Full-Stack Development and Debugging

Category	Workshop
Content Type	Workshop
Status	Not Started
Workshop Day	Day 2

Workshop Overview

Purpose:

Teach participants how to turn a locally running frontend into a functional application with authentication, persistent data, and user-specific behavior using Firebase.

Participants will also practice debugging full-stack problems using browser Developer Tools, application errors, Firebase information, and AI engineering assistants.

The workshop introduces hooks after participants have gained enough experience with Codex and Claude Code to recognize repetitive development tasks that may benefit from automation.

Duration: 60 to 120 minutes

Instructor(s): TBD

Workshop Day: Day 2

Learning Objectives

By the end of this workshop, participants should be able to:

- Explain the difference between frontend behavior, authentication, and persistent data
- Create and configure a Firebase project
- Connect Firebase to a web application

- Implement user sign-up, sign-in, and sign-out
- Use Firestore to store and retrieve persistent application data
- Associate application data with authenticated users
- Understand basic Firebase security considerations
- Use browser Developer Tools to investigate application problems
- Distinguish between symptoms, errors, and root causes
- Give Codex and Claude Code useful debugging context
- Test AI-generated fixes before accepting them
- Identify repetitive development tasks that may benefit from hooks
- Use hooks selectively to improve an AI-assisted development workflow

---

Prerequisites

Participants should have completed Day 1 or be able to:

- Run a web application locally
- Navigate a project in Visual Studio Code
- Use the integrated terminal
- Install dependencies using npm
- Use Codex or Claude Code for basic development tasks
- Use basic Git and GitHub workflows
- Commit and push project changes
- Use a browser and locate Developer Tools

Participants should bring the project they worked on during Day 1 when possible.

Instructor Preparation

- [ ] Prepare a Firebase project for the live demonstration
- [ ] Prepare a small application that currently stores data only locally or temporarily

- [ ] Verify Firebase Authentication works with the demonstration project
- [ ] Verify Firestore reads and writes work correctly
- [ ] Prepare an example authenticated user flow
- [ ] Prepare an example of user-specific Firestore data
- [ ] Prepare at least one intentional or reproducible Firebase-related bug
- [ ] Prepare at least one browser console or network debugging example
- [ ] Test the Firebase configuration process from a clean project
- [ ] Prepare examples of useful and unnecessary hooks
- [ ] Verify Codex and Claude Code can access the demonstration project
- [ ] Prepare Firebase setup instructions for participants

Required Accounts

- Firebase / Google account
- GitHub
- Claude
- OpenAI / Codex

Required Software

- Visual Studio Code
- Git
- Node.js
- npm
- Modern web browser
- Claude Code
- Codex

Required Files / Repositories

- Day 1 project or starter repository
- Firebase setup guide

- Example Firebase configuration
- Debugging exercise
- AI debugging prompt examples

Workshop Outline

Time	Section	Format	Notes
0 to 10 min	From Website to Application	Teach	Explain authentication, persistence, and application data
10 to 25 min	Firebase Setup	Teach / Demo / Exercise	Create Firebase project and connect application
25 to 40 min	Authentication	Demo / Exercise	Sign-up, sign-in, sign-out, auth state
40 to 55 min	Firestore and Persistent Data	Demo / Exercise	Store and retrieve application data
55 to 70 min	User-Specific Data	Teach / Demo	Associate records with authenticated users
70 to 85 min	Debugging Full-Stack Problems	Demo / Exercise	Console, Network, Firebase, AI-assisted investigation
85 to 95 min	Hooks and Workflow Automation	Teach / Demo	Identify repetitive tasks worth automating
95 to 120 min	Build and Instructor Support	Exercise	Participants continue implementing and debugging

For a 60-minute session, prioritize Firebase setup, authentication, Firestore, and debugging. Hooks can be moved to supplemental material if necessary.

Concepts to Teach

Concept 1: Authentication vs. Application Data

What participants need to understand:

Authentication answers:

"Who is this user?"

The database answers:

"What information should the application remember?"

These are related but separate responsibilities.

