AI CERTs™

AI+ Vibe Coder (4 Hours)

Program Detailed Curriculum

Executive Summary

The AI+ Vibe Coder certification equips learners with essential skills to thrive in the evolving world of artificial intelligence and coding. Designed for beginners and professionals seeking foundational knowledge, the program emphasizes core AI concepts, coding practices, and ethical applications. Participants gain hands-on experience with AI tools, algorithms, and problem-solving techniques to build practical solutions. The certification fosters critical thinking, creativity, and collaboration, ensuring graduates are prepared for future opportunities in AI-driven industries. By completing this program, learners demonstrate their readiness to contribute effectively to projects that combine innovation, technical skills, and responsible AI development.

Course Prerequisites

- **Basic Computer Skills** – Comfortable with operating systems and files.
- **Mathematics Fundamentals** – Understanding of algebra and basic statistics.
- **Logical Thinking** – Ability to approach problems step by step.
- **Programming Curiosity** – Interest in learning coding from scratch.
- **English Proficiency** – Ability to follow technical instructions clearly.

Module 1

Introduction to Vibe Coding & AI Tools

1.1: What is Vibe Coding?

- **Concepts:** Vibe coding uses AI to translate natural language into code. Key concepts include the role of NLP in understanding prompts and the importance of reducing barriers to coding for non-technical users.
- **How Does Vibe Coding Work:** Vibe coding involves three key components: the user's prompt, the AI tool interpreting it, and the generated code. This process allows for the rapid creation of both simple scripts and complex applications.
- **How to Use the Generated Code:** Once generated, users can copy AI-generated code into their development environment, customize it, and test it. AI tools also assist with debugging and adapting the code to specific project needs.
- **Use Case:** Vibe Coding for Rapid Prototyping in a Start-Up: A tech start-up uses vibe coding to quickly build MVPs by generating both frontend and backend code, reducing development time and enabling faster iteration.

1.2: Evolution of AI in Software Development – Low Code vs No Code vs Vibe Coding

- **Discussion:** The Evolution of AI in Software Development: This section explores how low-code and no-code platforms simplify development, and how vibe coding takes it further by automating the entire process using natural language prompts.
- **Example Use Case of Vibe Coding vs Low Code and No Code:** Compare practical examples where low-code and no-code tools fall short in handling customization, while vibe coding generates tailored solutions directly from natural language descriptions.
- **Use Case: Low Code vs Vibe Coding in Enterprise IT:** An enterprise IT department adopts both low-code and vibe coding tools to rapidly create a complex internal application, improving efficiency and collaboration across teams.

1.3: Overview of Common AI Coding Tools by Functionality

- **Some of the Most Popular AI-Powered Coding Tools:** This section introduces popular AI coding tools, detailing their functionalities and use cases. Tools like GitHub Copilot enhance coding efficiency, while ChatGPT and Claude assist with generating secure, reliable code.
- **How to Use the Generated Code:** Once AI tools generate code, users can copy it into their IDE, customize it as needed, test it, and deploy the app. Each AI tool offers unique features to refine the generated code.

1.4: SDLC for a Vibe Coding Product

- **Basic SDLC for a Traditional Product:** Traditional SDLC involves distinct phases, including requirements gathering, system design, coding, testing, deployment, and maintenance, often requiring manual coding and debugging.
- **SDLC for Vibe Coding Products:** Vibe coding streamlines the SDLC by automating tasks such as code generation and debugging. AI tools assist throughout the entire development process, improving efficiency and accessibility.
- **Example of SDLC for a Vibe Coding Product:** An example demonstrates how vibe coding tools assist in building a task management app, from requirement gathering to deployment, showing how AI-generated code accelerates the development lifecycle.

1.5: Hands-on Lab: Familiarizing Learners with Multiple AI Coding Tools

In this lab, learners will practice using GitHub Copilot, ChatGPT, and Claude to generate a simple "Hello World" program. They'll compare the tools' ease of use, output quality, and functionality.

