the lifecycle of a real software engineering project: ideation, development, debugging, deployment, and delivery
- To promote collaboration, peer learning, and real-time problem solving in a team setting
- To develop written and verbal communication skills through structured documentation, live demonstrations and retrospectives.

1.2 Learning Objectives

By completing this project, you will:

- Apply theoretical concepts into a hands-on, tangible full-stack software solution
- Gain experience designing system architectures with clear separation of concerns
- Improve front-end design with responsive layouts and user interaction flows
- Implement backend systems that include route handling, middleware, and database queries
- Integrate APIs for CRUD functionality and asynchronous data operations
- Understand security, validation, and error-handling in production-ready applications
- Communicate technical processes through written documentation and oral presentation

1.3 Team Collaboration Guidelines

- Teams must have exactly two members (assigned on Canvas)
- Responsibilities should be divided equally — both frontend and backend work must be shared
- Each member must take ownership of specific features, completing both frontend and backend for those components
- Commits in GitLab must reflect ongoing, equal contributions from both members
- Any collaboration issues must be brought to the TA/instructor before the submission deadline

1.4 Additional Note

1. The purpose of this project is not just to test your coding skills — it is designed to measure your ability to produce clean, structured, and scalable code that solves a meaningful problem, backed with quality documentation, clear architecture, and logical reasoning.
2. The project requires thoughtful planning, creative thinking, and technical precision. Teams that plan early, divide responsibilities clearly, and iterate frequently will find success. Start early, ask questions often, and aim to build something you're proud of.
3. Incorporating advanced technologies used to solve technical problems won't be penalized. However, if the inclusion of technologies different to the required interferes with the review of the project at the time it is reviewed by the TA, the TA can decide whether to penalize the project or not.

1.5 Final Timeline Overview

Proposal — *2%* — **Already Submitted**

Final Project Components Due on May 11, 2025:

Software Implementation — *5%*
Documentation + Video — *3%*

Weekly Mini-Assignments begin April 16. Each will have a 1-week deadline. Completion of these tasks contributes significantly to your final project grade.

Final Project Demo: May 12–15, 2025

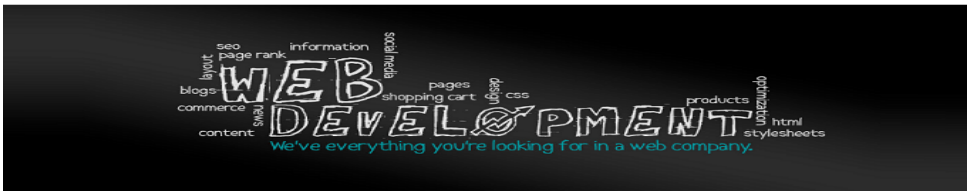
Demo — *7%*
Additional details regarding demo scheduling will be shared by the TAs during the first week of May.

1.6 What you already know!

Now that everyone has submitted their proposals, you should have decided between the two project options:

- **Option 1:**
  - Scaling your Midterm Project into the Final Project by adding new features and restructuring your existing HTML, CSS, and JavaScript code into a full-fledged React and Node.js application.
  - If you chose to build on your Midterm Project, your Final Project will be evaluated in comparison to it. We will specifically look for meaningful improvements and the addition of new functionalities.
- **Option 2:**
  - Starting a completely new idea from scratch, which involves designing new wireframes, developing the UI/UX, and implementing the planned features for the application.
  - If you started with a new idea, your Final Project should be comprehensive and clearly convey the project's vision, scope, and execution strategy as outlined in your proposal.

There are no restrictions on which option you choose. However, please keep the above mentioned criteria in mind.



2 Software Requirements

The Software component of the Final Project accounts for **5% of your overall grade** and represents the core deliverable of your team's work. It is the final implementation of your planned project and should demonstrate completeness, technical depth, and professional quality.

The development process is supported by a structured sequence of **Mini-Assignments** released weekly starting on **April 16, 2025**, and continuing through **May 11, 2025**. Each Mini-Assignment is designed to help teams incrementally build the application's architecture, UI/UX, backend functionality, and integration logic. If your team consistently completes all Mini-Assignments on time, you will have developed a strong and near-complete foundation for the final submission.

