ada developers academy

Ada Core Curriculum Overview

Full-Stack Web Development

The Ada Developers Academy core program teaches full-stack web development with a practical focus. We believe that this skillset that allows students to grow quickly, transition to industry, and to choose what specialty of software development they want to pursue in their post-Ada careers.

We cover Python, SQL, Flask, HTML/CSS, JavaScript, React, and computer science fundamentals. We focus on teaching skills that are readily transferable from one technology stack to another, making Ada students adaptable and flexible candidates for all types of developer positions.

Objectives

- Learn How to Learn
    - Create a model of independent self-learning study skills to facilitate learning new technologies
    - Implement features in high-level code for medium-sized systems
    - Establish a model of debugging skills to facilitate debugging, reading, and understanding existing code in medium-sized systems
- Prepare for Industry
    - Apply coding and communication skills combined to team-based software development
    - Combine code skills and job preparation activities to prepare for a successful transition to software development
- Build Communities Rooted in Social Justice
    - Integrate skills learned from social justice, advocacy, diversity, equity, and inclusivity to co-author a supportive learning community

ada developers academy

Core Content: Technical Overview

Precourse: Programming Fundamentals

Timing: -3 weeks to Orientation Week

Objective: Review and practice core programming concepts.

- Variables and scope
- Functions
- Iteration
- Control Flow (conditionals)
- Lists and Dictionaries

Unit 1: Programming Fundamentals

Timing: Orientation Week to Week 6

Objective: Further core programming concepts, learn computer science fundamentals and explore test-driven development, exception handling, data structures, Big O, object-oriented programming, and version control with Git/Github.

- Practice logical problem-solving using guided techniques to break down problems and utilize pseudocode
- Create and iterate over nested data structures
- Use version control to record, maintain, and collaborate on projects
- Identify, debug, and handle errors by using the stack trace, VS Code's debugger
- Handle exceptions using try and except clauses
- Explore the relationship between memory allocations and arrays in Python
- Use linear and binary search algorithms to search through array
- Write and utilize automated unit tests using Pytest
- Dive into Object-oriented programming by building and testing classes with instance and class methods in Python
- Utilize inheritance and composition
- Understand the relationship between memory, references, and values
- Analyze algorithms using Big O notation
- Understand what an LLM is and the common pitfalls when using an AI helper such as ChatGPT
- Ask ChatGPT to do something and validate its responses
- Use ChatGPT to debug and better understand sections of code

ada developers academy

Unit 2: Intro to Back-End Web Development

Timing: Week 7 to Week 12

Objective: Learn the fundamentals of back-end development; design, build, and maintain a database using PostgreSQL; design APIs in Flask to interact with databases and extend programs using third-party APIs.

- Use CRUD actions and SQL Queries to build, modify, and interact with databases
- Create one-to-many and many-to-many relationships between tables
- Design and build an API to explore HTTP, REST, and the client/server model relationship
- Build and design requests and responses using data and JSON
- Build an API in Flask to define endpoints, query params, create models, and perform CRUD actions using HTTP requests
- Test APIs using Postman and Pytest
- Deploy APIs using Heroku
- Explore use cases for hash tables and recursion in problem-solving
- Collaborate in small groups to practice agile and maintaining projects using git
- Make calls to the OpenAI API to use ChatGPT programmatically
- Review code generated by ChatGPT

Unit 3: Intro to Front-End and Full-Stack Web Development

Timing: Week 13 to Week 19

Objective: Learn the basics of front-end development; design and build static websites using HTML, CSS, and JavaScript; develop web apps in React to present data and handle user interaction.

- Organize web content in HTML, using best semantic HTML practices
- Apply styles to web pages using CSS selectors by element type, class, ID, and relationship in the DOM
- Design the layout and flow of elements using CSS Grid and Flexbox
- Solve problems using variables, loops, and functions in ES6
- Utilize automated tests in JavaScript
- Manipulate the DOM and handle events in JavaScript
- Make and handle asynchronous API calls in JavaScript
- Create and render functional components in a React JS webapp
- Manage props and PropTypes in a React JS webapp
- Manage state using the useState hook in a React JS webapp

- Make API calls using the useEffect hook in a React JS webapp
- Practice full-stack development with distinct front-end and back-end layers
- Understand how GitHub Copilot can be used as a tool to augment software development

Capstone: Independent Project

Timing: Week 20 to Week 25

Objective: Synthesize coding skills; scope and manage a project; develop a working solution; produce and demonstrate work; express personal creativity or passions; and display potential as a software developer.

