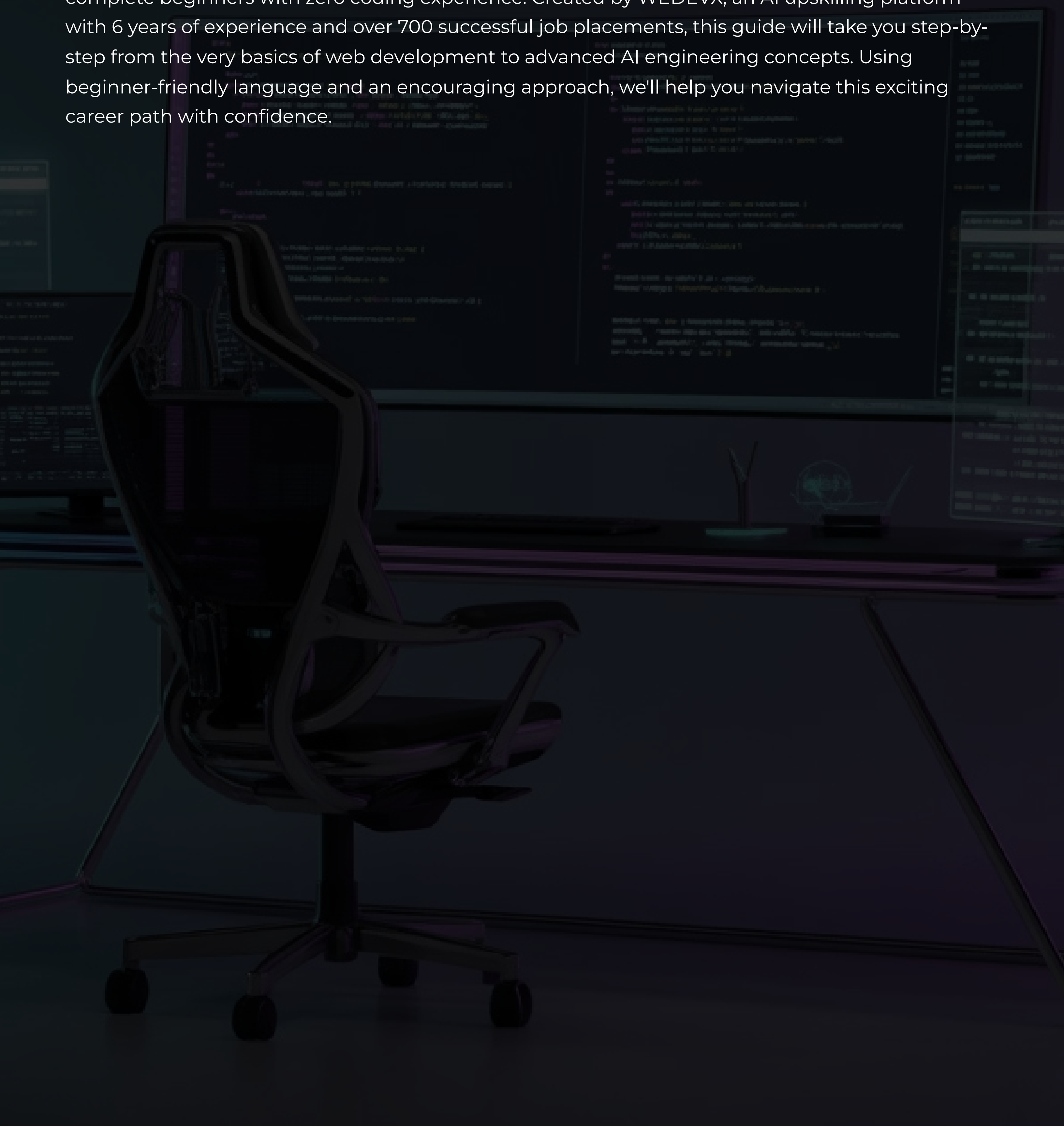
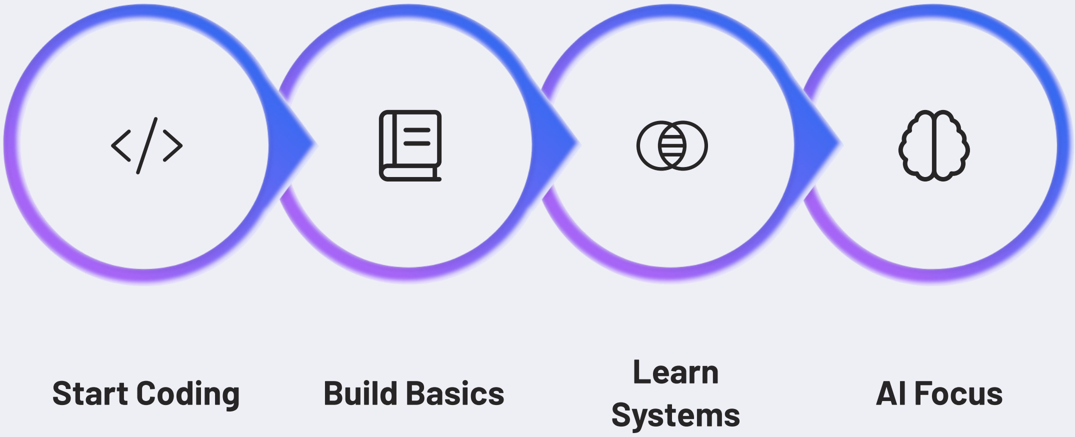
AI Engineering Roadmap: From Zero to Hero