However, your final submission must go beyond baseline functionality. It should demonstrate final refinements, a polished user experience, responsive design, and full integration across all views and features outlined in your project proposal. The project should reflect production-level quality in both design and functionality. A complete `CODE FREEZE` must be in effect by **May 11, 2025**. Any commits or pushes made after this deadline will incur significant penalties, with deductions of up to **50%** of your Final Project grade.

2.1 Final Application Expectations

Your application must:

- Be a Single Page Application (SPA) or Multi-Page Application (MPA) built in React
- Include user-friendly navigation, responsive layouts, and complete CRUD functionality
- Reflect a clean, professional, and accessible design
- Connect to a database (MongoDB or MySQL) and store persistent data
- Implement full user interaction flows with routing and error handling
- Follow all file structure listed below

2.2 Functionality and Quality Benchmarks

Complete Implementation: The final version must implement 100% of the features defined in the proposal and any required views.

Smooth Navigation: Use React Router (or equivalent) for clean transitions between pages. Avoid reloads or broken navigation flows.

Visual Design: Your project's UI must demonstrate a reasonable level of effort and intentionality. While you are free to determine the aesthetic style, applications with minimal styling or placeholder content will be penalized. Tools like Tailwind CSS, Bootstrap, and custom CSS modules are welcome.

Responsiveness: Ensure that your interface adapts appropriately to different screen sizes, including desktop, tablet, and mobile views.

Advanced Features: At least one technically advanced feature is required. Suggestions include:

- Real-time search or filter functionality
- Admin panel with authentication logic
- File uploads with preview
- External API integrations (e.g., maps, weather)

2.3 Technical Requirements

Your application must include:

- **Frontend Technologies:** React (with JSX), React Router
- **Backend Technologies:** Node.js, Express.js
- **Database:** MongoDB or MySQL
- **API Integration:** RESTful endpoints using Fetch or Axios
- **React Hooks:** Must include `useState` and `useEffect`
- **Async Functions:** Use of `async/await` where required for API requests

CRUD Operations (Required) You must implement the following operations for at least two distinct data types (e.g., products, users):

- POST – Add new items
- GET – Fetch and display list of items
- PUT – Update item data (e.g., title, price)
- DELETE – Remove item by ID or name or some unique key

2.4 Views and Pages

Your application must contain the following views:

1. **Login and Signup** – User authentication with proper validation
2. **Home or Landing Page** – Introduction and entry point for the app
3. **Detail/Edit Page** – Allows viewing or editing of individual items
4. **Intermediate Process Page** – Simulates a transactional workflow (e.g., checkout)
5. **Confirmation Page** – Display success messages after actions (e.g., form submission)

6. **About Page** – Includes:
    - Course: SE/COM S 3190 – Construction of User Interfaces, Spring 2025
    - Team member names and ISU emails (photos optional, but encouraged)
7. **Company/Client View (Optional)** – If your project was developed for an organization

2.5 Folder and File Structure

Directory Organization:

- `teamXX_final/` – Root project folder
- `frontend/` – React codebase
    - `src/`
        - components/
        - assets/
        - `App.jsx`
        - `main.jsx`
- `backend/` – Node/Express server
    - `api/` – API endpoints, Database schema definitions, Logic for each endpoint
    - `config/` – DB connection files and environment variables
    - `server.js`
- `Documents/` – Architecture PDF, planning files, demo video

2.6 Use of JSON and External Files

Static data (if used) must be stored in external JSON files and put in the `assets` folder in the `frontend` directory. Do not hardcode data into the application logic. JSON may be used for dropdown values, UI themes, or demo datasets but not for core database-driven records.

Allowed Libraries and Frameworks

- CSS: Tailwind, Bootstrap, custom
- API: Fetch, Axios

Note on Penalties and Technical Scope You are encouraged to explore advanced libraries and technologies. However, if the inclusion of external libraries or tools causes the reviewers difficulty in running or evaluating your project, the TA reserves the right to deduct points. Your application must be self-contained and runnable on any standard development machine.

