**DESIGN:**

URL: https://mehrbawan.github.io/swe632\_hw3/

**Scenarios**

**Scenario 1:**

A frontend developer needs a responsive design for a login page, including username and password entry, a “remember” and “forget password” option, and more at the start of their design process. Instead of writing the HTML, CSS, and JavaScript from scratch, they describe the requirements to the UI Code Assistant on the Generate page, which generates the code instantly.

**Scenario 2:**

A UX designer in need of design review uploads a PNG mockup of their site’s completed landing page to the UI Code Assistant on the Analyze page. The tool analyzes the layout, and suggests improvements (e.g., "Increase font size for better readability") to successfully better the design in terms of accessibility and usability.

**Scenario 3:**

A frontend developer needing to verify their page’s layout in the middle of the programming process pastes their HTML/CSS code for a checkout screen into the UI Code Assistant. The tool flags inefficiencies (e.g., "Use CSS Grid to optimize layout") and suggests optimizations.

**Argument**

UI development is time-consuming and error-prone, especially for developers without lots of frontend experience. Existing tools like Bootstrap provide building blocks and can aid in speeding up the process but lack personalized feedback for a given developer’s goals. The process can be further slowed down due to lengthy code and design reviews. An AI-driven tool like UI AI can optimize the frontend development process by speeding up small, repetitive tasks through code generation as well as instant code and design feedback on demand.

**Prototype Plan**

The prototype is implemented as a web app known as AI UI with the goals outlined above. The design has three main components to provide well-rounded frontend support:

* Code Generator: Text area for describing components + dropdown for framework (HTML/React/Vue). Outputs code snippets (as in generator.html).
* Design Analyzer: Image file upload for designs/URL input for live sites. Returns AI feedback (analyze.html).
* Code Review: Paste actual code to receive suggestions (review.html).

**Socratic Questioning**

* + Use Socratic questioning to critique your scenarios and the argument
    - Ensure you cover all four types of questions covered in class 8
    - Document questions and answers

**Socratic Questioning to Critique Design**

**Clarification Questions**

* *What are the main functions of the UI Code Assistant?*  
  → It generates components, analyzes existing code, and evaluates UI designs for improvement.
* *Is this tool intended for beginner or expert developers?*  
  → Both. Beginners can benefit from the code generation feature for tasks they lack skills in and can learn best practices, while experts save time on repetitive tasks.

**Assumption Questions**

* *What assumptions am I making about the problems developers face?*  
  → That developers spend too much time on boilerplate code and manually reviewing UI code, and perhaps that my own personal experiences in frontend dev are universal.
* *Am I assuming that automated suggestions are always beneficial?*  
  → Yes, but to account for this the app should work in a way that users can easily accept or reject suggestions such as through redirection or regenerating reports.

**Evidence Questions**

* *How do I know developers need this tool?*  
  → From complaints found on code forums like Stack Overflow and Reddit, as well as in-person testimony from students learning frontend development in colleges.
* *What existing tools support this idea?*  
  → Tools like GitHub Copilot and ChatGPT being used for code-related issues suggest there's a growing market for AI-powered code assistance.

**Implication and Consequence Questions**

* *What happens if this tool fails to give accurate suggestions?*  
  → Users may leave unsatisfied and lose trust in the tool which may lead to a drop in users.
* *What are the consequences of relying on the assistant too much?*  
  → Developers may stop learning the fundamentals of frontend dev and become over-reliant on tools. To counter this, we can add educational tooltips and links to documentation or in-house tutorials.

**IMPLEMENTATION:**

**Walkthrough**

**Front Page**A screenshot of a computer

AI-generated content may be incorrect.

Regular front page layout. Theme is clean and modern inspired by other AI tools like ChatGPT, DeepSeek, and UXPilot.

A screenshot of a computer screen

AI-generated content may be incorrect.

Quick UI search bar on front page has jQuery autocomplete for popular queries users may be asking.