- Demonstrate self-direction, time management, and independent learning
- Learn and implement new technologies
- Complete a product life cycle from conception to delivery, including deployment
- Utilize agile practices learned to assist in project completion
- Create and deliver a compelling technical presentation

Unit 4: Computer Science Fundamentals (CS Fun)

Timing: Week 26 to Week 47

Objective: Prepare for technical interviews by expanding upon data structures and algorithms content

- Linked Lists
- Binary Search Trees
- Stacks
- Queues
- Divide and Conquer
- Dynamic Programming
- Graphs: Breadth-first search, Depth-first search, Dijkstra's algorithm

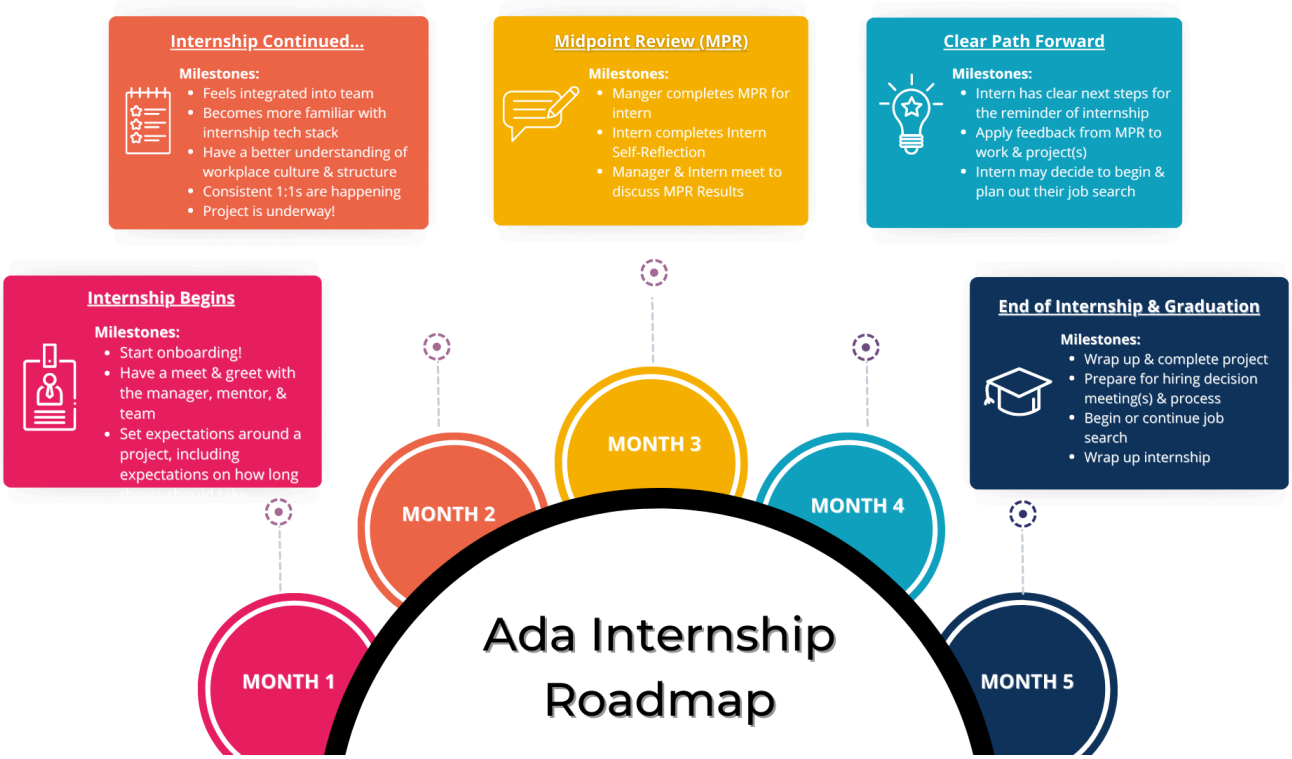
ada developers academy

Workplace & Career Development Overview

Applied Learning: Internship

Timing: Week 26 to Week 47

Objective: Become a hireable junior developer; build technical, communication, and problem-solving skills in a real-world production environment; learn about the diverse day-to-day experiences of tech workers; prepare for your job search and career after Ada.



View a larger version of the Ada Internship Roadmap.

Live Sessions

Timing: Week 20 to Week 47

Objective: Take personal ownership of career decisions; develop foundational skills to lead an entry-level software developer job search; practice self-advocacy in tech/corporate workplaces; build community with peers in support of each other's careers; and navigate changes and challenges in the tech industry with a growth mindset.

ada developers academy

Workplace Development	Career Development
• Self-Advocacy: Understanding Your Values and Needs • Managing Up & Setting Boundaries • Internship Prep & Best Practices • Building Your Practice of Self-Care	• Networking & Relationship-Building • Creating Your Elevator Pitch • Creating Your Job Search Plan • Your Resume • Your LinkedIn • Technical Recruiting & Hiring • Interviewing Foundations • The Behavioral Methods of Technical Interviewing • Offer & Salary Negotiation

Resources Required

We require that students have a reliable workspace with internet access.

We require that students use a MacBook that meets the following specifications:

- Macbook Air with M1 (2020 or newer)
- Macbook Pro (2017 or newer)
- 8 GB RAM minimum
- 64 GB drive space minimum, with 8 GB space available

Software or third-party tools used in the classroom:

- GitHub (GitHub Student Pack)
- Learn our Learning Management System
- Google (GSuite)
- Slack
- Zoom
- Replit
- Render
- VSCode
- OpenAI