3 Documentation Requirements

Your documentation serves as the technical narrative of your Final Project. It explains what your system does, how it is structured, and why specific design and implementation decisions were made. The purpose of the documentation is not only to provide a written record of your development but also to help the TA and instructor assess your understanding of web development practices, architecture, and feature integration.

This deliverable is worth 1.5% of your total grade. The document must be clear, comprehensive, and professionally formatted. **You are required to prepare this documentation using `LaTeX`.** Submissions using other editors will not be accepted unless exported from LaTeX. Overleaf ensures consistent styling, easier formatting of code snippets, and better integration of visual elements.

1. Format and Submission

- Submit a single PDF file named: `TEAM_NAME_Technical_Report.pdf`
- Push the file to the Documents folder in the GitLab team repository by the Final Project deadline (May 11, 2025)
- Recommended tool: Overleaf downloaded as PDF

2. Suggested Structure Your documentation should follow a logical structure that presents your project in a way that's easy to follow. Below is a detailed breakdown of the required sections.

Cover Page

- Course code and title: SE/COM S 3190 – Construction of User Interfaces
- Final Project Documentation
- Semester: Spring 2025
- Full names and ISU email addresses of both team members

Table of Contents

Use auto-generated headings to make your document easy to navigate.

1. Introduction

- A brief overview of the project, what problem it solves, and why it is relevant
- Description of the application's target users and key features
- State whether your idea is original or inspired by an existing application

2. Project Description

- Description of all major functionalities
- Explanation of your UI flow — e.g., "Login leads to Dashboard, which allows CRUD operations on Posts"
- Clear mention of your CRUD operations and what entities they apply to

3. File and Folder Architecture

Required Description:

- Describe the physical layout of your project folder: `frontend/`, `backend/`, `Documents/`
- Mention any static files, external libraries, or helper functions

Optional Diagram: Use a simple folder tree diagram using bullet points or ASCII text (no ER diagrams or database schemas required).

4. Code Explanation and Logic Flow

- Describe how your frontend interacts with your backend (e.g., through API routes)
- Mention how you use state and props in your components
- Provide 2–3 code snippets (with explanations) that show:
  - A key React component
  - One of each backend route (e.g., POST, GET, PUT, DELETE)
  - Your method of communicating with the database (MongoDB or MySQL)

5. Web View Screenshots and Annotations

- Include full-page screenshots of each of your views/pages
- Annotate or describe each screenshot with the following:
  - What the page does
  - What user actions are allowed
  - What happens after an interaction (e.g., a successful form submission)

6. Installation and Setup Instructions

This is one of the most important sections — your application must be runnable.

- Clearly outline steps to run your `frontend/` and `backend/` servers
- Include installation commands for:
  - Node modules (e.g., `npm install`)
  - Database setup (MongoDB local or Atlas, or MySQL setup steps)

- Include commands to start the servers (e.g., `npm run dev`, `nodemon server.js`)
- Mention any .env variables needed to run the backend

7. Contribution Overview

- List all features and who implemented them
- Example:
  - Team Member 1 Name — Login/Signup, Product CRUD Backend, Tailwind Integration
  - Team Member 2 Name — Home Page UI, Filters, Confirmation View, Documentation
- Ensure that each student owns both frontend and backend portions

8. Challenges Faced

- Mention 2–3 specific issues you encountered during development
- Describe how you solved them (e.g., state conflicts, API issues, async bugs)

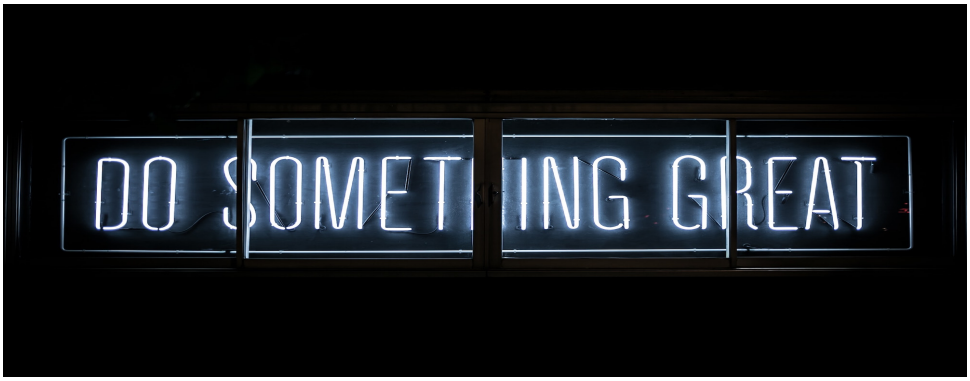
9. Final Reflections

- Summarize what you learned through this project
- Reflect on what you might do differently next time
- Optionally, mention whether you would want to deploy this app in the future