Welcome to your comprehensive journey into AI engineering! This roadmap is specially designed for complete beginners with zero coding experience. Created by WEDEVX, an AI upskilling platform with 6 years of experience and over 700 successful job placements, this guide will take you step-by-step from the very basics of web development to advanced AI engineering concepts. Using beginner-friendly language and an encouraging approach, we'll help you navigate this exciting career path with confidence.



Your Learning Journey Overview

Before diving into the specifics, let's understand the big picture of your learning path. This roadmap is divided into two major sections: Full-Stack Software Engineering (Weeks 1-24) and AI Engineering (Modules 1-7). This strategic approach ensures you build a solid foundation in software development before specializing in AI.



You'll notice that we start with modern, exciting tools that make coding fun and accessible, then gradually build up your technical skills. This approach keeps you motivated while ensuring you develop the comprehensive knowledge needed for success in the AI field.

Don't worry about the technical terms you see now - each one will be explained thoroughly when we reach that stage of learning. Remember, every expert was once a beginner, and this roadmap is designed to take you from zero knowledge to professional competence at a comfortable pace.

Weeks 1-2: Vibe Coding with Modern Tools

Your journey begins with what we call "Vibe Coding" - a fun, low-pressure introduction to programming using the most beginner-friendly modern tools available. During these first two weeks, you'll get comfortable with the idea of writing code without getting bogged down in complex theory or intimidating setups.

You'll use three amazing tools that make coding approachable and even enjoyable for complete beginners:

- Replit: An online coding platform where you can write and run code directly in your browser - no complicated installations required!
- Lovable: A simple tool that helps you create charming digital experiences with minimal code
- V0: A cutting-edge tool that lets you build web interfaces through conversation

By starting with these state-of-the-art tools, you'll experience immediate results and build confidence. This approach is like learning to drive in a modern car with automatic transmission, navigation, and driver assistance - you'll focus on the journey rather than struggling with the mechanics.

Don't worry about mastering everything in these first two weeks. The goal is simply to get comfortable with the environment, have fun, and begin thinking like a programmer. You'll be surprised how quickly you progress when learning feels like play rather than work!

Week 3: HTML & CSS Fundamentals

Now that you've gotten your feet wet with some fun coding tools, it's time to learn the building blocks of every website: HTML and CSS. Don't worry - these are actually two of the most beginner-friendly languages in the tech world!

HTML Basics

HTML is like the skeleton of a website. It's not even a programming language - it's a markup language that tells browsers what each part of your page is (headings, paragraphs, images, etc.). You'll learn to create structured content using simple tags.

