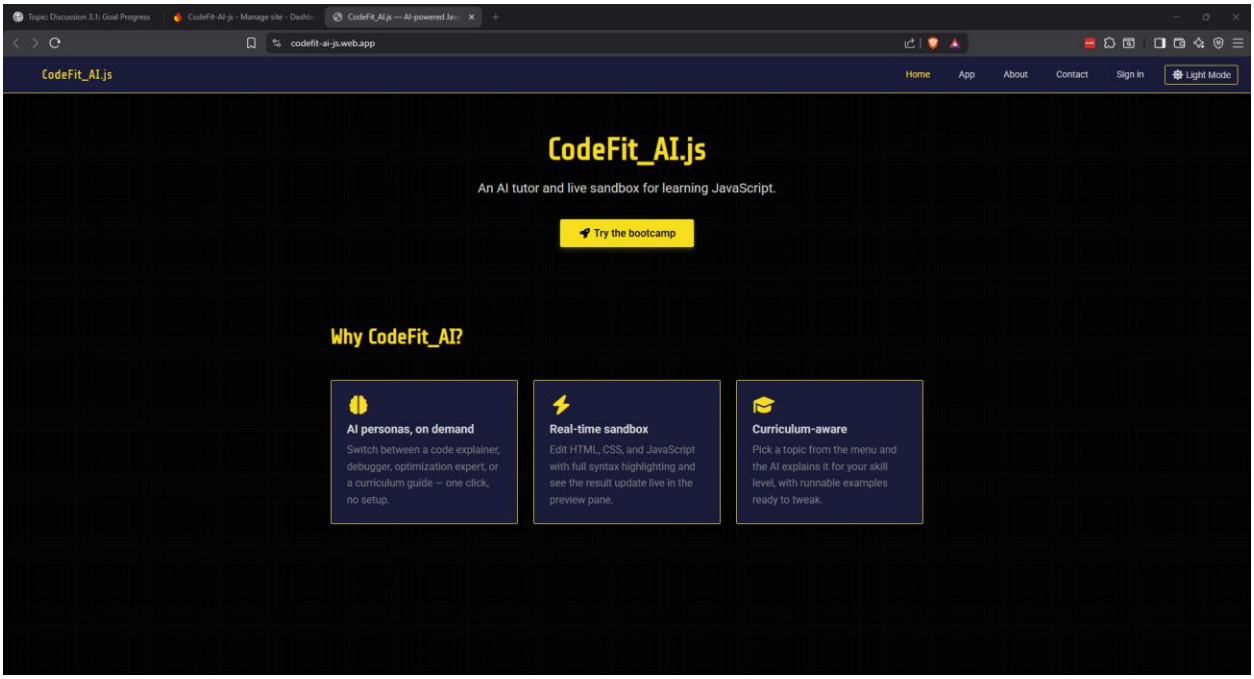
CodeFit_AI.js

Fernando Trejo

AUTHOR(S):
Fernando Trejo



Technical Field

This project involves the technical fields of Advancing Computer Science and Artificial Intelligence.

Background Information

Computer Science has the highest dropout rate of any major in the US - about 10.7%, against a roughly 40% overall yearly dropout rate (https://lanterncredit.com/student-loans/computer-science-dropout-rate). Coding bootcamps see a similar pattern, with graduation rates ranging from 69% to 89% depending on the program (https://careerkarma.com/blog/online-coding-bootcamps-rates-and-outcomes/).

The main reason is simple: most people considering a coding career don't really know what the job involves day-to-day, and they don't know whether they're a good fit for it. They commit time and money before finding out - which is expensive to undo.

This affects three groups of people:

- Aspiring software engineers can spend years and a lot of money chasing a career that turns out not to fit them.
- Educators and mentors put their time into students who may not be suited for the field.
- The tech industry ends up flooded with new applicants who don't have the skills the jobs require.

Prior Art

A few existing platforms cover pieces of what CodeFit_AI.js does, but none of them combine all of it.

Codewars - A community for practicing coding challenges (what the tech industry calls "problem solving"). Great for sharpening skills you already have, but it doesn't teach the fundamentals from scratch.

- Link: https://www.codewars.com/
- Similar platforms: LeetCode, HackerRank.

Codecademy - An online platform offering free coding classes. Solid for structured lessons, but the feedback you get is pre-written, not personalized to your code.