1.6: Case Studies

- **Case Study - 1:** AI-Powered Plywood Cutting Visualizer Using Vibe Coding: WoodWorks Innovations used vibe coding tools to develop a Plywood Cutting Visualizer, automating cut optimization and visual planning. AI-driven development enabled rapid prototyping, cutting material waste, and improving workflow efficiency.
- **Case Study - 2:** AI-Driven Resume Scoring App Development Using Vibe Coding: TalentTech Solutions leveraged vibe coding to create a Resume Scoring Agent, automating resume analysis, scoring, and ranking based on job descriptions. This streamlined recruitment, reducing manual review time and bias.

Module 2

Prompting for Code – Basic and Best Practices

2.1: Anatomy of a Good Prompt

- **Key Components of a Good Prompt:** Explore the core elements of a prompt, such as clarity, specificity, context, and directness, to ensure the AI generates accurate and functional code**.**
- **Why a Good Prompt Matters:** Understand the importance of crafting a good prompt and how it directly impacts the quality, accuracy, and efficiency of AI-generated code.

2.2: Prompt Types – Instructive, Descriptive, Iterative

- **Concepts and Description:** Gain insight into the characteristics of instructive, descriptive, and iterative prompts, including their practical use cases in guiding AI tools.
- **Explanation with Examples:** Examine examples of each prompt type and discover how they influence the AI's ability to generate high-quality, functional code.

2.3: Prompting Patterns – Zero-Shot, Few-Shot, Chain-of-Thought

- **Concepts and Description:** Understand the role of different prompting patterns, including zero-shot, few-shot, and chain-of-thought, in guiding AI towards optimal code generation.
- **Explanation and Example:** Review practical examples of each prompting pattern to see how they can be applied to real-world coding tasks for more accurate AI responses.

2.4: Hands-on Lab: Practice Zero-Shot, Few-Shot, and Chain-of-Thought Prompting

Engage in a hands-on lab to refine prompts and observe how different prompting techniques improve AI-generated code through iterative refinement.

2.5: Use-Cases

- **Use-Case 1:** Creating a Python Calculator: Apply different prompting techniques to develop a Python calculator, demonstrating how prompt clarity and structure enhance the AI's code generation.
- **Use-Case 2:** Optimizing AI-generated Code Using Different Prompt Types: Refine a simple prompt to create a Python function for factorial calculation, using various prompting methods to enhance code accuracy and efficiency.

2.6: Assignment (Task) for Self-learning

Complete practical assignments such as creating a Python calculator or a to-do list app to reinforce prompt crafting and coding skills.

Module 3

Debugging & Testing via AI

3.1: Reviewing and Refining AI-generated Code

- **Concepts and Description:** Understand the importance of reviewing AI-generated code to ensure it performs intended tasks, handles edge cases, and follows best practices for optimization and error handling.
- **Discussion:** Refining AI-Generated Code: Explore techniques for refining AI-generated code, focusing on optimizing algorithms, improving readability, and handling errors effectively to ensure robust and efficient software.

3.2: Prompting for Bug Fixes and Test Coverage

- **Concepts and Description:** Gain insight into bug fixing and test coverage in AI-generated code, learning the importance of identifying bugs and creating unit tests to ensure software reliability and correctness.
- **Examples and Code Walkthrough:** Follow detailed examples of bug fixing and generating unit tests for a Python function, demonstrating how AI can assist in these processes to enhance code quality.

- **How to Use the Generated Code:** Learn how to implement AI-generated bug fixes and unit tests in a development environment, including copying code, testing, and refining based on output analysis.
- **Discussion:** Why Bug Fixing and Test Coverage are Important: Explore the significance of debugging and test coverage in software development, and how AI tools can accelerate the process, ensuring robust and functional applications.

3.3: Using AI-generated Unit Testing

- **Concepts and Description:** Understand unit testing's role in software development and how AI tools can automatically generate tests to verify code behavior, ensuring the software functions as intended.
- **Example - 1 and Code Walkthrough:** Walk through a practical example where AI generates unit tests for a function that calculates averages, demonstrating how to test code for correctness and edge cases.
- **Example - 2 and Code Walkthrough:** Examine an example where AI generates unit tests for a palindrome checker function, showing how tests cover edge cases and ensure accurate functionality.
- **Importance of AI-Generated Unit Testing:** Discover the benefits of using AI-generated unit tests, including increased speed, improved code quality, and comprehensive coverage of various input scenarios.