3. Evaluation Criteria Your documentation will be evaluated based on:

- Completeness of sections outlined above
- Clarity, readability, and technical explanation
- Quality and structure of the writing
- Proper use of screenshots and visual support
- Setup instructions that work and allow reproducibility

Reminder: Do not include unnecessary technical artifacts such as entity-relationship diagrams, full database schemas, or low-level algorithms. The goal is to show what you built, how it works, and how to run it — all through the lens of applied web development.



4 Video Requirements

Your project video serves as a concise and professional walkthrough of your application. It is your opportunity to demonstrate the core features, overall system structure, and user experience in a clear, narrated format. The video helps instructors and TAs quickly assess the functionality and completeness of your project.

This component is worth 1.5% of your final grade. A well-produced video will clearly highlight your application's architecture, interface, and features while demonstrating technical correctness and usability.

4.1 1. Video Duration and Format

- Length: **3 to 5 minutes**. Videos exceeding this limit may not be fully watched.
- Format: **MP4 only**. Do not use unsupported formats such as .mov, .avi, etc.
- Screen Resolution: At least 720p preferred.

4.2 2. Recording Guidelines

- **Record your screen using screen capture software** such as OBS Studio, Screenity, QuickTime, or Screencast-O-Matic.
- **Narrate your video using your own voice**. Clearly explain what is being shown on the screen as you walk through the application. **Both team members must actively present the portions of the project they developed**, providing clear explanations of the features, logic, and interactions involved.
- **Mobile phone recordings are not accepted.** Videos must be desktop-recorded to ensure high resolution and clarity.
- Minimize background noise and ensure your voice is clear and audible.
- Multiple takes may be necessary — we encourage you to rehearse and revise.

4.3 3. Content Checklist

Your video must include the following elements, in logical order:

- **Introduction:** State your name, project name, and what the application does
- **Folder Overview:** Briefly show your folder structure — frontend, backend, and documents
- **Walkthrough Your Code:** Briefly show key portions of your code (e.g., React component, API route, or database call). This part should be concise and should not exceed **1 minute and 10 seconds**.
- **Running the App:** Show both frontend and backend servers running
- **UI Navigation:** Demonstrate at least 3–4 views/pages (e.g., Home, CRUD Page, Confirmation, About)
- **Data Handling:** Showcase two full CRUD operation (one per team member) and how it updates in the database
  - **Note:** Full CRUD operation includes demonstrating POST, GET, PUT and DELETE.
- **Key Feature Highlight:** If you implemented an advanced feature (e.g., file upload and preview, search bar), demonstrate it.
- **Wrap-Up:** Conclude by summarizing the experience or stating that full technical details are provided in the documentation briefly.

4.4 4. Submission Requirements

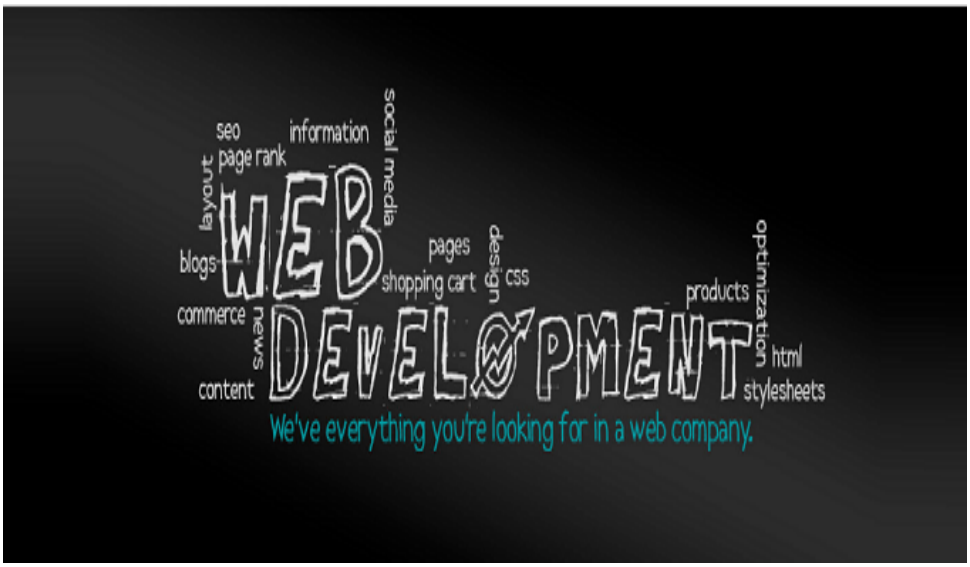
- Submit a single MP4 file named: $\texttt{TEAM}_{N}AME_{F}inal_{V}ideo.mp4Pushthevideointo the Documents fol$
- File size should not exceed 250MB (compress using HandBrake or Clipchamp if necessary)

4.5 5. Evaluation Criteria

TAs and instructors will assess your video based on:

- Clarity of narration and video quality
- Logical sequence of content
- Coverage of the required checklist
- Demonstration of a working system and visible CRUD operations
- Overall professionalism and completeness

Reminder: This video is a core part of the evaluation process. It should not be rushed. Treat it as a professional demonstration — like a product pitch — that communicates your application's structure, features, and technical completeness.



5 Live Demo Presentation Requirements

The live demo is a mandatory, real-time presentation in which both team members must showcase their Final Project to their assigned TA or instructor. This component offers an opportunity for students to explain the technical and functional aspects of their project, demonstrate collaboration, and answer questions to validate individual and shared understanding of the system.

This presentation is worth 5% of your final grade. It is a key indicator of your project readiness, technical knowledge, and communication ability.

1. Scheduling and Format

- Demo scheduling details will be shared by your assigned TA via Canvas and Discord during the first week of May. You must schedule your live demo exclusively with your designated TA.
- Scheduling with a TA other than the one assigned to your team will not be permitted. Unauthorized scheduling may result in disqualification from the demo component and a potential score of zero.
- Each team will be allocated a **20-minute slot**. Slots will be filled on a first-come, first-served basis, so we recommend signing up early.
- All demo presentations will be held **in person**, either in Pearson 0145 or Atanasoff B005, depending on your TA's location. These locations are subject to change. Please check with your TA.
- Teams must arrive on time with their entire application fully set up and operational.
- Please be aware that as this is the Final's Week, some demo sessions may require additional time due to project complexity, troubleshooting, or extended Q&A. If other teams experience technical difficulties or require additional time, it may delay your scheduled slot. Plan accordingly and allocate sufficient time before and after your demo window.

2. Demo Structure and Expectations

The live demo will be divided into 3 main phases:

1. **Introduction (3 minutes):**
    - Briefly introduce the team and the project idea
    - State what problem it solves and the core technologies used
2. **Live Walkthrough (12 minutes):**
    - Show the full functionality of your application
    - Navigate through key views and explain the logic and flow
    - Demonstrate at least one CRUD sequence end-to-end
    - Highlight any advanced feature implemented (e.g., filters, file uploads, dynamic dashboards)
    - Show backend console outputs and/or database changes for full-stack validation
3. **Q&A and Technical Justification (5 minutes):**
    - Be prepared to answer detailed questions on:
        - Folder and file structure (`frontend/`, `backend/`, `Documents/`)
        - APIs used and routes implemented
        - Use of React Hooks (`useState`, `useEffect`)
        - Database model(s) and persistence strategy
        - Application setup, configuration, and dependencies

3. Equal Participation is Mandatory

- **Both team members must be present and actively participate in the demo.**
- Each student should clearly explain the portions they implemented.
- Regardless of how tasks were split, **both students must demonstrate full knowledge of the entire application** — frontend, backend, API, and database.
- TAs may ask either student any question, even about features they did not directly code. Full familiarity is expected.