- Link: https://www.codecademy.com/
- Similar platforms: FreeCodeCamp, Coursera, Udemy, Pluralsight.

Google's Grasshopper - A mobile and desktop app that taught beginners to code in JavaScript. A professor pointed me to it; it appears to have been shut down and is no longer available on either platform.

- Link: https://blog.google/company-news/outreach-and-initiatives/grow-with-google/grasshopper-desktop-learn-to-code/

Project Description

CodeFit_AI.js is a web app that helps people figure out whether software engineering is the right career for them - before they spend years and a lot of money finding out the hard way.

The app works in three steps:

1. Assess: The app will evaluate the user's aptitude and personality and tell them whether they fit the profile of a software engineer.
2. Redirect or teach: If the user isn't a strong fit for software engineering, the app recommends other IT careers that match their results. If they are a fit, they start learning the fundamentals of web development and JavaScript through an interactive coding platform with an AI mentor.
3. Reward and guide: Users who don't finish the fundamentals get reminders to come back. Users who do finish earn a certificate of completion and a personalized roadmap toward becoming a full-stack, frontend, or backend developer - plus exclusive offers from partnered coding bootcamps and universities.

The user interacts with AI throughout the app, but most heavily during the coding lessons. The goal is to give aspiring software engineers an honest, unbiased read on whether coding is the right fit for them.

Innovation Claim

What makes CodeFit_AI.js different is that it combines three things no existing platform puts in one place: a career-fit assessment, AI-guided coding lessons, and a personalized roadmap based on the user's results.

Most existing platforms do one of these well. Codewars and LeetCode help you practice. Codecademy and FreeCodeCamp teach the fundamentals. None of them ask the most important question first: is this career a good fit for you in the first place?

CodeFit_AI.js answers that question up front using aptitude and personality assessments. From there, users who fit the profile learn the basics of web development and JavaScript with an AI mentor that gives feedback on their actual code - not pre-written hints. When they finish, they walk away with a roadmap tailored to their goals (frontend, backend, or full stack).

The result: aspiring developers find out whether coding suits them before committing time and money to a bootcamp or degree, and the ones who do commit have a clearer head start.

Usage Scenario

The core idea behind CodeFit_AI.js - assessing fitness, then guided learning - isn't limited to aspiring software engineers. The same approach can help anyone considering a career change, especially in the tech industry.

For example, a marketing professional thinking about moving into tech could use the app to see whether they're a better match for data analysis, digital marketing, or IT project management. From there, the app can point them to learning resources for that path, with the same AI-driven mentorship and progress tracking. That broadens who the app helps and addresses a real need: most career guidance today is generic, not personal.

The app is built for four main groups of users:

- Aspiring software engineers: Usually high school or college students exploring whether software engineering is the right career for them.
- Career changers: Professionals from other fields who want to move into software engineering or another IT role, and need a tailored recommendation based on their existing skills.
- Lifelong learners: People who already have a career but want to pick up new skills, either for personal growth or to stay competitive.

- Schools, universities, and coding bootcamps: Institutions that want to offer their students a career-fit and skill-building tool to help them make smarter decisions about their path.

Evaluation Criteria

The following questions will be used to judge whether the project succeeded.

1. Assessment & Recommendations

- Does the app successfully assess users' aptitude for software engineering?
- Does the assessment feel unbiased and trustworthy to users?
- If a user isn't a fit, does the app give them clear, personalized recommendations for other IT fields?
- Do users find the results and recommendations accurate and valuable?

2. Learning Experience

- Does the app effectively guide users through the fundamentals of web development?
- Are the coding lessons engaging and educational?
- Does the AI mentor give helpful, accurate assistance during lessons?
- Do users feel motivated to finish the lessons and assessments?

3. Platform & User Experience

- Does the interactive coding platform run smoothly, without major technical issues?
- Is the user interface intuitive and easy to use?
- Is there a clear onboarding process for new users?
- Can the app handle many users at the same time without slowing down?

4. Completion & Rewards

- Does the app remind users who haven't finished the fundamentals to come back?
- Does the app track user progress accurately and give meaningful feedback?
- Are certificates issued automatically when users finish the fundamentals?
- Does the app give finishers a clear, detailed career roadmap?
- Are finishers offered exclusive opportunities to join partnered bootcamps or universities?