3.4: Detecting Hallucinations and Unsafe Code

- **Concepts and Description:** Understand hallucinations in AI-generated code and unsafe practices like SQL injection. Learn methods to detect these issues and refine code for safety and accuracy.
- **Discussion:** Why Detecting Hallucinations and Unsafe Code is Important: Discuss the importance of detecting logical errors (hallucinations) and security vulnerabilities (unsafe code), and how to ensure AI-generated code is secure and reliable.

3.5: Hands-on Lab: AI-Assisted Debugging and Unit Testing

This lab provides practical experience in using AI tools for debugging and testing code, simulating real-world scenarios to practice identifying and fixing bugs.

3.6: Activity Section

- **Activity - 1:** Finding Bugs in the Provided Code: Analyze a bug-ridden to-do list code snippet, identify issues, and answer questions to demonstrate your ability to debug AI-generated code effectively.
- **Activity - 2:** Bug-Ridden Code Snippet Example: Login Form Validation: Review and debug a login form validation system, focusing on detecting and fixing logical errors in AI-generated code.

Module 4

Building a Simple Full-Stack App with Prompts

4.1 Planning the App: Frontend + Backend

- **Concepts and Description:** Understanding full-stack development, focusing on the integration of frontend (UI) and backend (server-side) components in building cohesive web applications.
- **Code Example and Explanation:** In this section, learners will build a simple task management app, implementing both frontend (HTML + JavaScript) for task input and display, and backend (Node.js + Express) for handling and storing tasks.

4.2 Using IDEs and Code Generators to Scaffold Code

- **Concepts and Description:** An overview of Integrated Development Environments (IDEs) and how AI tools like GitHub Copilot can speed up the coding process, reducing manual effort in scaffolding full-stack applications.

4.3 Connecting Components Using Natural Language

- **Concepts and Description:** Overview of the frontend-backend communication process, where frontend sends requests (GET, POST) to backend, processes data, and dynamically updates the user interface.
- **Code Example and Explanation:** This section demonstrates how to connect the frontend and backend of a task management app using the fetch API, handling task data with POST and GET requests, and managing tasks with in-memory storage.

4.4 Deploying and Testing the MVP in Simulated Environment

- **Concepts and Description:** The concept of MVP and why it's critical to deploy early versions of applications with core functionalities to gather feedback and iterate improvements.
- **Code Example and Explanation:** This section covers local testing and deployment of a task management app, using Netlify for the frontend and Heroku for the backend. Learners will test the app locally before deploying it for public access.

4.5 Hands-on Lab: Building and Connecting the Frontend and Backend for Contact Form Submission

Learn how to build a full-stack contact form application with frontend (HTML) and backend (Express.js), connecting them via the fetch API. Test locally and deploy to Heroku (backend) and Netlify (frontend).

4.6 Hands-on Lab: Building a Standalone Desktop Calculator Application Using Tkinter

In this lab, learners will build a standalone desktop calculator app using Python and Tkinter. The app will handle arithmetic operations, store results in memory, and provide an interactive user interface.

4.7 Assignment and Task (Self-learning)

- **Hands-on Assignment 1:** Task Management System - Full-Stack Development Using Prompts:Building a task management system with CRUD operations, integrating frontend and backend using prompts, and testing the application locally before deployment.
- **Hands-on Assignment 2:** AI-Powered Contact Form with User Validation: Creating a contact form with user validation and backend processing, ensuring proper validation and handling of user inputs, and deploying the app for real-world use.

Module 5

Code Ethics, Security, and AI Limits

5.1 AI Limitations and Biases

- **Concepts and Description:** Introduces the fundamental issues with AI limitations, such as its inability to understand context and biases, highlighting how data and algorithms influence AI-generated outcomes.
- **Discussion on AI Bias and Limitations:** Discusses the practical consequences of biases in AI models, providing examples of biased AI-generated code and explaining the importance of mitigating such risks in development.

5.2 Prompt Injection and Mitigation Strategies

- **Concepts and Description:** Introduces prompt injection, explaining how attackers manipulate AI inputs to generate harmful outputs and the resulting risks to system security and data privacy.
- **Example and Code Walkthrough:** Presents a practical example of prompt injection in AI-generated code, explaining how harmful commands can be inserted and providing solutions for sanitizing inputs.

5.3 Data Privacy and Secure Coding (Non-Technical Focus)

- **Concepts and Description:** Defines data privacy and secure coding, explaining the need to safeguard sensitive data and adhere to security best practices when using AI tools for coding.
- **Examples of Data Privacy and Secure Coding Practices:** Presents non-technical examples of how AI can be used to secure data, focusing on encryption and secure transmission practices to protect user privacy.

5.4 Responsible Use of AI in Production

- **Concepts and Description:** Discusses what constitutes responsible AI use, emphasizing accountability, transparency, and avoiding harmful outcomes through ethical AI design and deployment.
- **How to Use AI Responsibly in Production:** Outlines practical steps for deploying AI systems in production, such as regular audits, bias detection, and maintaining data privacy and security throughout the AI lifecycle.

5.5 Hands-on Lab: Build Awareness of AI Limitations and Responsible Practices

Provides a practical lab where learners apply ethical practices by identifying unsafe AI behaviors and re-engineering prompts to generate secure, fair AI-generated code.

Module 6

Capstone Project – Prompt-Driven App

6.1 Apply All Learned Skills in a Real-World Project

- **What Does It Mean to Apply Learned Skills in a Real-World Project?:** Applying skills means taking theoretical knowledge and turning it into practical solutions. Learners will create fully functional apps, from frontend design to backend logic, using AI assistance for code generation.
- **Prompt for AI to Scaffold Code Examples:** Learn how AI tools like GitHub Copilot and ChatGPT can generate scaffolding for apps, such as a to-do list, by providing clear instructions to build backend routes and frontend interactions.

6.2 Collaborate and Iterate Using AI Tools

- **What Does It Mean to Collaborate with AI Tools?:** Collaboration with AI tools involves using AI as a virtual co-developer. These tools speed up development, automate tasks, and help learners continuously improve code through iterative feedback loops and suggestions.
- **Examples of Using AI Tools for Iterative Development:** Learners will see how AI tools help iterate on real-world projects like To-Do Lists or Note-Taking apps by generating code, suggesting optimizations, and streamlining the development process.

6.3 Demonstrate End-to-End Development Using Prompts

- **What Does End-to-End Development Mean:** End-to-end development refers to creating an application from scratch, covering everything from UI design to backend logic. Learners use AI tools to streamline the development process and ensure app functionality.
- **Example of End-to-End Development:** AI-Powered To-Do List Application: In this example, learners will build a To-Do List app, using AI prompts for both frontend and backend, to handle tasks like adding, editing, and deleting, all while ensuring integration.

6.4 Capstone Project Use Case: AI-Powered To-Do List Application

This use case focuses on creating a full-stack To-Do List app. Learners will use AI tools to build the app's frontend, backend, and storage, creating a functional MVP for real-world scenarios.

6.5 Capstone Project Use Case: AI-Powered Note-Taking Desktop App

Learners will build a frontend-only Note-Taking app, focusing on UI design, storage, and CRUD operations. AI tools will guide the process, offering prompts for design, logic, and feature refinement.

6.6 Assignment and Task (Self-learning)

- **Capstone Project 1:** AI-Powered Task Management Web App: Learners will create a task management web app using HTML, CSS, and JavaScript, leveraging AI tools for efficient UI and logic generation, with localStorage for persistent task management.
- **Capstone Project 2:** AI-Powered Personal Journal App: A personal journal app will be built using AI tools for UI generation and localStorage for **data persistence. Learners will implement CRUD operations and refine the app iteratively with AI assistance.**
- **Capstone Project - 3:** AI-Powered Budget Tracker Web App: In this project, learners will create a budget tracker that categorizes expenses. AI tools will assist with UI, logic, and expense management, ensuring the app is functional and tracks spending efficiently.

6.7 Use Case

A real-world example where learners apply AI-driven techniques to build an AI-Powered Budget Tracker, demonstrating the power of AI in simplifying app development, while managing and tracking finances.