CS 343 Frontend Web Development Project

Team-Based Claim System (1000 Points Maximum)

Project Overview

Teams of 3 students will create a useful frontend web application that:

- Uses **vanilla HTML/CSS/JavaScript** (no frameworks like React/Vue)
- Integrates with a **third-party API** from the public-apis repository
- Utilizes **local storage** for data persistence
- Features **responsive design** and **accessible UI**
- Duration: **10 weeks**

Grading Scheme

Each claim is worth a certain number of points. The grading scale is as follows:

Points	Letter Grade
900-1000	A
800-899	B
700-799	C
600-699	D
Below 600	F

In addition, the following requirements must be met for a letter grade:

- Application must be deployed and accessible via URL to receive a D or higher.
- A deployed application with only partial Basic Claims implemented will receive a D (650).
- Application must implement all Basic Claims to receive a C or higher (720).
- Application must implement all Basic and eight Intermediate Claims to receive a B or higher (800).
- Application must implement all Basic, eight Intermediate, and two Stretch Claims to receive an A (900).

Each extra Intermediate Claim beyond the required eight adds 10 points. A third Stretch claim adds 50 points. No more that three Stretch claims can be counted. Points are capped at 1000.

Two-Person Teams

Teams of two students face slightly lowered requirements:

- Application must be deployed and accessible via URL to receive a D or higher.
- A deployed application with only partial Basic Claims implemented will receive a D (650).
- Application must implement all Basic Claims to receive a C or higher (720).
- Application must implement all Basic and six Intermediate Claims to receive a B or higher (800).

- Application must implement all Basic, six Intermediate, and one Stretch Claim to receive an A (900).

Each extra Intermediate Claim beyond the required six adds 10 points. A second Stretch claim adds 50 points. No more that two Stretch claims can be counted. Points are capped at 1000.

When to Make Claims

A claim should be made when the feature is fully implemented and tested. Claims can be made at any time during the project, but it is recommended to make claims as soon as possible to ensure they are evaluated before the project deadline.

Claims may generally be made at two times:

1. **During a lab or work session**: Ask the instructor to evaluate your claim. He will check the feature and, if it meets the requirements, mark it as completed and award points.
2. **In office hours**: If you are unable to make a claim during a lab or work session, you can visit the instructor during office hours to have your claim evaluated.

Note: **No more than three non-Basic claims can be made in a week.** Plan accordingly. Claims cannot be made once final exam week begins.

Claims marked with an asterisk (*) indicates that they are collectible at the end of the project only, as they involve the whole application and not just a single feature.

Note: Claims are binary (complete/incomplete) and must be demonstrable during evaluation.

BASIC CLAIMS (Minimum Viable Application)

These claims represent the absolute minimum requirements for a functioning web application. Without all of these, the application would not be considered complete.

- **B1**: Created a landing/home page
    - *Assessment*: Page exists, loads without errors. The page may be a dashboard view, a welcome page, etc., but should provide a clear entry point to the application and its features.
- **B2**: Implemented a navigation menu that links to all major sections
    - *Assessment*: Menu is visible on all pages, all links function correctly and navigate to intended pages
- **B3**: Built at least 3 distinct pages/views (home, main feature, about/contact)
    - *Assessment*: Application has 3+ unique pages with different content and purposes, each accessible via navigation
- ***B4**: *Created a consistent page layout structure across the application
    - *Assessment*: All pages share common header/footer/navigation positioning and styling
- **B5**: Integrated with a chosen third-party API and successfully retrieved data

    - *Assessment*: API calls execute successfully, data is fetched and displayed on the page, API source is documented

- **B6**: Implemented localStorage or OPFS to save and retrieve user data
    - *Assessment*: Data is saved to localStorage or OPFS, persists after page refresh, can be retrieved and displayed
- **B7**: Added at least one form of user interaction (button clicks, form input, etc.)
    - *Assessment*: Interactive elements respond to user input and produce visible changes or actions
- **B8**: Created dynamic content that changes based on API data or user input
    - *Assessment*: Page content updates in response to API data or user actions without full page reload
- ***B9**: Implemented proper semantic HTML5 elements throughout
    - *Assessment*: Uses appropriate HTML5 tags (header, nav, main, section, article, footer) instead of generic divs
- **B10**: Applied custom CSS styling beyond browser defaults
    - *Assessment*: Visual styling is clearly customized with colors, fonts, spacing, and layout different from unstyled HTML
- ***B11**: Implemented responsive design that works on mobile devices (320px+ width)
    - *Assessment*: Application is usable and content remains accessible when viewed at 320px width and larger
- ***B12**: Added basic accessibility features (alt text, proper headings, sufficient color contrast)
    - *Assessment*: Images have alt text, heading hierarchy is logical (h1->h2->h3), text/background color contrast ratio meets WCAG guidelines

---

INTERMEDIATE CLAIMS (Enhanced Features)

- **I13**: Added error handling for API failures with user-friendly messages
    - *Assessment*: When API fails (test by disconnecting internet), application displays helpful error message instead of crashing
- **I14**: Implemented loading indicators during API requests
    - *Assessment*: Visible loading spinner, progress bar, or "loading..." text appears during API calls
- **I15**: Created a search or filter functionality for displayed data
    - *Assessment*: Users can enter search terms or select filters that reduce/modify the displayed dataset

- **I16**: Added form validation with clear error messages
    - *Assessment*: Invalid form inputs trigger specific error messages, form submission is prevented until corrected
- **I17**: Implemented data sorting capabilities (by date, name, category, etc.)
    - *Assessment*: Users can click buttons/dropdown to reorder displayed data by different criteria
- **I18**: Added CSS animations or transitions for improved user experience
    - *Assessment*: Smooth visual transitions occur on hover, click, or page changes (not just instant state changes)
- **I19**: Created multiple data display formats (list view, grid view, cards, etc.)
    - *Assessment*: Users can toggle between 2+ distinct visual layouts for the same data without page reload
- **I20**: Implemented pagination or "load more" functionality for large datasets
    - *Assessment*: Data is presented in chunks with navigation controls or load-more button for additional content
- **I21**: Added user preference settings that persist in localStorage (or OPFS)
    - *Assessment*: Users can modify settings (theme, display options, etc.) that save and restore on page reload
- **I22**: Created interactive elements with hover/focus states
    - *Assessment*: Buttons, links, and interactive elements visually change on mouse hover and keyboard focus
- **I23**: Implemented favorite/bookmark functionality with localStorage (or OPFS) persistence
    - *Assessment*: Users can mark items as favorites, favorites persist after page reload, can be viewed separately
- **I24**: Added data export feature (download as JSON or text file)
    - *Assessment*: Users can click a button to download current data/results as a file to their computer
- **I25**: Integrated multiple API endpoints or different data sources
    - *Assessment*: Application successfully fetches and displays data from 2+ different API endpoints
- **I26**: Created a user dashboard or summary view
    - *Assessment*: Dedicated page/section shows aggregated information or user-specific data overview

- **I27**: Added keyboard navigation support for all interactive elements
    - *Assessment*: All buttons, links, and form elements can be reached and activated using only keyboard Tab/Enter/Space
- **I28**: Added skip links and improved heading hierarchy
    - *Assessment*: "Skip to main content" link at page top, headings follow logical order (h1->h2->h3, no skipping levels)
- **I29**: Created comprehensive alt text and ARIA labels for complex UI elements
    - *Assessment*: Images have descriptive alt text, interactive widgets have ARIA labels, screen reader friendly

---

STRETCH CLAIMS (Advanced Challenges)

- **S30**: Created data visualization components (charts, graphs, interactive maps)
    - *Assessment*: Application displays data in visual chart/graph format using canvas, SVG, or charting library
- **S31**: Added real-time data updates or live feeds
    - *Assessment*: Data refreshes automatically without user action, or displays live-updating information
- **S32**: Implemented drag-and-drop functionality for user interface elements
    - *Assessment*: Users can drag items from one location to another, changes are visually reflected and functionally meaningful
- **S33**: Created image/file upload and preview capabilities
    - *Assessment*: Users can select files from their device, preview appears in the application
- **S34**: Built a single-page application with client-side routing
    - *Assessment*: URL changes without page reload, browser back/forward buttons work, different URLs show different content
- **S35**: Added shareable URLs that restore specific application states
    - *Assessment*: Copying and pasting URL in new browser window/tab restores the same view/data/filters
- **S36**: Created custom keyboard shortcuts for power users
    - *Assessment*: Specific key combinations (Ctrl+S, etc.) trigger application functions, shortcuts are documented
- **S37**: Built interactive tutorials or guided tours

    - *Assessment*: Step-by-step overlay or modal system guides users through application features

- **S38**: Created a progressive web app (PWA) with installability
    - *Assessment*: Browser offers "Install app" option, manifest.json exists, service worker registered
- **S39**: Implemented geolocation features with user permission handling
    - *Assessment*: Requests user location permission, displays different content based on location
- **S40**: Implemented unique, creative features not covered by other claims
    - *Assessment*: Feature is genuinely novel and adds significant value (instructor discretion on uniqueness/creativity)
    - You must get prior approval from the instructor *early* in the project if you want to claim this.
- **S41**: Created print-optimized layouts and print CSS
    - *Assessment*: Print preview shows clean, readable layout different from screen version
- **S42**: Added internationalization support (multiple languages)
    - *Assessment*: Users can switch between 2+ languages, all interface text changes accordingly
- **S43**: Implemented advanced accessibility patterns (ARIA live regions, complex widgets)
    - *Assessment*: Uses ARIA live regions for dynamic content, complex widgets follow WAI-ARIA authoring practices
- **S44**: Built comprehensive user help system or documentation
    - *Assessment*: Dedicated help section with searchable documentation, FAQ, or context-sensitive help
- **S45**: Added touch/gesture support for mobile interactions
    - *Assessment*: Swipe, pinch, or other touch gestures work on mobile devices and trigger specific actions