5. Feedback, Privacy & Maintenance

- Is there a way for users to give feedback on their experience?
- Is the app regularly updated and improved based on that feedback?
- Does the app meet relevant data privacy and security standards?

Objectives and Tasks Associated with the Project

High-Level Objectives:

1. Assess user aptitude and personality
2. Provide personalized career recommendations
3. Teach the fundamentals of web development
4. Build and maintain an interactive coding platform with an AI mentor
5. Issue certificates and provide career roadmaps
6. Partner with coding bootcamps and universities
7. Build a user-friendly, intuitive interface
8. Keep the app fast and scalable
9. Collect user feedback and use it to improve the app
10. Comply with data privacy and security standards

Tasks for each objective:

1. Assess user aptitude and personality

- Develop the assessment framework
    - Task: Build a framework for measuring user aptitude and personality.
    - Status: Not started
    - Decision-Making Transparency: Base the questions and scoring on industry standards and existing psychological research.
- Build the assessment tool
    - Task: Design the online assessment and integrate it into the web app.
    - Status: Not started
    - Decision-Making Transparency: Work with a UX designer to keep the flow smooth.
- Validate the results
    - Task: Pilot-test with a small group to confirm the assessment is accurate and reliable. May be unnecessary if using third-party services.
    - Status: Not started
    - Decision-Making Transparency: Review pilot results and refine the tool based on feedback and data.

2. Provide personalized career recommendations

- Build the recommendation algorithm
    - Task: Create an algorithm that suggests careers based on the user's assessment results. May be unnecessary if the assessments come from a third party.
    - Status: Not started

    - Decision-Making Transparency: Use machine learning models and consult with career-development experts.
- Add recommendations to the app
    - Task: Wire the recommendation logic into the app's user flow.
    - Status: Not started
    - Decision-Making Transparency: Make sure the recommendations are clear, useful, and easy to act on.

3. Teach the fundamentals of web development

- Design the curriculum
    - Task: Outline what the web development fundamentals course will cover.
    - Status: In progress (curriculum menu with HTML/CSS/JavaScript topics exists)
    - Decision-Making Transparency: Consult experienced educators and industry professionals where helpful - though I can write the bulk of this myself.
- Build the lesson modules
    - Task: Create interactive lessons covering HTML, CSS, and JavaScript.
    - Status: In progress (topic-scoped AI chat answers per topic; formal lessons not yet built)
    - Decision-Making Transparency: Follow instructional design principles so the lessons teach effectively.
- Add progress tracking
    - Task: Track which lessons each user has completed and where they left off.
    - Status: Not started (chat history persists per topic; lesson-completion tracking not yet built)
    - Decision-Making Transparency: Make sure tracking is accurate and the feedback it surfaces is meaningful to the user.

4. Build and maintain an interactive coding platform with an AI mentor

- Build the AI mentor
    - Task: Create an AI assistant that helps users during coding lessons.
    - Status: Complete (4 roles: Code Explainer, Debugger, Optimization Expert, Curriculum Explainer; powered by OpenAI API with streaming)
    - Decision-Making Transparency: Use natural language processing and machine learning to handle user questions.
- Integrate the mentor into the platform
    - Task: Connect the AI mentor to the interactive coding environment so users can ask for help without leaving the lesson.
    - Status: Complete (chat alongside the 3-pane code sandbox; AI sees the active lesson and sandbox code; an optional voice mode lets users ask by speaking and hear the reply read back).

    - Decision-Making Transparency: Keep the interaction intuitive and unobtrusive.
- Test and improve AI performance
    - Task: Continuously test the mentor and tune it over time.
    - Status: Ongoing
    - Decision-Making Transparency: Use user feedback and performance metrics to guide improvements.

5. Issue certificates and provide career roadmaps

- Design the certificate templates
    - Task: Create professional certificate templates for users who complete the fundamentals.
    - Status: Not started
    - Decision-Making Transparency: Make sure the certificates look credible and feel meaningful.
- Build the career roadmaps
    - Task: Create detailed roadmaps users can follow after the fundamentals, based on their career goals.
    - Status: Not started
    - Decision-Making Transparency: Consult industry experts to keep the roadmaps realistic and actionable - don't rely on my expertise alone here.