CSS Styling

CSS is what makes websites beautiful. It's responsible for colors, layouts, fonts, and animations. You'll learn how to transform your plain HTML into visually appealing designs that respond well on different devices.

During this week, you'll build several small projects that gradually increase in complexity. You might start with a simple personal profile page and end the week with a responsive layout that looks good on both computers and phones.

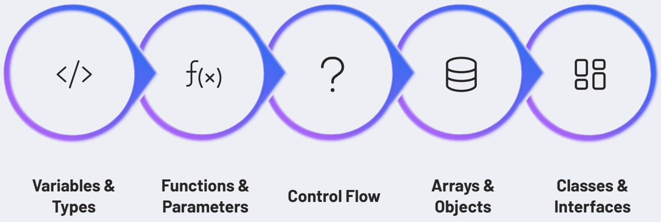
Remember, the web was built on HTML and CSS long before complicated frameworks came along. Even professional developers use these fundamentals every day. By understanding these core technologies, you're setting yourself up for success with everything that follows in this roadmap.

The best part? You can see your results instantly in the browser. Each new skill you learn immediately translates into visible improvements in your projects, which is incredibly satisfying and motivating!

Weeks 4-8: TypeScript for Beginners

Now we're ready to dive into your first true programming language: TypeScript. Don't let the name intimidate you - we'll take it step by step over these five weeks, ensuring you build a solid foundation.

Why TypeScript?



TypeScript is a supercharged version of JavaScript (the language that powers interactive websites). We're starting with TypeScript because it helps catch mistakes early and makes your code more reliable - perfect for beginners who are still learning!

During these five weeks, you'll gradually learn:

- Basic concepts like variables, functions, and control flow
- How to think logically like a programmer
- Working with different types of data
- Creating reusable code with functions and modules
- Object-oriented programming principles

We'll use plenty of real-world examples and mini-projects to make these concepts concrete. For instance, you might build a simple quiz game that tests your knowledge, or a tool that helps track your daily habits.

> "TypeScript might seem challenging at first, but it's actually protecting you from the confusing errors that plague JavaScript beginners. It's like learning to ride a bike with training wheels - they provide extra support while you're building confidence." - WEDEVX Instructor

By the end of week 8, you'll have transformed from someone who's never written code to someone who can create programs that solve real problems. This is a major milestone on your journey to becoming an AI engineer!

Weeks 9-11: ReactJS for Frontend Development

Congratulations on making it through TypeScript! Now it's time to put those skills to use by learning ReactJS, the most popular frontend framework used by companies worldwide. During these three weeks, you'll discover how to build interactive user interfaces - the parts of applications that users actually see and interact with.

React Fundamentals

Learn the core concepts of React including components, props, and JSX syntax. You'll build your first simple React app and understand how React's component-based architecture makes building interfaces easier.

State Management & Hooks

Discover how to make your apps interactive by managing data with React's state system. You'll learn about hooks like useState and useEffect that give your components dynamic behavior.

Building & Deploying Projects

Create complete frontend applications and deploy them to the internet using Vercel. You'll finish with projects you can share with friends and family or include in your portfolio.

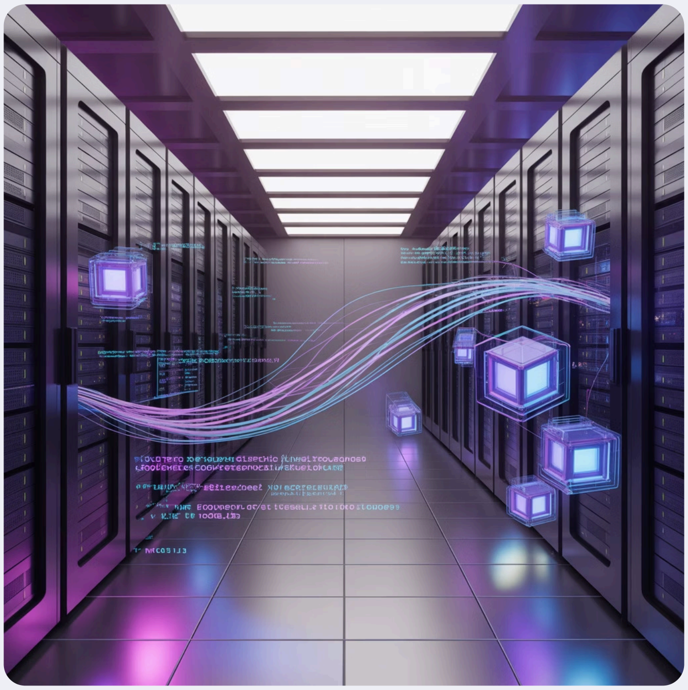
What makes React special is its component-based approach. Instead of building entire pages at once, you'll create reusable pieces (like buttons, forms, or navigation bars) that you can combine like LEGO blocks to build complex interfaces.

By the end of week 11, you'll deploy your projects to Vercel, a platform that makes your applications available online for anyone to use. There's nothing quite like the feeling of sharing your first web application with the world and saying "I built this!"

✓ **Pro Tip:** As you learn React, build projects that genuinely interest you. Whether it's a recipe finder, a workout tracker, or a movie recommendation app - your enthusiasm for the subject matter will keep you motivated through the challenging parts of learning.

Weeks 12-16: NodeJS for Backend Development

So far, you've learned how to create beautiful, interactive interfaces with HTML, CSS, and React. Now it's time to explore what happens behind the scenes with NodeJS - the technology that powers the server side of web applications.



Backend development is all about handling the parts of applications users don't directly see: storing data, authenticating users, processing payments, and communicating with other services. Think of it as the engine under the hood of a car - not visible to passengers, but essential for the journey.

During these five weeks, you'll learn:

- Setting up a NodeJS environment
- Creating API endpoints to communicate with your frontend
- Handling requests and sending responses
- Working with middleware for common tasks
- Managing user authentication and security
- Connecting to databases and external services

Client Request

A user interacts with your frontend, triggering a request for data or an action

Server Processing

Your NodeJS application receives the request, processes it, and performs necessary operations

Database Interaction

If needed, your server communicates with a database to store or retrieve information

Server Response

Your NodeJS application sends back the requested data or confirmation to the

Weeks 16-18: SQL for Databases

Now that you can build both frontend interfaces and backend servers, it's time to learn how to permanently store and organize your application's data using SQL (Structured Query Language). Database skills are crucial for any software engineer, and especially important for future AI engineers who will work extensively with data.

SQL has been the industry standard for databases for decades, and for good reason - it's powerful, reliable, and used by companies of all sizes. During these three weeks, you'll learn:

Database Fundamentals

Understand how databases are structured with tables, rows, and columns. Learn to create databases and tables with the right relationships between them.

CRUD Operations

Master the four basic database operations: Create (INSERT), Read (SELECT), Update (UPDATE), and Delete (DELETE) - the foundation of all database interactions.

Advanced Queries

Learn to write more complex queries using JOINs to combine data from multiple tables, aggregate functions to analyze data, and transactions to ensure data integrity.

Database Integration

Connect your NodeJS backend to a SQL database, allowing your applications to persistently store and retrieve data between user sessions.

We'll use practical examples throughout this section, like building a product inventory for an online store or a user management system for a social media application. These projects will help you see how databases fit into real-world applications.

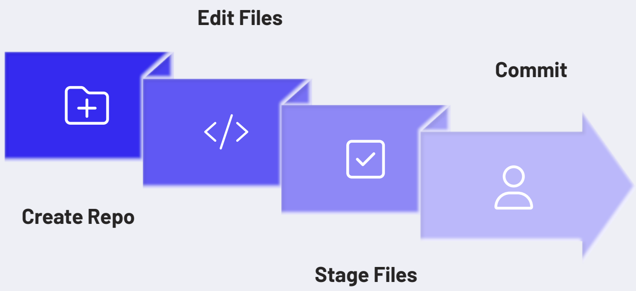
> "Databases might seem less flashy than frontend work, but they're the foundation that makes persistent applications possible. Without databases, all your user data would disappear when they close their browser!" - WEDEVX Instructor

By week 18, you'll be able to design database schemas, write efficient queries, and integrate database operations into your full-stack applications - completing the trifecta of frontend, backend, and database skills!