4. Evaluation Criteria

You will be assessed based on:

- Organization and clarity of presentation. Functionality and completeness of the application
- Technical depth of your explanations

- Collaboration and equal participation. Ability to answer technical questions accurately

Reminder: The live demo is a professional interaction. Test your setup in advance, and speak clearly. This session allows the instructor and TA to validate that you understand what you built and how it works.

6 Final Project Evaluation Rubrics

6.1 Software Component Rubric (Spring 2025)

Item	**Item Description**	**Points**
Application Functionality	App runs successfully without crashes; routes and actions behave as expected	35
CRUD Implementation	Complete implementation of CRUD for at least two data entities (frontend + backend + DB)	50
UI/UX Design	Interface is responsive, visually cohesive, and user-friendly with a clear interaction flow	35
Routing and Navigation	Use of React Router with proper navigation, fallback views, and seamless transitions	35
React Hooks	Correct usage of useState, useEffect, and any advanced state logic	20
Backend/API Integration	API endpoints are properly implemented and tested using fetch or axios	25
Advanced Feature	A non-trivial feature is included (e.g., filters, file uploads, external APIs, dashboards)	25
Folder/File Organization	Adheres to required structure: `frontend/`, `backend/`, `Documents/`	25
Total		**250**

6.2 Documentation Rubric

Item	**Item Description**	**Points**
Formatting	Submitted a PDF that was prepare using typset formatting with LaTeX with clean layout and styling	10
Introduction and Project Summary	Clear overview of the project's purpose, users, and goals	10
Architecture Overview	Clear explanation of folder structure, technologies, and integration flow	15
Code Snippets	Meaningful code segments with explanation of functionality	10
Screenshots and View Descriptions	Well-annotated images of application pages	15

Setup Instructions	Steps to run frontend/backend locally with configuration steps	15
Work Contributions	Breakdown of features by team member	10
Reflection	Challenges faced and key takeaways clearly articulated	15
Total		**100**

6.3 Video Rubric

Item	**Item Description**	**Points**
Length and Format	Video is 3–5 minutes long, in MP4 format	10
Narration	Both team members explain their contributions clearly	20
App Demo	Demonstrates app execution with working views	20
CRUD Flow	Shows at least one complete CRUD workflow	20
Folder Structure	Shows and explains frontend/, backend/, and documents	5
Code Walkthrough	Brief overview of selected code segments within 1 min 10 sec	15
Technical Completeness	Shows outputs, database interaction, and advanced features	10
Total		**100**

6.4 Demo Rubric

Mini-Assignments contribute directly to your Demo rubric score (80 points out of 375).

Item	**Item Description**	**Points**
Presentation Structure	Clear introduction, logical flow, and project explanation by both team members	35
Participation	Equal engagement and communication of each member's contributions	50
System Functionality	Complete walkthrough of app demonstrating stable and interactive features	80
Q&A Readiness	Correctly answers technical questions related to routing, CRUD, folder structure, and config	65
Technical Fluency	Demonstrates deep understanding of the full system architecture and integration	65
Mini-Assignment #1	Initial planning and screen sketches submitted on time	5
Mini-Assignment #2	React routing and two working views implemented	25
Mini-Assignment #3	Backend API + database integrated and one working CRUD operation	25

Mini-Assignment #4	Advanced feature added and app styling polished for responsiveness	25
Total		**375**

7 Submission Details & Deadline

Due Date: May 11, 2025, 11:59 PM CST
Format: Push everything to GitLab by 11:59pm

NO LATE SUBMISSIONS will be accepted

Your GitLab repository at the time of the deadline will be considered your final submission. All updates after the deadline will result in penalties.

8 Grade Release and Review Window

Final project grades will be released by 12:00 PM (noon) on May 17, 2025 on Canvas. You will have time until 6:00 PM on the same day to contest your grade if you believe there has been a mistake or oversight.

Use this window effectively and ensure any concerns are communicated clearly through email or discussed during the review hours posted on Canvas. We plan to finalize and submit the course grades to the Registrar's Office shortly after, so no changes will be possible after this time.