6. Partner with coding bootcamps and universities

- Identify potential partners
    - Task: Research and reach out to coding bootcamps and universities that could partner with the app.
    - Status: Not started
    - Decision-Making Transparency: Pick partners based on their reputation, course offerings, and how well they line up with the project's goals.
- Establish partnership agreements
    - Task: Negotiate and formalize the partnership terms.
    - Status: Not started
    - Decision-Making Transparency: Make sure the agreements work for both sides and that the terms are spelled out clearly.

7. Build a user-friendly, intuitive interface

- Conduct user research
    - Task: Talk to potential users to understand what they need and prefer. May not be necessary - see UI/UX note below.
    - Status: Not started

    - Decision-Making Transparency: Use surveys, interviews, and usability testing.
- Design the UI/UX
    - Task: Design the interface based on what the research surfaces. I already have an experienced UI/UX designer, so this may largely be handled.
    - Status: In progress (live UI exists with site-wide light/dark theme, off-canvas navbar, profile, snippets list, and a mobile-responsive layout that becomes a chat-only tutor on phones).
    - Decision-Making Transparency: Iterate on the designs based on user feedback and testing.

8. Keep the app fast and scalable

- Set up performance monitoring
    - Task: Add tools that track how the app is performing. Some of this may come built into the web hosting service.
    - Status: Complete (built into Firebase Hosting and Cloud Functions)
    - Decision-Making Transparency: Use real-time data to spot and fix performance issues as they happen.
- Optimize the code and infrastructure
    - Task: Keep tuning the code and infrastructure so the app handles more users as it grows. This will be ongoing.
    - Status: Ongoing (Firebase auto-scales; a per-user rate limit guards against runaway usage and cost; tuning continues as features ship).
    - Decision-Making Transparency: Follow industry best practices for performance.

9. Collect user feedback and use it to improve the app

- Create feedback channels
    - Task: Give users easy ways to share their experience with the app.
    - Status: Not started
    - Decision-Making Transparency: Make the channels easy to find and easy to use.
- Review feedback and ship improvements
    - Task: Regularly review what users say and make the changes that matter most.
    - Status: Not started
    - Decision-Making Transparency: Prioritize changes based on impact to users and how feasible they are to build.

10. Comply with data privacy and security standards

- Run a security audit
    - Task: Run a full security audit to find vulnerabilities. Some of this may be handled by the hosting service, but an audit is still worth doing.

    - Status: Not started (Firebase App Check noted as optional follow-up before public launch)
    - Decision-Making Transparency: Bring in third-party experts so the assessment is unbiased.
- Implement security measures
    - Task: Apply the fixes the audit recommends.
    - Status: In progress (Firebase Auth gates the app; Firestore rules restrict every user to their own data; secrets in Secret Manager; security response headers - HSTS plus clickjacking and content-type protections - sent on every page; a per-user rate limit on the backend functions.
    - Decision-Making Transparency: Follow best practices and any legal requirements around data privacy and security.

Description of Design Prototype

The prototype is a working web app that runs entirely on Firebase, Google's app-hosting platform. It's split into two parts: a client (the website the user sees in their browser) and a backend (the part that talks to the AI and stores user data). The app is live in production, and the same code can be run locally for development.

What's in the prototype today:

- A landing page introducing CodeFit_AI.js, with links to the app, about, and contact pages.
- A sign-in page where users can create an account with email and password or sign in with Google. Until they sign in, the app itself is hidden.
- The main app page, which combines the AI tutor and the code sandbox side by side:
    - An AI chat assistant the user can talk to. The user picks one of four roles for the AI - Code Explainer, Debugger, Optimization Expert, or Curriculum Explainer - which changes how the AI responds. The AI is also aware of the lesson the user is on and the code they've written. The user can also turn on voice mode and hold a spoken, back-and-forth conversation with the tutor - speak a question into the mic and hear the reply read back - with typed chat still available at any time.
    - An in-browser code sandbox with three editor panels (HTML, CSS, and JavaScript), a live preview window that updates as the user types, and a "Run" button that executes their JavaScript and shows the output.
- A Curriculum menu the user can browse to pick a web development topic. Selecting one opens that topic's conversation with the AI and resumes the conversation if the user has visited that topic before. Each topic's chat history can also be cleared with one click to start the topic over fresh.

- A Snippets menu where users can save their sandbox code under a name, reload it later, rename it, or delete it. Snippets are saved to the user's account and follow them across devices.
- A user profile in the navbar showing the user's name and avatar. The user's preferred AI role and light/dark theme are saved to their account so they're the same on any device.
- The whole site works on phones, not just desktop. On a small screen the layout adapts: the top navigation collapses into a menu, and the app page becomes a chat-only tutor - the code sandbox is a desktop feature, so on a phone the user still gets the AI tutor (including voice) and the curriculum, without the editor.

