CODE THE DREAM

# Learning Objectives: Full-Stack JS Practicum

*Last updated Summer 2025.*

Below is an overview of Code the Dream's full-stack practicum curriculum. Each sprint is two weeks long and has specific learning objectives that the team should accomplish.

| # | Sprint Name | Team Goals | Frontend Goals | Backend Goals | Deliverables |
|---|---|---|---|---|---|
| 1 | Foundation & Framing<br><br>**Note**: NO CODING | • Get to know your team and define working norms.<br>• Set weekly schedule (full team + frontend/backend check-ins).<br>• Agree on app idea and user problem.<br>• Define MVP scope and key features.<br>• Set up Jira board with Sprint 1 tasks<br>• Create and organize team Github repo. | • Collaborate on rough wireframes for core screens.<br>• Identify needed UI components and page structure.<br>• Coordinate with backend for API needs. | • Draft a simple data model/schema sketch.<br>• List initial API routes needed by frontend.<br>• Coordinate with frontend on data needs. | • Written problem statement + target user persona.<br>• Prioritized MVP feature list.<br>• Low-fidelity wireframes for core screens.<br>• Initial data model sketch (ERD or list).<br>• List of planned API endpoints.<br>• Project board with Sprint 1 planning tasks completed.<br>• Weekly meeting schedule documented. |
| 2 | MVP Kickoff | • Begin implementing MVP features with clear task assignments.<br>• Review and revise wireframes based on team feedback.<br>• Maintain clear communication and consistent standups.<br>• Use Jira to manage Sprint 2 tasks and update | • Build out UI for core MVP screens (e.g. homepage, login, dashboard).<br><br>• Set up routing and shared layout components.<br>• Connect to backend endpoints using mock or real data.<br>• Begin implementing basic form handling, user input, | • Set up backend project structure (e.g. routes, controllers, database config).<br>• Implement core API endpoints for MVP features.<br>• Build out database schema and seed basic test data.<br>• Support frontend with | • Working UI for at least 1–2 core user flows or screens (frontend).<br>• Functional backend routes for corresponding data.<br>• Connected frontend/backend interaction for basic data exchange (e.g. fetch user, create item).<br>• Updated wireframes if there are design or scope changes |

| | | | | | |
|---|---|---|---|---|---|
| | | statuses.<br>● Ensure every team member is contributing to code, not just planning. | and display logic | clean, testable API responses | ● Jira board updated daily with task movement and assignments.<br>● At least 1 merged PR per team member using feature branches.<br>● Documentation started: README updates, API route list, or schema diagram. |
| 3 | Feature Expansion | ● Continue building out core MVP features.<br>● Refine and update Jira board to reflect new or adjusted tasks.<br>● Communicate blockers early and seek mentor help.<br>● Review progress against MVP goals and adjust scope if needed.<br>● Maintain consistent GitHub workflows (feature branches, PRs, code reviews). | ● Complete all **c**ore UI screens (e.g. login, dashboard, main user flows).<br>● Connect remaining components to live API endpoints.<br>● Implement loading states, form validation, and error handling.<br>● Begin integrating JavaScript packages (e.g. date utilities, form libraries)<br><br>Document remaining screens needed to complete app (e.g. settings, profile, confirmations.<br>● Start writing basic test scripts (e.g. using Jest or Vitest) | ● Complete all API endpoints for MVP features.<br>● Begin integrating packages (e.g. authentication, validation, file handling).<br>● Begin writing basic test scripts for routes or utilities (e.g. Jest, Supertest).<br>● Ensure test data is seeded and accessible to frontend.<br>● Add error handling, input validation, and consistent response formats. | ● All core MVP features and core UI screens functional and connected.<br>● End-to-end data flow established (frontend → backend → database).<br>● At least one test script started (frontend or backend) and committed.<br>● Remaining screens and stretch features added to Jira backlog.<br>● Updated API route list, schema diagram, and README as needed. |
| 4 | Testing & Polish | ● Finalize remaining MVP screens and features.<br>●<br>● Review all user flows and address usability gaps. | ● Complete all remaining UI screens (e.g. settings, profile, confirmations).<br>●<br>● Refactor repetitive code | ● Finalize remaining API endpoints and error handling.<br>●<br>● Refactor backend logic | ● All MVP features fully complete.<br>●<br>● All UI screens finished and styled consistently. |

| | | | | | |
|---|---|---|---|---|---|
| | | • <br>• Maintain clear, updated Jira board with final tasks. <br>• <br>• Increase focus on testing, refactoring, and code quality. <br>• <br>• Prepare for final sprint (presentation, polish, and deployment if applicable). | into reusable components or hooks. <br>• <br>• Improve form validation, user feedback, and visual polish. <br>• <br>• Write or expand frontend test coverage using Jest, Vitest, or similar. <br>• <br>• Confirm all API integrations work smoothly and handle edge cases. | (controllers, services, routes) for clarity and modularity. <br>• <br>• Add or improve unit and integration tests (Jest, Supertest). <br>• <br>• Improve API documentation and schema clarity for the team <br>• <br>• Verify database constraints, seed data, and support for test cases. | • <br>• GitHub contains clean, refactored codebase with feature branches merged. <br>• <br>• Functional frontend and backend test suites, with test results committed. <br>• <br>• Documentation updated: README, API route list, schema, test instructions. <br>• <br>• Final Sprint 4 tasks marked complete in Jira and prep tasks added for Sprint 5. |
| 5 | Final Touches & Launch Prep | • Complete final QA: test all user flows and fix remaining bugs.<br>• Polish UI/UX for a smooth, user-friendly experience.<br>• Ensure clear task ownership for final deliverables.<br>• Keep GitHub clean: review, approve, and merge final PRs.<br>• Finalize all documentation and instructions.<br>• Prepare for presentation and demo (team narrative, feature highlights) | • Address any remaining UI/UX issues or feedback from mentors.<br>• Ensure all final screens are responsive and accessible.<br>• Polish styling and layout for presentation readiness.<br>• Complete frontend testing and clean up unused components or code.<br>• Confirm error states, modals, toasts, and alerts behave correctly. | • Final round of bug fixes and test coverage improvements.<br>• Ensure consistent API behavior (status codes, messages, errors).<br>• Clean up console logs, commented code, and test routes.<br>• Verify database is seeded with demo/test data if needed.<br>• Double-check API docs, schema, and readme are complete. | • All MVP and planned stretch features completed and functioning.<br>• Fully responsive and styled frontend.<br>• Frontend/backend code refactored, cleaned, and committed.<br>• Robust test coverage across both frontend and backend (even if minimal).<br>• Finalized documentation: README, API route list, schema, setup instructions.<br>• GitHub repo clean and production-ready (no leftover branches, no console.logs).<br>• Ready to present: clear summary of problem, features, and demo flow. |