**UI Code Generator**A screenshot of a computer

AI-generated content may be incorrect.

“Generate HTML/JavaScript code for standard components such as login, shopping cart, etc.” Center panel for custom code generation, side buttons add code for popular components into the code box.

A screenshot of a computer

AI-generated content may be incorrect.

Ex. pressing “data table” will provide premade code and a preview (preview does not work for all items as they’re bootstrap which I couldn’t get to work in the preview lol oops)

**Design Analysis**A screenshot of a computer

AI-generated content may be incorrect.

“Analyze your UI design and provide suggestions.” Uploading picture or linking website and pressing generate custom AI report on the right (right now dummy AI text).

**Code Review**A screenshot of a computer

AI-generated content may be incorrect.

“Analyze your UI code and determine any inefficiencies, improvements.” Here instead of UI layout images, paste code to receive tips and reviews. Like above, not actually AI but should generate dummy AI review.

**Testing**

**INTERFACE METRICS:**

**Learnability**

|  |  |
| --- | --- |
| **Metric** | **Examples in Project** |
| **Gulfs of Execution** | * Explicit sidebar on side always to show how to navigate. * Gulfs bridged by clear instructions, intuitive and familiar pages and setup. All buttons needed are on screen. |
| **Affordance** | * Buttons like "Generate Study Plan" and "Generate Weekly Schedule" are designed to look clickable/show selectability on hover. * Input fields are clear and method is evident:   + Self-entry text fields are white, while boxes where the TOOL will put text are tinted to indicate no clickability |
| **Signifier** | * Appropriate icons for each sidebar heading gives idea of each page (ex. code symbol for the code generator, code doc symbol for reviewer, stat symbol for analysis, etc.) * Page you are on is highlighted in sidebar. * Popular iconography for AI is used to show that it is an AI-driven tool (magic wand, sparkles, and lightning bolts are all rapidly becoming signifiers for AI). |
| **Gulfs of Evaluation** | * Quick and concise error messages upon erroneous behavior (ex. when pressing “generate” without writing any text in any of the AI query boxes) |

**Interface Architecture**

**Component States**

* Active/inactive navigation items (highlighted when selected)
* Form input states (empty, filled, submitted)
* Code generation state (before/after generation)
* Analysis report state (empty/loaded)

**Events**

* + Click events for navigation buttons and form submissions
  + Input events for search bars and text areas
  + Load events when fetching component data from JSON files
  + Hover events for interactive elements like buttons, cards (color change/glow)

**Model-View-Controller**

**Model**

* + Data in JSON files (queries.json, components.json)
  + Definitions in HTML code

**View**

* + HTML Page
  + Components on page that read user actions

**Controller**

* + JavaScript event handlers (script.js and scripts built in to html files)
  + AJAX calls to load JSON data
  + Logic to update views based on user actions

Ex. code from generator.html

Controller Logic

A screen shot of a computer code

AI-generated content may be incorrect.

Fetch gets data from model

Update View

**Hierarchy**

**Overall Website Hierarchy**

* Index Page
* Dashboard
* Generator
* Analyzer
* Code Reviewer

**General Page Hierarchy**

* Navbar
  + Home
  + Features
  + Pricing
  + Contact
  + Signup/Login
* Sidebar
  + Dashboard
  + Generator
  + Analyze
  + Code Review
  + Settings
  + Logout
  + Profile
* Main Content
  + Varies

**Generator Hierarchy**

* Navbar (Above)
* Sidebar (Above)
* Main Content
  + Header
  + AI Query Container
    - Text Field
    - Generate Button
    - Random Button
    - Language Choice
  + Generated Code Container
    - Text Field
  + Live Preview Container
    - Text Field
* Component Sidebar
  + Login Form, Registration Form, etc. components

**Analyze Hierarchy**

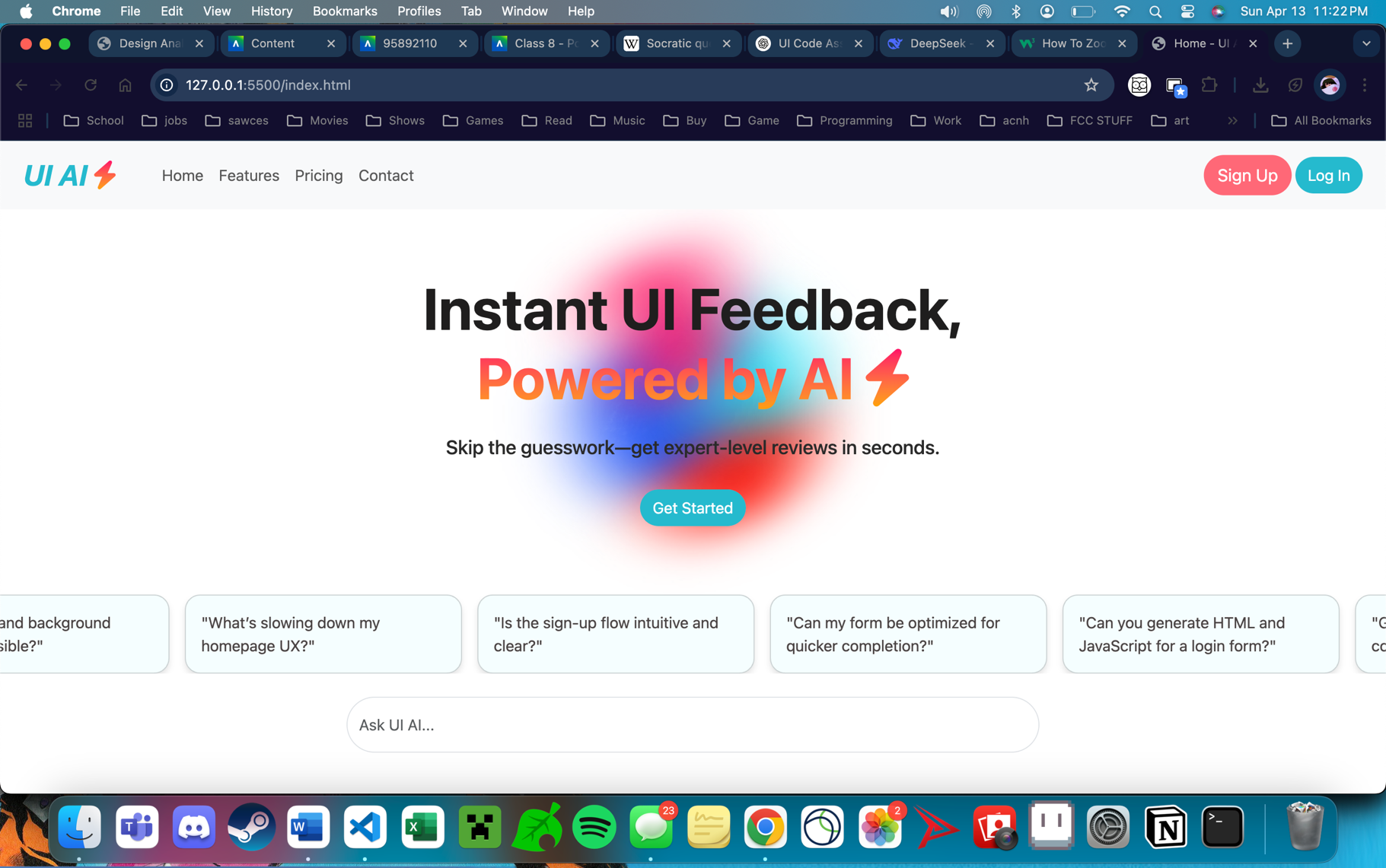
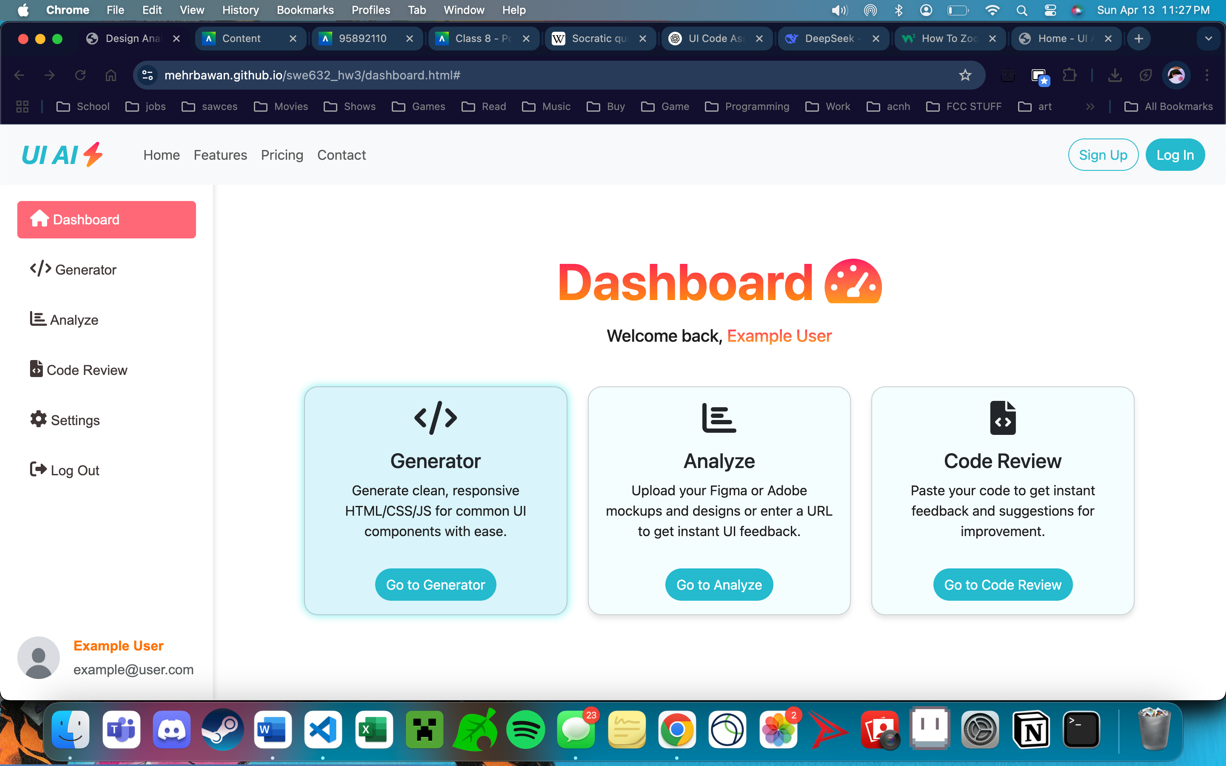
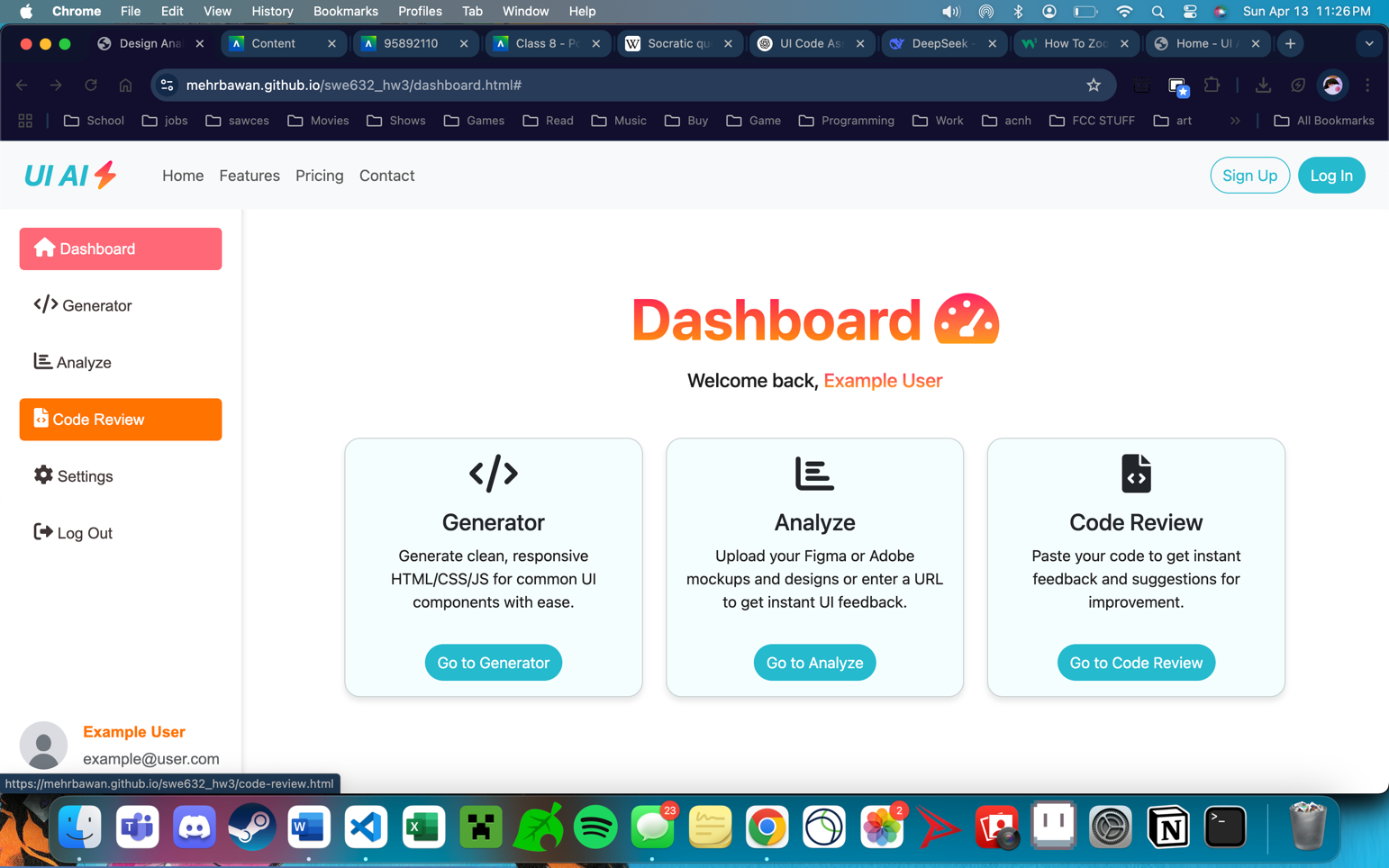
* Navbar (Above)
* Sidebar (Above)
* Main Content
  + Header
  + Upload Container
    - Choose File Button
    - Text Instructions
  + Live Web Upload
    - Website Link
    - Analyze Button
  + AI Report Container
    - Text Field
    - Regenerate Button
    - Download Button

**Code Review Hierarchy**

* Navbar (Above)
* Sidebar (Above)
* Main Content
  + Header
  + Code Container
    - Paste Text
    - Text Field
    - Submit Button
  + AI Feedback Container
    - Text Field
    - Regenerate Button
    - Download Button

**Pointing**

**Pointing Features**

* Button zoom on small buttons for easier clicking
  + 
  + (mouse pointer not visible in my computer’s screenshots, but pointer is on signup button)
* Buttons are spaced Far apart enough to avoid misclicks, and button borders are highlighted if not already explicit to avoid targeting errors.
  + ****