How it's built (in plain terms):

- The client is built with Vite (a modern build tool for fast development), vanilla JavaScript, and CodeMirror 6 (the same code-editor library used inside many developer tools). It runs as a multi-page site - landing, app, about, contact, and sign-in.
- The backend is a set of Firebase Cloud Functions (Google's serverless backend platform) that talk to OpenAI's API. One function generates the AI tutor's responses, streaming them back word-by-word the same way ChatGPT does, so they feel quick; two more power voice mode by turning the user's recorded question into text and the tutor's reply back into spoken audio. A per-user rate limit on these functions keeps automated abuse and runaway costs in check.
- Firebase Authentication handles sign-in and account creation (email/password and Google).
- Firestore, Google's cloud database, stores each user's chat conversations, code snippets, and preferences. Users can only ever read or write their own data, enforced by security rules on the database.
- The AI's instructions for each role live in the backend, so the AI's personalities can be tuned without redeploying the website.

How to run the prototype locally:

1. Install Node.js (version 24, which the Cloud Functions backend targets) and a Java Development Kit (required by the Firestore emulator).
2. Install the Firebase CLI with npm install -g firebase-tools.
3. From the project's root folder, run npm install once to install everything.
4. Add the OPENAI_API_KEY to a file at functions/.secret.local (the AI features won't work without it).
5. Run npm run dev. The client opens on http://localhost:8080, and the Firebase emulators run the backend, authentication, and database locally on ports 5001, 9099, and 8085.

How the prototype relates to the final project: This prototype demonstrates the learning half of the larger vision - the AI tutor, the interactive coding sandbox, accounts, and persistent progress

across devices. The career-fit assessment, certificates, roadmaps, and bootcamp partnerships described earlier in this brief will be added in later iterations. The app being live in production means real users can sign up and use it today.

Evaluation Plan

The evaluation will combine three approaches: hands-on testing of the prototype by the author, structured walk-throughs with a small group of test users, and ongoing feedback collection from real users now that the app is live in production.

The assessment and recommendations parts will be evaluated by running through the flow end-to-end and checking whether the results surface clearly in the app and whether the career suggestions that follow are easy to act on. Because the aptitude and personality assessments will most likely come from third-party services, the focus here is on how well those results are presented and explained - not on the accuracy of the assessments themselves, which is the third party's responsibility.

The learning experience will be evaluated through hands-on use. The author and a small test group will work through the fundamentals from start to finish, ask the AI mentor questions, and track whether the lessons teach the material and whether the mentor's answers stay accurate, on-topic, and helpful. Test sessions will be informal conversations rather than formal usability studies, since the focus at this stage is on substance, not polish.

The platform and user experience will be evaluated in two passes. The first is functional: does each feature work without crashing, does the page load fast, and is the layout obvious to someone seeing the app for the first time? This pass also covers the experience across devices - the responsive layout on phones (where the app runs as a chat-only tutor) and voice mode, checked end-to-end from spoken question to spoken reply. The second is load-related: now that the app is deployed, performance is monitored through Firebase's built-in tools to confirm it stays responsive as more users sign up. Firebase auto-scales the backend automatically, so most of the scaling concern is handled by the hosting platform itself.

The completion and rewards group is the most straightforward to check. Users who finish the fundamentals should receive a certificate, see a roadmap tailored to their goals, and be offered any available bootcamp or university partnerships. Each of those is a binary check, so evaluation is mostly a matter of verifying each one fires at the right time.

Finally, the feedback, privacy, and maintenance group will be evaluated by making sure there's a working feedback channel from day one, reviewing what users send in on a regular schedule, and shipping updates that reflect that feedback. Privacy and security baseline is already in place: Firebase Authentication handles all sign-in, and Firestore security rules restrict every user to their own data - they can't see or change anyone else's. The site also sends standard security response headers (HSTS plus clickjacking and content-type protections) on every page, and the

backend functions enforce a per-user rate limit to curb abuse. A one-time third-party audit and Firebase App Check (an extra layer of abuse prevention) are planned before a wider public launch.