Key points:

- Authentication establishes user identity
- Users have unique identifiers
- Firestore stores application data
- Refreshing the browser should not erase persistent data
- Application behavior can depend on authentication state
- Data can be associated with a specific authenticated user

Concept 2: Firebase Authentication

What participants need to understand:

Firebase Authentication allows the application to establish and maintain user identity.

Key points:

- Sign-up
- Sign-in
- Sign-out
- Authentication state
- User ID
- Authentication errors
- Protected functionality
- Do not store passwords manually in Firestore

Concept 3: Persistent Data with Firestore

What participants need to understand:

Firestore allows application information to remain available beyond the current browser session.

Key points:

- Collections
- Documents
- Fields
- Creating data
- Reading data
- Updating data
- Deleting data
- Document IDs
- User ownership

Example mental model:

User → Interface → Application Logic → Firebase → Firestore → Stored Data

Concept 4: User-Specific Data

What participants need to understand:

Authentication and data become particularly useful when the application knows which information belongs to which user.

A user's unique Firebase ID can be used to associate records with that user.

Key points:

- Authenticated user ID
- Data ownership
- Querying the correct records
- Avoid exposing another user's information
- Authentication alone does not automatically secure database records

Concept 5: Debugging with Evidence

What participants need to understand:

When something fails, immediately asking AI to "fix it" is usually less effective than first collecting evidence.

Use the debugging workflow:

Expected → Observed → Evidence → Hypothesis → Fix → Test → Verify

Key points:

- Read the actual error
- Check the browser console
- Inspect network activity when relevant
- Check terminal output
- Check Firebase configuration
- Determine whether the problem is frontend, authentication, database, configuration, or permissions
- Give AI the evidence you collected
- Test proposed fixes

Concept 6: Hooks and Workflow Automation

What participants need to understand:

After using AI coding assistants repeatedly, participants may notice recurring tasks or checks.

Hooks can automate selected repetitive actions in the development workflow.

The objective is not to create as many hooks as possible.

The objective is:

Use AI → Observe workflow → Identify repetition → Automate useful repetition

Key points:

- Understand the task before automating it
- Identify repetitive actions
- Prioritize checks that prevent mistakes
- Ask AI to recommend hooks based on the actual workflow
- Understand what a hook does before enabling it
- Keep automation simple and useful
- Do not automate processes participants do not understand

Live Demonstration

Objective

Transform a simple local application into an application with authentication and persistent user-specific data.

Then demonstrate how to investigate a problem using evidence rather than blindly asking AI to fix it.

Steps

1. Open the Day 1 demonstration project.
2. Show that the current application has no persistent user identity or cloud data.
3. Create or open a Firebase project.
4. Register the web application with Firebase.
5. Add Firebase configuration to the project.
6. Enable an authentication provider.
7. Implement sign-up.
8. Implement sign-in and sign-out.
9. Display behavior based on authentication state.
10. Create a Firestore database.
11. Store a piece of application data.
12. Refresh the application and demonstrate that the data persists.
13. Associate the data with the authenticated user's ID.
14. Retrieve only the appropriate user's data.
15. Introduce or encounter a problem.
16. Inspect the browser console, network activity, or Firebase information.
17. Give the relevant evidence to Codex or Claude Code.
18. Evaluate the proposed fix.
19. Apply the change.
20. Test the application again.

21. Verify the expected behavior.
22. Commit the working implementation to GitHub.

Expected Result

Participants should see a progression from:

Local Website → Authentication → Persistent Data → User-Specific Application

They should also see that debugging involves collecting evidence and understanding the problem before changing code.

Hands-On Exercise

Objective

Participants will add authentication and persistent data to their practice application and troubleshoot problems using the debugging workflow.

Instructions

1. Open the project from Day 1.
2. Verify that it runs locally.
3. Create or configure a Firebase project.
4. Connect Firebase to the application.
5. Enable the required authentication method.
6. Implement user registration.
7. Implement user sign-in.
8. Implement sign-out.
9. Display appropriate behavior based on authentication state.
10. Create a Firestore database.
11. Choose one piece of application information that should persist.
12. Save that information to Firestore.
13. Retrieve the information from Firestore.
14. Associate the information with the authenticated user.
15. Refresh the application and verify persistence.