Week 19: Version Control with Git

Now that you've built several components of full-stack applications, it's time to learn how professional developers track changes, collaborate, and maintain their code using Git. This powerful version control system is an essential skill for every developer, regardless of specialization.

Think of Git as a time machine for your code. It allows you to save snapshots of your project at different points in time, experiment with new features without breaking what already works, and collaborate seamlessly with other developers.



During this week, you'll learn:

- Creating repositories to track your projects
- Committing changes with meaningful messages
- Branching to work on features in isolation
- Merging your changes back into the main codebase
- Resolving conflicts when multiple changes overlap
- Using remote repositories to back up your code

We'll practice these concepts with hands-on exercises, gradually building up your confidence with Git's command-line interface. You'll also learn how to recover from common mistakes - an invaluable skill even experienced developers rely on regularly.

Fun Fact: Git was created by Linus Torvalds, the same person who created the Linux operating system. He built Git because he needed a better way to manage the thousands of contributions to the Linux kernel!

By the end of this week, you'll be able to maintain a professional development workflow, track all changes to your code, and collaborate effectively with other developers. These skills will be immediately useful as we move into automation with GitHub Actions next week.

Week 20: Continuous Integration with GitHub Actions

Building on your Git knowledge, this week introduces you to GitHub Actions - a powerful automation tool that helps ensure your code is always working correctly. This concept, called Continuous Integration (CI), is a cornerstone of modern software development practices.

GitHub Actions allows you to create automated workflows that run whenever certain events happen in your repositories. For example, you can automatically test your code whenever someone makes a change, or deploy your application whenever you merge code into the main branch.

Understanding Workflows

Learn how GitHub Actions workflows are defined in YAML files and triggered by repository events like pushes, pull requests, or scheduled times.

Setting Up Automated Tests

Create workflows that automatically run your tests whenever code changes, ensuring that new additions don't break existing functionality.

Implementing Continuous Deployment

Configure workflows that automatically deploy your application to hosting services whenever certain conditions are met.

During this week, you'll work on practical examples like:

- Setting up automated linting to ensure code style consistency
- Creating test workflows that verify your application works correctly
- Building deployment pipelines that update your live application automatically
- Setting up notifications for successful or failed workflows

These automation skills are incredibly valuable in professional settings, where teams need to maintain high quality standards while moving quickly. By implementing CI workflows, you'll ensure that your applications remain stable as they grow in complexity.

By the end of the week, you'll have transformed your development process from manual to automated, saving time and reducing errors - a significant step toward professional-grade software engineering!

Week 21: Cloud Infrastructure with AWS

This week, you'll be introduced to Amazon Web Services (AWS), the world's most comprehensive and widely adopted cloud platform. Cloud services are essential for modern applications, providing scalable, reliable infrastructure without the need to manage physical hardware.

AWS offers hundreds of services, but we'll focus on the fundamental ones that every developer should understand:

- EC2 (Elastic Compute Cloud): Virtual servers in the cloud
- S3 (Simple Storage Service): Unlimited file storage
- RDS (Relational Database Service): Managed database servers
- Lambda: Serverless functions that run your code without servers
- IAM (Identity and Access Management): Security and permissions

You'll learn the core concepts of cloud computing, including:

- Infrastructure as Code (IaC)
- Pay-as-you-go pricing models
- Scaling applications to handle traffic
- Managing cloud resources responsibly

1

Create AWS Account

Set up your AWS account with free tier access and secure it with multi-factor authentication.

2

Deploy a Static Website

Upload a React application to S3 and make it publicly accessible through CloudFront distribution.

3

Launch a Backend Server

Deploy your NodeJS application to EC2 or as serverless functions with Lambda

4

Connect to a Database

Set up an RDS instance and connect your application to store and retrieve data

Weeks 22-24: System Design

These final three weeks of the software engineering portion focus on system design - the art and science of architecting software systems that are scalable, reliable, and maintainable. This is where you'll learn to think beyond individual components and consider how entire systems work together.

System design skills separate junior developers from senior engineers, and they're especially crucial for AI engineering, where you'll often work with complex, distributed systems.

Architectural Patterns

Learn common software architecture patterns like microservices, monoliths, and event-driven architecture. Understand their strengths, weaknesses, and appropriate use cases.

Scalability Principles

Discover how to design systems that can handle growing numbers of users and data. Learn about horizontal and vertical scaling, load balancing, and caching strategies.

Database Design

Dive deeper into database design, including normalization, indexing, and choosing between SQL and NoSQL solutions based on your application's needs.

API Design

Learn to create well-designed APIs that are intuitive, efficient, and future-proof. Understand RESTful principles and documentation standards.

During these weeks, you'll work on case studies and design exercises, practicing how to approach system design questions like those commonly asked in technical interviews. You'll learn to:

- Break down complex requirements into manageable components
- Make appropriate technology choices based on requirements
- Consider trade-offs between different approaches
- Design for reliability, security, and performance
- Communicate your design decisions clearly

By the end of week 24, you'll have completed the software engineering portion of the roadmap! You'll have the skills to design, build, and deploy complete applications - a solid foundation for the AI engineering specialization that follows.

AI Engineering: Core Concepts & Working with Data

Congratulations on completing the software engineering portion of the roadmap! Now we begin the specialized AI engineering modules, where you'll learn to build intelligent applications powered by large language models (LLMs).

Module 1: Core Concepts & Setup

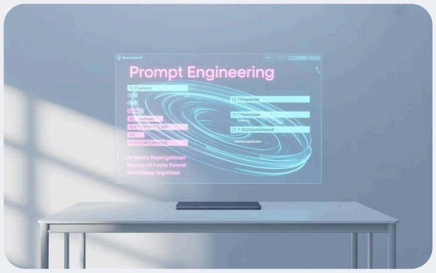
The journey begins with prompt engineering - the art of crafting effective instructions for AI models. You'll learn to:

- Use PromptTemplate for dynamic inputs
- Create endpoints that integrate with LLMs
- Understand LangChain's architecture for LLM apps
- Launch your first LangChain application

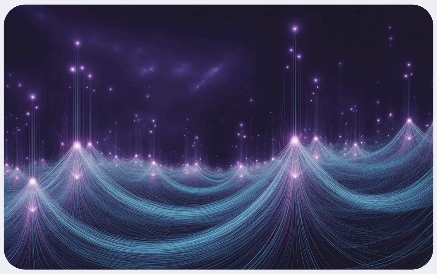
Module 2: Working with Data

Next, you'll learn to connect AI models with various data sources:

- Use LangChain's core abstractions: chains, memory, tools, and agents
- Load and parse data from PDFs, Notion, CSVs, and APIs
- Split large documents for chunked retrieval
- Create and store embeddings for vector similarity search
- Work with vector databases like PGVector, FAISS, Chroma, and Pinecone



Prompt Engineering



Vector Databases

Data Processing

AI Engineering: Advanced Concepts & Productionization

In the final portion of your AI engineering journey, you'll master advanced techniques and learn to build production-ready AI systems that can be deployed to real users.

Retrieval-Augmented Generation

Build systems that can answer questions based on your documents. Master advanced retrieval techniques including metadata filtering, multi-query retrieval, and reranking.

Agents & Tools

Create AI agents that can reason and use external tools like calculators, search engines, and custom functions to accomplish complex tasks.

LangGraph & Multi-Agent Systems

Build sophisticated workflows with branching logic and multiple specialized agents that collaborate to solve problems.

Voice AI Integration

Connect voice interfaces to your AI systems, enabling natural conversation and voice command execution.

The final module focuses on productionization - taking your AI applications from prototypes to production-ready systems:

- **Tracing & Debugging:** Use LangSmith to visualize token usage and trace execution paths through complex chains and agents
- **Deployment:** Build user-facing frontends and deploy your AI applications to production environments
- **Security & Optimization:** Implement best practices for securing API keys, controlling costs, and monitoring performance
- **Capstone Project:** Combine everything you've learned into a comprehensive AI application that showcases your skills

Throughout these advanced modules, you'll work on increasingly sophisticated projects