Project Completion Assessment

What went right: The foundation came together cleanly, and the project reached production. Three rounds of refactoring shaped the prototype into a clear structure, and a fourth phase migrated the entire backend to Firebase, Google's app-hosting platform. The app is now live, with real user accounts (email/password and Google sign-in), persistent chat history, code snippet save/load, and per-user preferences that follow users across devices. The AI chat assistant works the way it's meant to - the user picks a role for the AI, types a question, and gets a streamed reply in real time, the same way ChatGPT does. The in-browser code sandbox is functional too: three editors for HTML, CSS, and JavaScript, a live preview that updates as the user types, and a console panel that captures the output when they run their code. Switching between light and dark themes works both site-wide and inside the editor, and the user's choice is saved to their account. Two later additions rounded out the experience: an optional voice mode that lets users hold a spoken back-and-forth with the tutor, and a mobile-responsive layout so the app works on phones - where it runs as a chat-only tutor, since the code sandbox stays a desktop feature.

What went wrong: The original scope was too ambitious for an MVP. The first version of this brief described a complete system covering aptitude assessments, personalized career recommendations, certificates, roadmaps, and bootcamp partnerships - none of which fit into one semester. Trimming the prototype down to its core (the AI tutor and the coding sandbox) took longer than it should have. The Firebase migration brought its own bumps, especially during the first deployment: the OpenAI API key got silently corrupted by an invisible character when it was set through the PowerShell terminal, which broke every AI call until it was tracked down; the chat replies arrived all at once instead of streaming because Firebase Hosting was buffering them at its gateway, which forced a change to call the backend function directly; and a few smaller issues - a failed release leaving the site temporarily unreachable, an environment variable that wasn't being picked up at build time - added their own delays.

What I learned: A few things. First, an MVP needs to do one thing well before it tries to do everything - the most useful feedback I got early on was from professors pointing out that the original brief was covering too much ground. Second, getting the underlying tooling right early (workspaces, linting, formatting, clean module structure) paid off later. Every change since has been easier because the foundation was already clean. Third, production reveals issues that local testing never will. The Firebase emulators were useful for day-to-day development, but several of the deployment issues only showed up against the real Firebase infrastructure - things like the

hosting gateway buffering the streaming response, or how the terminal handled the secret. Hitting them sooner would have been less stressful than hitting them all at the end.

What I would do differently: I would have started with a much narrower scope from day one - just the AI chat assistant alongside the coding sandbox - and let the other features come in later iterations (this is the current direction given the time constraints). I would also have deployed to production earlier, even a barebones starter version, so the infrastructure-level surprises surfaced one at a time during development instead of all at once near the deadline. Getting rough versions in front of real users earlier is the same idea: a rough live app with real users gives you better signal than a polished local app with none.

Appendices

Appendix A: Project README and setup instructions – README.md

Appendix B: Live deployment of CodeFit_AI.js – https://codefit-ai-js.web.app

Appendix C: Screenshot of the landing page – landing-page.png (TBD)

Appendix D: Screenshot of the sign-in page – sign-in-page.png (TBD)

Appendix E: Screenshot of the main app page, showing the AI chat and code sandbox side by side – app-page.png (TBD)

Appendix F: Screenshot of the app in dark mode – app-dark-mode.png (TBD)

Appendix G: Screenshot of the AI Roles menu with the four roles – ai-roles-menu.png (TBD)

Appendix H: Screenshot of the Curriculum menu – curriculum-menu.png (TBD)

Appendix I: Screenshot of the Snippets menu in the navbar – snippets-menu.png (TBD)

Appendix J: Screenshot of the user profile area in the navbar – user-profile.png (TBD)

Appendix K: Screenshot of a returning user with persistent chat history loaded – chat-history.png (TBD)

Appendix L: References

What Is the Highest Dropout Rate by Major? (n.d.) Lantern by SoFi. Retrieved from https://lanterncredit.com/student-loans/computer-science-dropout-rate

Coding Bootcamp Graduation Rates and Outcomes (n.d.) Career Karma. Retrieved from https://careerkarma.com/blog/online-coding-bootcamps-rates-and-outcomes/

Codewars. Retrieved from https://www.codewars.com/

Codecademy. Retrieved from https://www.codecademy.com/

Learn to code with Grasshopper, now on desktop (n.d.) Google Blog. Retrieved from https://blog.google/company-news/outreach-and-initiatives/grow-with-google/grasshopper-desktop-learn-to-code/