16. Sign out and sign in again.
17. Verify that the correct data remains available.
18. Investigate any errors using Developer Tools and other available evidence.
19. Use Codex or Claude Code to assist with investigation when necessary.
20. Test any proposed fix.
21. Commit the working checkpoint to GitHub.

Success Criteria

Participants have successfully completed the exercise when:

- [ ] A user can create an account
- [ ] A user can sign in
- [ ] A user can sign out
- [ ] Authentication state affects application behavior
- [ ] At least one type of application data is stored in Firestore
- [ ] Stored data remains after refreshing the page
- [ ] Data is associated with the appropriate authenticated user
- [ ] Participants can locate browser errors using Developer Tools
- [ ] Participants can describe how they would investigate a failure
- [ ] The working implementation has been committed to GitHub

---

AI Prompts

Prompt 1: Firebase Implementation Planning

> I want to add Firebase to this application.
>
> I need:
>
> - User authentication
> - Persistent application data
> - Data associated with the authenticated user

> Before modifying anything, inspect the existing project and explain how Firebase should integrate with the current structure. Identify the files that will likely need to change, any dependencies that are required, and any configuration or environment variables that will be needed.
>
> Do not implement anything yet.

Purpose:

Teach participants to ask AI to understand the existing application and plan an integration before making large changes.

Prompt 2: Debugging with Evidence

> I am debugging the following problem.
>
> Expected behavior:
>
> [EXPECTED BEHAVIOR]
>
> Actual behavior:
>
> [ACTUAL BEHAVIOR]
>
> Browser console:
>
> [ERROR OR OUTPUT]
>
> Network information:
>
> [RELEVANT INFORMATION]
>
> Other relevant context:
>
> [CONTEXT]
>
> Investigate the likely root cause before changing code. Explain what evidence supports your diagnosis. Then recommend the smallest appropriate fix and explain how I should verify it.

Purpose:

Teach participants to give AI evidence rather than vague instructions.

Prompt 3: Discover Useful Hooks

> Based on how I have been using Claude Code in this project, identify repetitive actions or checks that could be automated with hooks.

> Suggest only hooks that would improve productivity, code quality, testing, or debugging.
>
> For each suggestion, explain:
>
> 1. What triggers the hook
> 2. What the hook would do
> 3. Why it would be useful
> 4. Any disadvantages or risks
>
> Do not implement anything yet.

Purpose:

Teach participants to analyze their workflow before introducing automation.

Prompt 4: Prioritize Hooks

> From the hooks you recommended, select the three highest-value options for this project.
>
> Prioritize hooks that prevent mistakes or eliminate repetitive manual work.
>
> Explain what each hook would do before implementing anything.

Purpose:

Prevent participants from adding unnecessary automation simply because AI suggested it.

Common Problems

Problem 1: Firebase Configuration Errors

Symptoms:

Authentication or database functionality fails immediately after Firebase is added.

Likely Cause:

Incorrect Firebase configuration, missing dependencies, incorrect initialization, or environment configuration.

How to Investigate:

Check:

- Browser console
- Firebase configuration
- Installed dependencies
- Firebase initialization
- Environment variables
- Development server output

Instructor Hint:

Ask:

"Which part is failing first, connecting to Firebase or using the Firebase service?"

Solution:

Identify the failing layer before changing implementation code.

Problem 2: Authentication Works but Data Does Not

Symptoms: The user can sign in successfully, but Firestore reads or writes fail.

Likely Cause: Authentication and Firestore are separate services. Possible causes include Firestore configuration, database rules, incorrect paths, incorrect queries, or application logic.

How to Investigate: Verify authentication independently, then investigate the Firestore request and related errors.

Instructor Hint:

Ask:

"What evidence tells us authentication is the problem?"

This encourages participants to avoid blaming the wrong system.

Solution: Isolate authentication from database behavior and investigate the failing operation.

Problem 3: Users Can See Incorrect Data

Symptoms: A user sees data created by another account.

Likely Cause: Data is not properly associated with or filtered by the authenticated user's ID.

How to Investigate: Inspect stored Firestore documents and application queries.

Instructor Hint:

Ask:

"How does the application currently know which records belong to this user?"

Solution: Establish clear ownership using the authenticated user's identifier and appropriate database access rules.

Problem 4: AI Keeps Making Changes Without Fixing the Bug

Symptoms: The participant repeatedly asks AI to fix the issue, resulting in increasingly large code changes.

Likely Cause: The AI does not have sufficient evidence or the participant has not isolated the failure.

How to Investigate:

Stop modifying the code and return to:

Expected → Observed → Evidence

Instructor Hint:

Ask:

"What do we actually know is failing?"

Solution: Collect evidence, isolate the failing component, then give the AI a focused debugging task.

Instructor Notes

- Prioritize understanding authentication and persistence over covering every Firebase feature
- Keep the Firebase data model simple
- Do not spend excessive time explaining Firebase internals
- Make participants test authentication and database functionality separately

- Encourage participants to inspect Developer Tools before asking instructors for solutions
- Ask participants what they expected and what they observed
- Avoid immediately providing fixes
- Encourage small changes followed by testing
- Treat live errors as debugging opportunities when appropriate
- Emphasize that authentication does not automatically mean database data is secure
- Introduce hooks only after participants have worked with AI coding assistants enough to recognize repetitive tasks
- Do not require every participant to implement hooks if time is limited
- Reserve substantial time for instructors to move around and assist participants

---

Participant Questions

Record useful questions that come up during practice sessions or the actual workshop.

- TBD during workshop

Resources

Documentation

- Firebase documentation
- Firebase Authentication documentation
- Cloud Firestore documentation
- Firebase Security Rules documentation
- Browser Developer Tools documentation
- Claude Code documentation
- Codex documentation

References

- Firebase setup guide
- Authentication flow reference
- Firestore data model example
- Debugging workflow cheat sheet
- AI debugging prompt sheet
- Hooks reference

Examples

- Example Firebase configuration
- Example authentication flow
- Example Firestore collection
- Example user-specific data structure
- Example debugging scenario
- Example hooks

Workshop Materials

Slides/Html

Create a short presentation covering:

- Website vs. data-driven application
- Authentication
- Firebase
- Firestore
- User-specific data
- Debugging with evidence
- Browser Developer Tools
- AI-assisted debugging
- Hooks and workflow automation

Handout / Cheat Sheet

Create a reference containing:

Authentication flow:

Sign Up → Authenticate → User ID → Application

Data flow:

User Action → Application → Firestore → Persistent Data

Debugging workflow:

Expected → Observed → Evidence → Hypothesis → Fix → Test → Verify

AI workflow:

Investigate → Provide Context → Evaluate Recommendation → Apply → Test

Repository

Use the same practice application from Day 1 when possible.

The Day 2 version should demonstrate authentication and persistent data without becoming a complete solution to the competition challenge.

Additional Files

- Firebase setup guide
- Authentication reference
- Firestore reference
- Debugging cheat sheet

- AI debugging prompts
- Hooks prompt sheet

End-of-Workshop Check

Participants should be able to demonstrate:

- [ ] Working user registration
- [ ] Working user sign-in and sign-out
- [ ] Application behavior that responds to authentication state
- [ ] Persistent Firestore data
- [ ] User-associated data
- [ ] Ability to locate and read browser console errors
- [ ] Ability to explain expected vs. actual behavior during debugging
- [ ] Effective use of Codex or Claude Code during investigation
- [ ] Understanding of when hooks may be useful
- [ ] A working GitHub checkpoint containing the Day 2 implementation

---

Improvements for Next Time

Complete this section after delivering the workshop.

What worked well:

What caused confusion:

What should be changed:

What took longer than expected:

Participant feedback: