

AI Software Engineer Professional Certification



Program Syllabus

Overview

The AI Software Engineering Professional Certification takes you from beginner to job-ready full stack engineer. Across 32 weeks of Sunday sessions, you learn how modern applications are architected, how to write clean and scalable code, and how to ship real products with the tools teams actually use. The program covers Python, front end and back-end engineering, relational databases, data structures and algorithms, requirements and product engineering, quality assurance, containerization and deployment, and generative AI engineering.

It is built around three projects that follow one another rather than disconnected exercises. You learn Python by building Contoso, a secure document vault, one layer at a time. You learn the web by building a Tesla interface by hand. You then carry both into Tesla Trip Intelligence, a full stack platform with authentication, route feasibility, charging station search, booking and payment flows, and a conversational layer that answers questions about your own trip history. Alongside these you take part in an **externship** with an industry partner. Your AI coding tools stay switched off for most of the program so that you learn to build by hand first, then you learn to work with an agentic AI workspace as an engineer who specifies and reviews.



Learning Objectives

- Master Python programming fundamentals including OOP, data structures, file I/O, and functional programming patterns.
- Build responsive, accessible web interfaces using HTML5, CSS3, Tailwind CSS, and modern JavaScript (ES6+).
- Develop component-based UIs with React, hooks, state management (Zustand, RTK Query), and Shadcn/ui.
- Create full-stack applications using Next.js with App Router, Server Components, and Server Actions.
- Design and query relational databases using PostgreSQL, advanced SQL, ERDs, and Prisma ORM.
- Build secure REST APIs with Node.js, Express, JWT authentication, and Clerk integration.
- Apply data structures & algorithms knowledge for technical interviews and system design.
- Translate business needs into requirements: use case diagrams, an SRS, and an Agile backlog managed in Jira.
- Produce low-fidelity and high-fidelity interface designs and interactive prototypes in Figma.
- Leverage an agentic AI workspace (Cursor IDE, MCP, hosted and local models) for accelerated development workflows.
- Build AI-powered features: ground a large language model in your application's own data with retrieval augmented generation, let it act through tool calling.

Program Information



Estimated Time

8 months, 17-22h/week*

7h Sunday session

10-15h self-practice



Skill Level

Beginner



Medium

English



Session Delivery

Live Delivered

*Recordings available

Required Hardware/Software

- **Download & Install Visual Studio Code**
 - <https://code.visualstudio.com/download>
- **Download & Install Python 3**
 - <https://www.python.org/downloads/>
- **Download & Install Node.js (LTS)**
 - <https://nodejs.org/en/download/>
- **Download & Install Git**
 - <https://git-scm.com/downloads>
- **Download & Install PostgreSQL**
 - <https://www.postgresql.org/download/>
- **Download & Install Postman**
 - <https://www.postman.com/downloads/>
- **Create GitHub Account**
 - <https://github.com>
- **Create Vercel Account**
 - <https://vercel.com/signup>
- **Download & Install Docker Desktop**
 - <https://www.docker.com/products/docker-desktop/>
- **Create Figma Account**
 - <https://www.figma.com/signup>

(* The length of this program is an estimation of total hours the average student may take to complete all required coursework, including lecture and project time. If you spend about 6-7 hours per week working through the program, you should finish within the time provided. Actual hours may vary)



Software Engineering Foundations

Software Engineering Foundations

In this module, you get your bearings before writing any code. You look at how the products you already use are put together, who does what inside a software company, and how a piece of software travels from an idea to something running in production. By the end of this module, you will be able to describe the shape of a software team, name the stages of the development lifecycle, and explain where your own work will sit in that picture.



Outcomes

By the end of this level, you will understand how a modern application fits together at a conceptual level, what the common engineering roles are and how they collaborate, and how the software development lifecycle structures a project from requirements through to maintenance.

Lesson 1

Professional Skills

(Week 1)

- **Program Orientation** - Understand how the program is structured, how the two weekly sessions work together, and what is expected of you each week.
- **Professional Skills Training** - Take part in an in-person outing built around hands-on exercises in risk analysis, critical thinking, communication, collaboration, and time management, the habits technical teams rely on day to day.

- **Anatomy of a Digital Product** - Explain in plain terms how a marketplace or ride-hailing application works end to end, from the user's screen to the servers behind it.

Lesson 2

How Software Products and Teams Work (Week 2)

- **Roles Inside a Software Company** - Identify what engineers, product managers, designers, QA, and DevOps each own, and how a feature moves between them.
- **The Software Industry Landscape** - Understand the kinds of companies that build software, the products they ship, and where entry-level engineers fit in.

- **SDLC Phases** - Walk through requirements, design, implementation, testing, deployment, and maintenance, and what is produced at each stage.

Lesson 3

SDLC (Week 2)

- **Comparing Delivery Models** - Contrast sequential and iterative approaches, and recognise which situations suit each.
- **Quality and Handover Gates** - Understand why work is reviewed and signed off between phases, and what a definition of done looks like.

Python Fundamentals



Python Fundamentals

In this module, you will build essential programming literacy through Python, one of the most versatile and beginner-friendly languages in software development. You will understand the fundamental building blocks of programming including variables, control flow, functions, and object-oriented concepts. The emphasis is on programmatic thinking rather than memorising syntax: given a scenario, you should be able to say how you would model it before you write a line. By the end of this module, you will have a working Python application you built yourself, one layer per session.



Module Project

Contoso Secure Document Vault

A command-line application where employees of a fictional company store and protect their own documents. Users create an account, log in, write documents marked internal or confidential, and encrypt them so that the saved content cannot be read without the right key. Each employee can only ever see their own documents.

You start with a simple cipher, then add file storage, accounts and passwords, and finally restructure it all with object-oriented programming.



- **Python Syntax, Data Types & Variables** - Understand Python's readable syntax, variables, strings, numbers, booleans, and type conversion fundamentals.
- **Control Structures** - Master if/else conditional statements, for loops, while loops, and nested control flow for decision-making in programs.

Lesson 1

Python Core (Week 3-5)

- **Functions and Scope** - Learn how to define functions, pass parameters, return values, and understand variable scope and namespace concepts.
- **List Comprehensions & Generators** - Write concise, Pythonic code using list comprehensions and memory-efficient generators for data processing.
- **Built-in Functions & Lambda Expressions** - Master built-in methods and functional programming with lambda functions, map(), filter(), and reduce().
- **Program Loops and Menus** - Drive a long-running terminal program from a loop, branch on user input, and keep the program alive until the user chooses to end it.

Milestone 1: The Protected Document Prototype - Implement the Caesar transformation as a set of functions, reduce any whole-number key to a shift, preserve case and punctuation, and verify that encrypting then decrypting returns the original text exactly. Wrap it in a menu loop that creates and reads documents held in memory, and enforce a three-attempt limit on the key that ends the operation and never the program.

Lesson 2

Python Data Structures & Advanced Features (Week 6-7)

- **Lists, Tuples, Sets & Dictionaries** - Deep dive into Python's built-in data structures, their use cases, performance characteristics, and common operations.
- **File I/O Operations** - Learn to read, write, and manipulate files including text, CSV, and JSON formats with proper resource management.
- **Working with JSON** - Serialise Python data structures to JSON-formatted text and read them back, and handle a data file that does not exist yet.
- **Hashing with hashlib** - Produce a one-way SHA-256 hash of a value, store the hash instead of the value, and verify by comparing hashes.
- **Exception Handling & Debugging** - Master try/except blocks, custom exceptions, error handling patterns, and debugging techniques for robust code.
- **Searching and Filtering Collections** - Find records by exact key and by case-insensitive partial match, and present multiple matches for selection.

Milestone 2: The Persistent Vault - Extend the prototype so employees and documents survive a restart: save to JSON-formatted data files after every change, load them at startup, and tolerate a missing file on a first run. Add account creation and login with passwords stored only as SHA-256 hashes, document ID and title search, and error handling that keeps the program alive through an invalid menu choice, a non-numeric key, or a number out of range.

- **Classes & Objects** - Understand object-oriented principles, class definitions, object instantiation, and the relationship between classes and instances.
- **Constructors & Instance Methods** - Learn init methods, self parameter, instance attributes, and defining behaviors through methods.

Lesson 3

Python Object-Oriented Programming (Week 8-9)

- **Property Decorators & Getters/Setters** - Implement encapsulation using @property decorator, controlled attribute access, and validation logic.
- **Class Methods & Static Methods** - Differentiate between instance, class (@classmethod), and static (@staticmethod) methods and their appropriate use cases.
- **Serialising Objects** - Convert objects to dictionaries for saving and rebuild them from dictionaries on load, and keep that conversion in one place.
- **Enforcing an Invariant Across a Program** - Apply a single access rule consistently to every operation, and recognise where a new code path could bypass it.

Milestone 3: The Object-Oriented Vault - Refactor the vault into Employee and Document classes with constructors, instance methods, and controlled attribute access through property decorators. Convert objects to dictionaries to persist them and rebuild them on load. Route every list, read, search and delete through an ownership check, and prove with two accounts that one employee cannot reach the other's documents.

Front-End Web Development Foundations



CSS

JS

HTML

<button>

Front-End Web Development Foundations

In this module, you will learn how to build responsive, accessible user interfaces for the modern web. You will understand how HTML structures content, how CSS and Tailwind control layout and styling, and how JavaScript makes a page respond to the person using it. The emphasis is on building by hand, with no framework and no template, so that you understand what is happening underneath before a tool starts doing it for you. By the end of this module, you will have a complete, interactive website you built yourself, one section per session.



Module Project

Tesla Interface Build

A product website for an electric car, built from an empty file. You create the page structure, the layout that adapts from phone to desktop, the visual styling, the animations, and a car configurator where a visitor's choices update what they see and the running price.

You start with semantic HTML and CSS, then add Tailwind for faster styling, and finally bring the page to life with JavaScript. The interface you build here is the starting point for Tesla Trip Intelligence later in the program.



Lesson 1

HTML5 & CSS3 Mastery (Week 3-6)

- **Semantic HTML5 & Accessibility** - Structure content meaningfully using semantic elements (header, nav, main, article, section) and ARIA attributes for screen readers.
- **CSS Fundamentals & Box Model** - Master selectors, specificity, the box model (margin, padding, border), and how browsers render elements.
- **Flexbox & CSS Grid Layouts** - Create flexible one dimensional layouts with Flexbox and complex two dimensional layouts with CSS Grid.
- **Responsive Design & Media Queries** - Implement mobile-first responsive designs using breakpoints and media queries for all screen sizes.
- **CSS Animations & Transitions** - Create smooth transitions, keyframe animations, and micro- interactions for enhanced user experience.
- **Tailwind CSS Introduction** - Learn the utility-first CSS approach, Tailwind's class naming conventions, and the rapid UI development workflow.
- **Tailwind Configuration & Customization** - Customize themes, colors, spacing, and extend Tailwind with custom utilities and components.

Milestone 1: The Static Product Page - Build the full page in semantic HTML and CSS: navigation, hero, feature sections, gallery and footer, laid out with Flexbox and Grid, responsive from phone to desktop, with transitions on the elements a visitor interacts with. Finally restyle the page with Tailwind.

Lesson 2

JavaScript Fundamentals (Week 7-9)

- **JavaScript Basics & Data Types** - Understand variables (let, const), primitive types, type coercion, and JavaScript's dynamic typing system.
- **Functions, Scope & Closures** - Master function declarations, expressions, arrow functions, lexical scope, and practical closure patterns.
- **DOM Manipulation & Events** - Select, modify, and create DOM elements; handle user events (click, submit, keypress) with event listeners.
- **ES6+ Features** - Use modern JavaScript features including arrow functions, destructuring, spread operator, template literals, and optional chaining.
- **Promises & Async/Await** - Handle asynchronous operations using Promises, Promise chaining, async/await syntax, and error handling.
- **Array Methods & Functional Programming** - Master map(), filter(), reduce(), find(), and other array methods for data transformation.
- **Modules & Browser APIs** - Organize code with ES6 modules (import/export) and work with localStorage, fetch API, and other Web APIs.

Milestone 2: The Interactive Configurator - Bring the page to life with JavaScript: a mobile navigation menu, and a car configurator where each selection updates the image, the specification and the running price.

Evaluation Week (Week 10)

Contoso Extension Sprint Submission

You receive a written specification describing new features for the vault and implement two or three of them on your own. You then walk a mentor through your code in a one-to-one review.

- Two or three new features, built against the specification
- Every existing behaviour still working
- A public GitHub repository with your own commit history
- A README explaining how to run the program and what you added
- A live demonstration and code walkthrough with a mentor

Tesla Interface Submission

You submit the completed product website, extended with the additional features set for the evaluation. You then demonstrate it and explain how you built it in a one-to-one review.

- A responsive page that works from phone to desktop
- The interactive car configurator, with selections updating the image and running price
- Clean, semantic HTML and consistent styling
- A public GitHub repository with your own commit history
- A live demonstration and code walkthrough with a mentor



Product Design & Requirements Engineering

Product Design & Requirements Engineering

In this module, you step back from writing code and learn how the work gets decided in the first place. You look at how requirements are drawn out of stakeholders and recorded in a form a team can build against, how a backlog is shaped and moved through a sprint, and how an interface is designed and tested as a prototype before anyone commits to building it. By the end of this module, you will be able to take a loosely described product idea and turn it into documented requirements, a planned backlog, and a clickable prototype.



Module Project

Tesla Trip Intelligence: Product Blueprint

The plan for the platform you build across the rest of the program. Starting from a loosely described product idea, you work out who uses it and what they need, write the requirements, organise the work into a backlog, and design the screens as a clickable prototype.

You start with use case diagrams and an Agile board, then design the interface in Figma from wireframe to high fidelity, and finally write the requirements specification. The blueprint you produce here is what you build against in the full stack modules that follow.



Lesson 1

Use Case Modelling & Agile with Jira (Week 11)

- **Use Case Diagrams** - Identify actors, use cases, and system boundaries, and model how users interact with a system.
- **Use Case Relationships** - Use include, extend and generalisation to show shared and optional behaviour, and keep a diagram readable as the system grows.
- **Use Case Descriptions** - Write out the main flow, the alternative flows and the failure cases for a use case, so the diagram is backed by detail a developer can build from.
- **Agile Fundamentals** - Understand sprints, backlogs, user stories, and ceremonies, and how a team plans and reviews its work.
- **From Use Cases to Backlog Items** - Break each use case into epics and stories small enough to build in a sprint, so the backlog comes from the model and not from guesswork.
- **Prioritisation and Estimation** - Rank a backlog by value and effort, estimate work with story points, and decide what fits in a sprint.
- **Working in Jira** - Create and refine issues, organise a backlog, run a board through a sprint, and track progress.

Milestone 1: Use Cases and Backlog - Identify the users of Tesla Trip Intelligence and what each one needs to do, draw the use case diagram, and turn it into a prioritised backlog of user stories on a Jira board, with the first sprint planned.

Lesson 2

Interface Design with Figma (Week 11)

- **User Flows** - Map the path a user takes through the product, screen by screen, before drawing any of the screens.
- **Low-Fidelity Wireframing** - Sketch layouts and user flows quickly to test structure before investing in visual detail.
- **High-Fidelity Design** - Build polished screens with type scales, colour, spacing, and reusable components.
- **Styles and Variables** - Define colour, type and spacing once as shared styles and variables, so a change in one place updates every screen.
- **Auto Layout** - Construct designs that resize and reflow predictably, mirroring how they will behave in code.
- **Components and Variants** - Build a component once, define its states, and reuse it across screens so the design stays consistent as it grows.
- **Interactive Prototyping** - Link screens into a clickable flow that can be walked through and tested before any code is written
- **Developer Handoff** - Prepare a design for the person who will build it: named layers, inspectable spacing and values, and exported assets.

Milestone 2: The Clickable Prototype - Design the core screens in Figma, from low-fidelity wireframes to high-fidelity designs built with reusable components and Auto Layout, and link them into a prototype that can be clicked through from planning a trip to booking a charging stop.

Lesson 3

Requirements & Product Engineering

(Week 12)

- **Eliciting Requirements** - Run stakeholder conversations that surface what a system actually has to do, and separate a stated request from the need behind it.
- **The Business Requirements Document** - Capture the business goal, the people affected and the measures of success in plain language, before any system detail is decided.
- **Functional and Non-Functional Requirements** - Distinguish what the system must do from how well it must do it, and write both so they can be tested.
- **Writing an SRS** - Record scope, actors, requirements, and constraints in a software requirements specification a team can build against.
- **User Stories and Acceptance Criteria** - Break requirements into stories with conditions that make "done" unambiguous.
- **Reviewing and Validating Requirements** - Walk the specification through with stakeholders, find the ambiguous and conflicting requirements, and get sign-off before work begins.
- **Scope and Change Control** - Recognise scope creep, and handle a change request without destabilising a plan the team has already agreed.

Milestone 3: The Requirements Specification - Write the SRS for Tesla Trip Intelligence: scope, users, functional and non-functional requirements, and user stories with acceptance criteria, each one traceable to a backlog item and a screen in your prototype.

Front End Development

The background is a complex, glowing digital interface in shades of blue and purple. It features various UI elements: a map with a red location pin and a blue path, a dashboard with multiple charts and graphs, a sidebar with a cube icon, and a main content area with a list of items and a search bar. The overall aesthetic is high-tech and futuristic, with a grid-like pattern and glowing lines connecting different components.

Front-end Engineering with React Ecosystem

In this module, you move from hand-written DOM code to React, the most widely used JavaScript library for building user interfaces. You will understand component architecture, state management, hooks, and modern React patterns, along with the libraries the industry builds a front end from. The emphasis is on understanding what the framework is doing for you, because you have already done it by hand. By the end of this module, you will have the complete front end of Tesla Trip Intelligence, built and deployed yourself, one feature per session.



Module Project

Tesla Trip Intelligence: Front End

The interface of the platform you planned in Level 4. A driver signs in, manages their vehicles, plans a trip, sees whether the battery will get them there, and finds charging stops on a map. You build every screen from your own Figma designs.

You start by rebuilding your Tesla interface as React components, then add a design system, navigation, forms and live data, and finally move the application to Next.js and deploy it.



Lesson 1

React Fundamentals (Week 13-15)

- **Component Architecture & JSX** - Understand React's component model, JSX syntax, component composition, and the virtual DOM rendering process.
- **Props & State Management** - Pass data between components with props, manage local state, and understand unidirectional data flow.
- **useState & useEffect Hooks** - Manage component state with useState and handle side effects (API calls, subscriptions) with useEffect.
- **Event Handling in React** - Handle user interactions with synthetic events, form submissions, and controlled components.
- **Conditional Rendering & Lists** - Render content conditionally and efficiently display lists with proper key management.
- **Thinking in Components** - Break a design into a component tree, and decide where each piece of state should live.
- **From DOM Code to Components** - Rebuild a section of your hand-written Tesla interface as components, and account for exactly what React removed.

Milestone 1: Component Foundations - Rebuild your Tesla interface as React components: vehicle cards, a vehicle list rendered from data, and a trip entry form driven by state, props and controlled inputs.

Lesson 2

Modern UI with Shadcn/ui (Week 16)

- **Shadcn/ui Setup & Architecture** - Install and configure Shadcn/ui, understand its component architecture and the relationship with Radix UI.
- **Radix UI Primitives Understanding** - Learn how Radix primitives provide accessible, unstyled components as building blocks for custom design systems.
- **Component Customization with Tailwind** - Customize Shadcn/ui components using Tailwind utility classes for brand-consistent styling.
- **Building Reusable Component Library** - Create a custom component library with variants, extending base components for project needs.
- **Dark Mode Implementation** - Implement theme switching and dark mode support using CSS variables and Tailwind's dark mode utilities.
- **From Figma to Components** - Translate the styles, variables and components in your Figma file into a theme and a matching set of coded components.
- **Accessible Forms and Dialogs** - Build inputs, dialogs and menus that work with a keyboard and a screen reader.

Milestone 2: The Product Design System - Set up Shadcn/ui with a custom theme taken from your Figma designs, and build the application shell for Tesla Trip Intelligence: navigation, page layout, dialogs, and light and dark modes.

Lesson 3

Advanced React Patterns (Week 17-18)

- **Custom Hooks Creation** - Extract reusable logic into custom hooks for data fetching, form handling, and local storage.
- **Context API for State Management** - Share state across component trees without prop drilling using React Context.
- **useReducer for Complex State** - Manage complex state logic with useReducer for predictable state transitions.
- **React Router v6 for Navigation** - Implement client-side routing, nested routes, protected routes, and navigation patterns.
- **Code Splitting & Lazy Loading** - Optimize bundle size with dynamic imports, React.lazy(), and Suspense boundaries.
- **Component Composition Patterns** - Use children and compound components to build flexible components without passing props through every layer.
- **Refs and Third-Party Libraries** - Use refs to reach the DOM directly, and wrap a non-React library such as a map inside a component.

Milestone 3: Navigation and Shared State - Add routing for the dashboard, vehicles, trip planner and bookings views, protect the signed-in routes, and move shared logic into custom hooks and context.

Lesson 4

Forms & Data Fetching (Week 19)

- **React Hook Form with Zod Validation** - Build performant forms with React Hook Form and schema-based validation using Zod.
- **TanStack Query for Server State** - Manage server state, caching, background updates, and data synchronization with TanStack Query.
- **Optimistic Updates** - Implement optimistic UI updates for instant feedback while awaiting server confirmation.
- **Error Boundaries** - Handle errors gracefully with error boundaries to prevent entire app crashes.
- **Loading States & Skeletons** - Create polished loading experiences with skeleton loaders and progressive content rendering.
- **Working with a Mock API** - Develop against a mocked backend, so the front end can be built and tested before the real API exists.
- **Pagination and Filtering** - Fetch, page and filter long lists such as trip history without loading everything at once.

Milestone 4: Forms and Data - Build the vehicle registration and trip planning forms with schema validation, fetch and cache data from a mock API, and handle loading, error and optimistic states.

Lesson 5

State Management (Week 20-21)

- **Zustand for Global State** - Implement lightweight global state management with Zustand's simple API and persist middleware.
- **RTK & RTK Query Deep Dive** - Use Redux Toolkit for complex state and RTK Query for powerful data fetching with caching.
- **State Persistence Strategies** - Persist state to localStorage, handle hydration, and manage state across sessions.
- **Performance Optimization** - Use React.memo, useMemo, useCallback for optimal rendering performance.
- **Choosing the Right State Tool** - Tell apart local, shared, server and global state, and pick the simplest tool that fits each.
- **Derived State and Selectors** - Compute values such as remaining range from existing state, and avoid storing the same fact twice.
- **Integrating a Map** - Render routes and charging station markers from a mapping provider, and keep the map in step with application state.

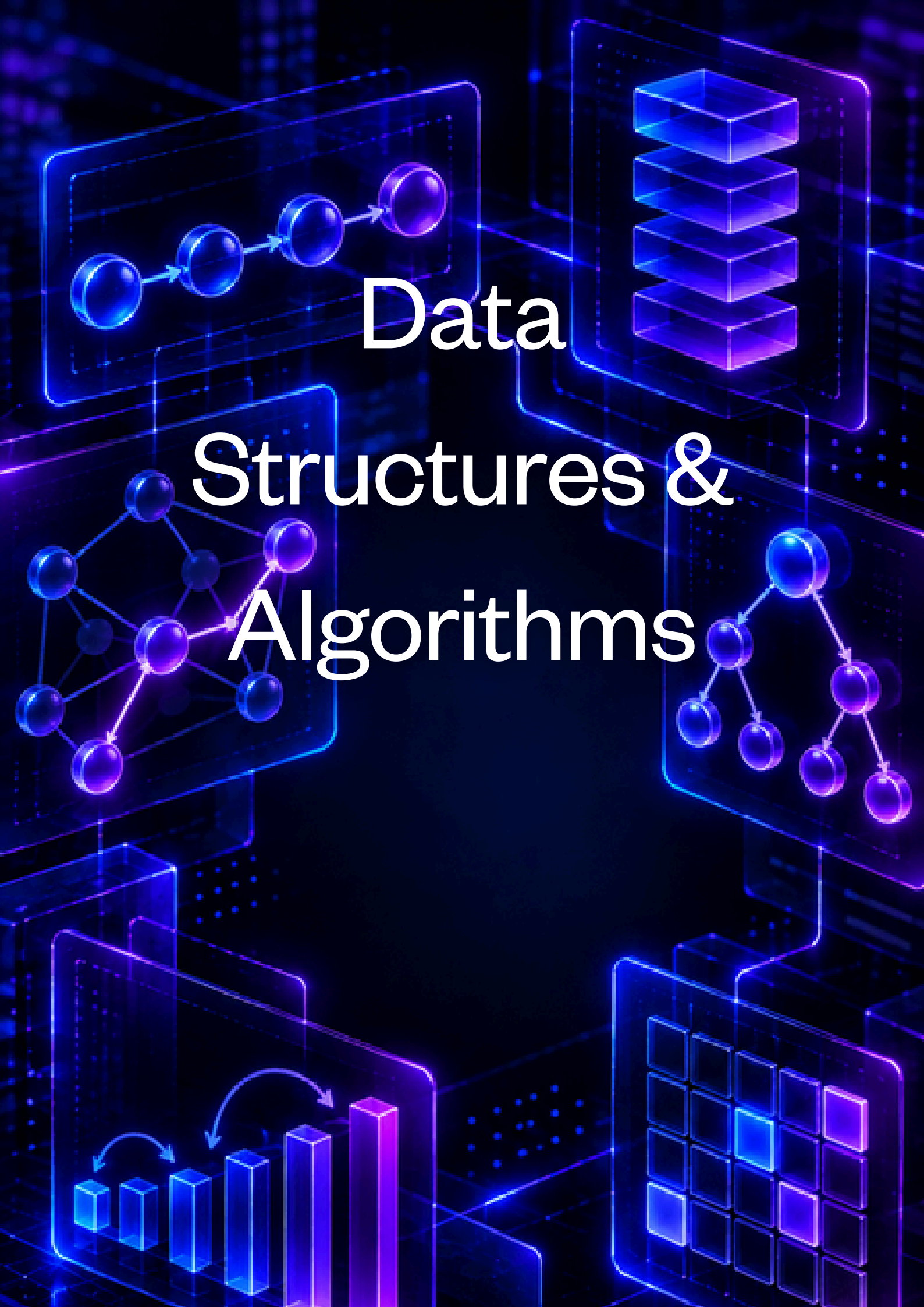
Milestone 5: Trip Planning Interface - Complete the vehicle selection, route entry and trip feasibility views, with global state that persists across sessions and a map view showing the route and the charging stations along it.

Lesson 6

Next.js Framework (Weeks 22-23)

- **App Router Architecture** - Understand Next.js App Router, file-based routing, layouts, and the new mental model.
- **Server & Client Components** - Differentiate server and client components, choose the right rendering strategy, and optimize bundle size.
- **Server Actions** - Handle form submissions and mutations directly with Server Actions without API routes.
- **API Routes** - Build API endpoints within Next.js for backend functionality and data handling.
- **Static & Dynamic Rendering** - Implement ISR (Incremental Static Regeneration), dynamic routes, and hybrid rendering strategies.
- **SEO Optimization** - Configure metadata, Open Graph tags, structured data, and sitemap generation for search engine visibility.
- **Deployment Strategies** - Deploy to Vercel, configure environment variables, set up preview deployments, and monitoring.

Milestone 6: The Deployed Front End - Move the application to Next.js with the App Router, choose server or client rendering for each view, and deploy to Vercel with environment configuration and preview deployments.



Data Structures & Algorithms

Data Structures and Algorithms

In this module, you will develop the problem-solving skills that technical interviews test and that efficient software depends on. You will understand fundamental data structures, algorithm patterns, and complexity analysis. The emphasis is on reasoning out loud about a trade-off, because an interviewer wants to hear how you think and not a memorised answer. By the end of this module, you will have a set of solved problems that you wrote yourself and can explain line by line.



Module Project

Interview Problem Portfolio

A public repository of problems you have solved, each with a short written explanation of your approach and its time and space complexity. It is what you revise from before a technical interview, and what you talk through when a company asks how you would solve a problem.

You start with the core data structures, then move to searching, sorting and recursion, and finish with dynamic programming and graphs.



Lesson 1

Core Data Structures (Week 13-14)

- **Arrays & Strings Manipulation** - Master array operations, string algorithms, and in-place manipulation techniques.
- **Linked Lists & Stacks** - Implement singly/doubly linked lists, stack data structure, understand traversal and common patterns.
- **Queues & Dequeues** - Build queue and double-ended queue structures, understand FIFO principles, and practical applications.
- **Hash Tables & Sets** - Implement hashing, handle collisions, use Sets for uniqueness, and solve problems efficiently.
- **Time and Space Complexity** - Analyse an algorithm in Big O terms, and state the cost of each operation on each structure.
- **Two Pointers and Sliding Window** - Apply the two most common array patterns to problems that would otherwise need nested loops.
- **Choosing a Data Structure** - Compare structures for a given problem, and explain the trade-off you are accepting.

Milestone 1: The Data Structures Set - Implement a linked list, stack, queue and hash table from scratch, and solve a set of problems with each, recording the approach and complexity of every solution in your portfolio.

Lesson 2

Algorithms & Problem Solving (Weeks 15-16)

- **Searching Algorithms** - Implement linear search, binary search, and understand when to apply each approach.
- **Sorting Algorithms** - Implement bubble, selection, insertion, merge sort, quicksort, and understand their trade-offs.
- **Recursion & Backtracking** - Solve problems recursively, understand call stack, and implement backtracking solutions.
- **Divide and Conquer** - Split a problem into smaller copies of itself, solve them, and combine the results.
- **A Problem-Solving Method** - Restate the problem, work an example by hand, write the simple solution first, then improve it.
- **Analysing Recursive Solutions** - Work out the time and space cost of a recursive function, including the call stack.
- **Explaining Your Solution** - Talk through your reasoning while you code, the way a technical interview expects.

Milestone 2: The Algorithms Set - Implement binary search, merge sort and quicksort, solve a set of recursion and backtracking problems, and add each to your portfolio with a comparison of the simple and the improved solution.

Lesson 3

Dynamic Programming & Graphs (Week 17)

- **Dynamic Programming Basics** - Understand memoization and tabulation, and solve classic DP problems step by step.
- **Graph Representations** - Represent graphs with adjacency lists and matrices and choose between them.
- **Graph Traversals** - Implement breadth-first and depth-first search and apply them to connectivity and shortest-path problems.
- **Trees and Tree Traversals** - Traverse and search binary trees, and recognise the problems that are trees in disguise.
- **Shortest Path on Weighted Graphs** - Find the cheapest route between two points when each step has a different cost.
- **Recognising the Pattern** - Read a new problem and identify which technique it calls for before writing any code.

Milestone 3: The Advanced Set - Solve a set of dynamic programming, tree and graph problems, including a shortest-path problem, and complete a timed mock interview in which you solve and explain one unseen problem.



SQL & Database Design

SELECT

SQL & Database Design

In this module, you will master relational database concepts and PostgreSQL, the open-source database system used by companies worldwide. You will understand how to design efficient database schemas, write complex queries, and optimize database performance. The emphasis is on design decisions, because a schema is far harder to change later than the code around it. By the end of this module, you will have designed and built the database that Tesla Trip Intelligence runs on.

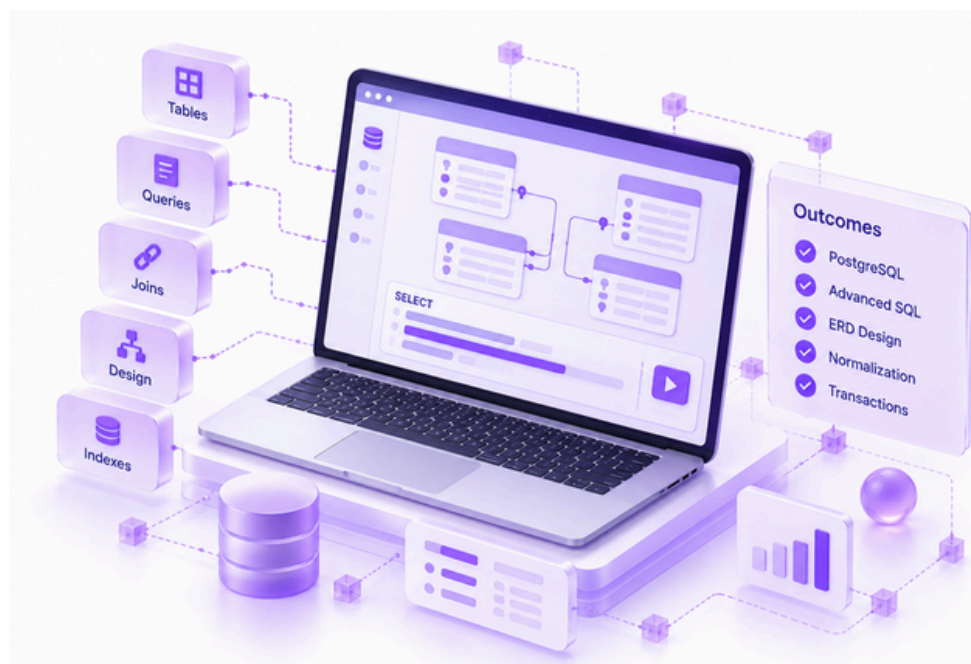


Module Project

Tesla Trip Intelligence: Database

The data behind the platform. Drivers, vehicles, trips, charging stations, bookings, charging sessions and payments, stored so that the application can answer questions quickly and can never record two bookings for the same slot.

You start by creating tables and writing queries, then write the reports behind a driver's trip history, and finally design the full schema and make it safe when many drivers use it at once.



Lesson 1

SQL Fundamentals (Week 18)

- **Introduction to PostgreSQL** - Set up PostgreSQL, understand relational database concepts, and navigate pgAdmin/psql interfaces.
- **Basic SQL Syntax & CRUD Operations** - Write CREATE, READ, UPDATE, DELETE operations and understand SQL statement structure.
- **SELECT Queries** - Filter with WHERE, sort with ORDER BY, limit results with LIMIT, and combine multiple conditions.
- **Data Types & Constraints** - Choose appropriate data types, implement NOT NULL, UNIQUE, CHECK, and DEFAULT constraints.
- **Primary Keys & Foreign Keys** - Design effective primary keys, establish relationships between tables, and understand referential integrity.
- **Pattern Matching and NULLs** - Filter with LIKE, IN and BETWEEN, and handle missing values correctly.
- **Changing a Table Safely** - Add, rename and remove columns on a table that already holds data.

Milestone 1: First Tables - Create the driver, vehicle and charging station tables with appropriate types, constraints and keys, and write the create, read, update and delete queries the application will need.

Lesson 2

Advanced SQL (Week 19-22)

- **Primary Keys** - Design effective primary keys, understand surrogate vs. natural keys, and implement auto-incrementing IDs.
- **Foreign Keys** - Establish relationships between tables, understand referential integrity, and cascade options.
- **JOINS (INNER, LEFT, RIGHT, FULL)** - Combine data from multiple tables using different join types for various query needs.
- **Aggregate Functions** - Use COUNT, SUM, AVG, MIN, MAX for data analysis and reporting queries.
- **GROUP BY & HAVING Clauses** - Group results, filter groups, and create summary reports from large datasets.
- **Subqueries and CTEs** - Build a query out of other queries, and use common table expressions to keep complex logic readable.
- **Window Functions** - Compute running totals, rankings and row-to-row comparisons without collapsing the result.

Milestone 2: Trip and Spend Reports - Write the queries behind a driver's history: trips per vehicle, energy and cost totals by month, and rankings and running totals using joins, aggregates and window functions.

Lesson 3

Database Design (Weeks 20-21)

- **ERD (Entity Relationship Diagrams)** - Design database schemas visually, identify entities, attributes, and relationships.
- **Database Normalization (1NF, 2NF, 3NF, BCNF)** - Apply normalization rules to eliminate data redundancy and ensure data integrity.
- **Denormalization Strategies** - Know when and how to denormalize for read performance in production systems.
- **Indexes & Query Optimization** - Create indexes, analyze query plans with EXPLAIN, and optimize slow queries.
- **ACID Properties & Transactions** - Understand Atomicity, Consistency, Isolation, Durability and implement transactions.
- **From Requirements to Schema** - Read a requirements specification and derive the entities, relationships and rules the database must hold.
- **Modelling State and History** - Represent a lifecycle such as a booking as a current state plus a record of events, and decide what is stored and what is calculated.

Milestone 3: The Complete Schema - Design the full ERD for Tesla Trip Intelligence from your requirements specification, adding trips, availability, bookings, charging sessions and payments. Normalise it, and index the paths the application actually queries.

Lesson 4

Advanced Database Concepts (Week 22)

- **Stored Procedures** - Write reusable database logic with stored procedures and user-defined functions for complex operations.
- **Triggers & Functions** - Automate database actions with triggers for audit logs, data validation, and derived data.
- **Database Security Best Practices** - Implement user permissions, role-based access, and secure connection practices.
- **Backup & Recovery Strategies** - Understand backup methods, point-in-time recovery, and disaster recovery planning.
- **Concurrency and Locking** - Understand what happens when two transactions touch the same row, and prevent conflicting writes.
- **Views** - Save a complex query as a view, so reports and dashboards read from one trusted definition.

Milestone 4: Safe Under Concurrency - Add a transaction that makes a double booking impossible, an audit trigger that records every booking change, and role-based permissions separating drivers from administrators.

AI-First Development Workshop

(Week 23)

AI Tools for Developers

- **GitHub Copilot Mastery** - Leverage AI pair programming for code completion, generation, and learning new patterns.
- **Cursor IDE Advanced Features** - Use Cursor's AI capabilities for intelligent code editing, refactoring, and exploration.
- **Claude/ChatGPT for Code Generation** - Generate boilerplate, debug issues, and explain complex code with LLMs.
- **Prompt Engineering for Coding** - Write effective prompts to get better code suggestions and explanations.

AI-Powered Development Workflow

- **Code Review with AI** - Use AI to review code for bugs, security issues, and best practice violations.
- **Documentation Generation** - Auto-generate documentation, comments, and README files with AI assistance.
- **Test Case Generation** - Generate unit tests, edge cases, and test data using AI tools.
- **Debugging with AI Assistance** - Diagnose bugs faster by explaining errors to AI and getting solution suggestions.

Backend Development



Node.js & Backend Development

In this module, you will learn to build robust, scalable backend systems using Node.js and Express. You will understand server architecture, REST API design, database integration, authentication, and how to work with outside services. The emphasis is on the workflows every commercial product needs: who you are, what you are allowed to touch, what you have booked, and what you paid. By the end of this module, you will have the working API behind Tesla Trip Intelligence, built yourself, one feature per session.

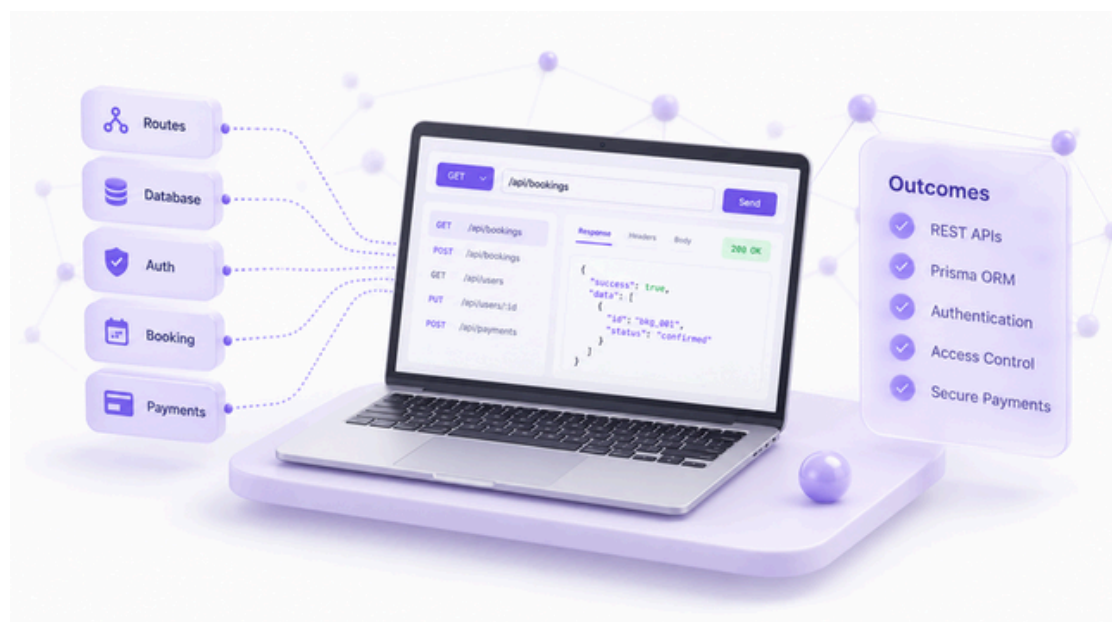


Module Project

Tesla Trip Intelligence: Backend API

The server behind the platform. It registers drivers and their vehicles, works out whether a trip is possible, finds charging stations through a mapping service, takes bookings, and handles payment.

You start with an Express API and a connection to a mapping service, then connect your database and build the booking flow, and finally add authentication, access control and secure payments.



Lesson 1

Node.js & Express Fundamentals (Weeks 24-25)

- **Node.js Architecture & Event Loop** - Understand Node.js runtime, non-blocking I/O, event loop, and when to use Node.js.
- **Building REST APIs with Express** - Create Express applications, define routes, and structure API endpoints following REST principles.
- **Middleware Concepts** - Implement request processing pipeline with built-in, third-party, and custom middleware functions.
- **Route Handling & Parameters** - Handle path parameters, query strings, request bodies, and implement route organization.
- **Error Handling Patterns** - Implement centralized error handling, custom error classes, and async error wrappers.
- **Request Validation** - Check every incoming request against a schema before it reaches your logic.
- **Consuming Third-Party APIs** - Call a mapping service for routes and charging stations, keep it behind a service layer, and handle slow responses and rate limits.

Milestone 1: The API Skeleton - Stand up the Express application with vehicle and trip routes, request validation, centralised error handling, and a charging station search that calls the mapping service.

Lesson 2

Database Integration (Weeks 25-26)

- **Prisma ORM Setup & Models** - Initialize Prisma, define schema with models, relations, and field attributes.
- **Database Migrations with Prisma** - Create and manage database migrations for schema evolution and team collaboration.
- **Seeding Data** - Write seed scripts for development data, testing fixtures, and initial production data.
- **Relationships & Queries** - Query related data, implement filtering, pagination, and complex Prisma operations.
- **Connection Management** - Handle database connection pooling, reconnection strategies, and production configuration.
- **Transactions in Prisma** - Run several database operations as one unit, so a failure part-way through leaves nothing half-done.
- **Building a Booking Flow** - Implement availability, hold, confirm and cancel as endpoints over the booking states in your schema.

Milestone 2: The Data Layer - Connect the API to PostgreSQL through Prisma: models and migrations for your schema, seed data for vehicles and charging stations, filtered and paginated queries, and a booking flow that cannot double-book a slot.

Lesson 3

Authentication & Security (Week 26-27)

- **JWT Concepts & Implementation** - Understand JSON Web Tokens, implement token generation, verification, and refresh patterns.
- **Clerk Integration for Authentication** - Integrate Clerk for production-ready authentication with social logins and user management.
- **Session Management** - Compare session-based vs. token-based auth, implement secure session handling.
- **Role-Based Access Control (RBAC)** - Implement authorization with roles, permissions, and protected routes.
- **API Security Best Practices** - Protect APIs with rate limiting, input validation, sanitization, and security headers.
- **Ownership Checks** - Make sure every vehicle, trip and booking can only be reached by the driver it belongs to.
- **Secure Payment Integration** - Connect a payment gateway for bookings, verify its confirmation before trusting it, and keep the booking and the payment consistent when one fails.

Milestone 3: The Trip Intelligence API - Complete the backend: authenticated vehicle and trip management, ownership enforced on every record, administrator routes protected by role, payment for bookings with verified confirmation, and a Postman collection demonstrating every flow.

Quality Assurance & Testing

(Weeks 28-29)

Testing Fundamentals

- **Testing Importance** - Understand why testing is critical for software quality, reliability, and maintainability.
- **Types & Concepts** - Learn Unit Testing, Integration Testing, and End-to-End (E2E) testing methodologies and when to apply each.
- **Testing Tools (Playwright/Cypress)** - Set up and configure modern testing frameworks for comprehensive test coverage.
- **Frontend Testing** - Write component tests, integration tests, and E2E tests for React/Next.js applications.
- **Backend Testing** - Test API endpoints, database operations, and authentication flows with proper mocking strategies.

DevOps & Containerization

The background is a vibrant, futuristic digital landscape. It features a grid-like floor with glowing blue and purple lines. In the foreground, there are several glowing blue cubes, some stacked on top of each other. To the left, a stack of three cubes is connected by a glowing blue arrow to a square icon containing a gear. Further right, another square icon with a checkmark is connected by a glowing blue arrow to a small cube. In the background, there are several glowing blue rectangles, some of which are connected by glowing blue lines. On the right side, a glowing blue rocket is launching upwards, leaving a trail of glowing blue smoke. The overall scene is illuminated with a strong blue and purple glow, creating a high-tech, digital atmosphere.

DevOps & Containerization

In this module, you learn to package an application so it runs the same way everywhere, and to automate the path from a commit to a running deployment. You will understand how containers isolate an application, how an orchestrator keeps it running, and how a pipeline tests and ships each change. The emphasis is on repeatability, because a release that depends on one person's laptop is a release waiting to fail. By the end of this module, you will have Tesla Trip Intelligence running live, deployed by a pipeline you built yourself.



Module Project

Tesla Trip Intelligence: Deployment

The platform, live on the internet. The front end, the API and the database run as separate services, and every change you push is tested and deployed automatically.

You start by putting each part of the application in a container and running them together, then build the pipeline that takes your code from a commit to production.



Lesson 1

Containerization with Docker & Kubernetes (Weeks 28-29)

- **Containers vs. Virtual Machines** - Understand what a container is, what problem it solves, and how it differs from a virtual machine.
- **Images & Dockerfiles** - Write Dockerfiles, build images in layers, and keep them small and reproducible.
- **Multi-Container Applications** - Run an application, a database, and supporting services together with defined networks and volumes.
- **Kubernetes Fundamentals** - Understand pods, deployments, and services, and how an orchestrator keeps workloads running.
- **Scaling & Configuration** - Manage replicas, environment configuration, and secrets in a cluster.
- **Docker Compose for Development** - Start the whole application on any machine with one command.
- **Health Checks and Logs** - Tell whether a running service is healthy, and find out why when it is not.

Milestone 1: The Containerised Platform - Write Dockerfiles for the front end and the API, and run them with PostgreSQL through Docker Compose, with health checks and configuration kept out of the images.

Lesson 2

Git Workflows & Deployment (Week 30)

- **Git Workflows & Branching Strategies** - Implement GitFlow, feature branches, pull request workflows, and collaborative development.
- **GitHub Actions for CI/CD** - Automate testing, linting, and deployment with GitHub Actions workflows.
- **Environment Variables & Secrets** - Manage secrets securely in CI/CD pipelines and deployment platforms.
- **Vercel Deployment for Frontend** - Deploy frontend applications to Vercel with automatic preview deployments.
- **Railway/Render for Backend** - Deploy backend services to Railway or Render with database provisioning and scaling.
- **Pull Request Reviews** - Protect the main branch, require a passing pipeline and a review, and write a pull request someone can actually review.
- **Monitoring and Rollback** - Watch a release after it goes live, and return to the previous version when something breaks.

Milestone 2: The Automated Pipeline - Build a GitHub Actions pipeline that runs your tests on every pull request and deploys the front end to Vercel and the API to Railway or Render on merge, with secrets managed outside the repository.

AI Engineering



AI Engineering

In this module, you learn to build features on top of large language models rather than only use them. You will understand what a model can do out of the box, how to ground it in your own application's data, and when adapting the model itself is the right answer. The emphasis is on answers you can trust, because a model that sounds confident and is wrong is worse than no feature at all. By the end of this module, you will have added a working AI assistant to Tesla Trip Intelligence, built yourself, one capability per session.



Module Project

Tesla Trip Intelligence: Conversational Assistant

A chat interface on top of your platform. A driver asks about their vehicles, trips, charging and spending in plain language, and gets an answer drawn from their own records. They can also ask it to find and book a charging stop.

You start by connecting a language model to your API, then ground it in your data with retrieval augmented generation, and finally test it and compare the ways of improving it.



Lesson 1

LLM Foundations (Week 30)

- **How Language Models Work** - Understand tokens, context windows, and the transformer architecture at a working level.
- **Model Capabilities & Limits** - Recognise what models are reliable at, where they fail, and how hallucination arises.
- **Prompting for Reliable Output** - Structure instructions, examples, and constraints so results are consistent enough to build on.
- **Calling a Model from Your API** - Send requests from your backend, keep keys on the server, and handle streaming and failures.
- **Structured Output** - Make a model return data in a fixed format, and check it before the rest of your code uses it.
- **Tool Calling** - Let a model call your own API operations, and require the user's confirmation before anything is changed.
- **Cost, Speed and Model Choice** - Compare models on price, speed and ability, and choose per task.

Milestone 1: First Model Call - Add a chat endpoint to Tesla Trip Intelligence that calls a language model with a structured prompt, returns validated output, and can search for charging stations through a tool call.

Lesson 2

Retrieval Augmented Generation (Week 31)

- **Why Retrieval** - Understand why grounding a model in external documents beats relying on what it memorised.
- **Embeddings & Vector Search** - Represent text as embeddings, store them, and retrieve the passages most relevant to a query.
- **Chunking & Indexing** - Split source documents sensibly so retrieval returns useful context rather than fragments.
- **Building a RAG Pipeline** - Connect retrieval to generation and evaluate whether grounding actually improved the answer.
- **Metadata and Filtering** - Tag what you index, and narrow a search by those tags before ranking the results.
- **Grounding and Citation** - Make the model answer only from what was retrieved, show its sources, and say when it does not know.
- **Retrieval Over Your Own Records** - Answer questions about trips, charging and spend with safe database queries, limited to the signed-in driver's records so the assistant can never reach another driver's data.

Milestone 2: Ask Your Own Data - Build the retrieval layer of the assistant: document search over help and charging guides with cited answers, and questions about a driver's own trips, charging sessions and spend answered from their records only.

Lesson 3

Fine-Tuning (Week 31)

- **When to Fine-Tune** - Compare fine-tuning against prompting and retrieval, and recognise which problems each one solves.
- **Preparing Training Data** - Assemble and clean a dataset in the format a fine-tuning run expects.
- **Running and Evaluating a Fine-Tune** - Train an adapted model and measure whether it beats the baseline on your task.
- **Building an Evaluation Set** - Collect real questions with known answers, and use them to measure any change you make.
- **The Cost of Fine-Tuning** - Weigh the data, time and upkeep a fine-tuned model needs against what it gains.
- **Choosing an Approach** - Decide between prompting, retrieval and fine-tuning for a given problem, and justify the choice.

Milestone 3: The Evaluated Assistant - Complete the conversational assistant: answers from documents and records, and bookings through confirmed tool calls. Submit it with an evaluation set of real questions, the results it produces, and a short written case for why you would or would not fine-tune.

Externship



Real-World Industry Experience

The STEM Link Externships Program provides comprehensive real-world software development experience working with actual industry partners.

Program Structure

Dimension	Project
Duration	Weeks 18-32 (15 weeks)
Team Size	3-5 students per team
Company Assignment	1 industry partner per team
Internal Mentor	Dedicated STEM Link mentor
Weekly Commitment	12-15 hours (work + meetings)
Company Roles	Project Manager, BA, Tech Lead, Architect, QA Lead

Phase 1: Onboarding & Discovery (Week 18)

- **Week 18: Externship Launch & Team Formation** - Program overview, team formation, company profiles shared, role assignments, and project briefing from the PM covering business goals, users, and success metrics.

Phase 2: Requirements & Planning (Weeks 19-20)

- **Week 19: Requirements Gathering & SRS** - Stakeholder interviews, FR/NFR definition, personas, competitor research, SRS finalisation, architecture design, ERD, API planning, tech stack selection, approval.
- **Week 20: UI/UX Design & Sprint Planning** - User flows, Figma wireframes, design system, backlog creation, Sprint 1 planning.

Phase 3: Development Sprints (Weeks 21-30)

Sprint	Week	Project
1	21-22	Infrastructure: Repo setup, CI/CD, DB schema, auth, frontend scaffold
2	23-24	Core MVP: Core features, APIs, frontend flows
3	25-26	Expansion: Integrations, uploads, responsiveness
4	27-28	Polish: UX refinement, performance, error handling
5	29-30	Quality: Testing, bug fixes, security, cross-browser

Week 1 of Each Sprint:

Sunday Morning: Sprint Kickoff + Standup (30 min)

Sunday Afternoon: Development (2-3 hrs)

Weekdays: Async Work - Git commits, Slack discussions

Week 2 of Each Sprint:

Sunday Morning: Standup (15 min)

Sunday Mid-Morning: Sprint Review/Demo with PM + Tech Lead (1 hr)

Sunday Afternoon: Retrospective (45 min)

Sunday Late Afternoon: Next Sprint Planning with PM (1 hr)

Phase 4: Quality Assurance & Deployment (Week 31)

- **Week 31: User Acceptance Testing** - Company-led testing, bug fixing, performance & security review, UAT report.
- **Week 31: Production Deployment & Handoff** - Production deploy, configuration, documentation, final review, live URL delivery.

Phase 5: Final Presentation (Week 32)

- **Week 32: Final Pitch & Showcase** - Formal demo, technical deep-dive, impact analysis, reflection, pitch deck, demo video.

Career Coaching & Job Placement Guidance

Professional Development

- **LinkedIn Optimization** - Craft a compelling headline, summary, and experience section that attracts recruiters.
- **GitHub Portfolio Curation** - Organize repositories, write clear READMEs, and showcase best projects prominently.
- **Technical Blog Writing** - Start a blog to demonstrate expertise and improve communication skills.
- **Personal Branding** - Build a consistent professional presence across platforms.

Resume & Portfolio

- **ATS-Optimized Resume Building** - Format resumes for Applicant Tracking Systems while maintaining readability.
- **Portfolio Website Development** - Build a personal portfolio site showcasing projects and skills.
- **Project Documentation** - Write compelling project case studies explaining problem, approach, and impact.
- **Open Source Contributions** - Find and make first contributions to open source projects.

Interview Preparation

- **Technical Interview Patterns** - Practice coding interviews with common patterns and approaches.
- **Behavioral Questions (STAR Method)** - Structure answers using Situation, Task, Action, Result framework.
- **Mock Interviews** - Participate in mock technical and behavioral interviews with feedback.

Job Search Strategy

- **Job Search Platforms** - Navigate LinkedIn Jobs, Indeed, AngelList, and niche job boards effectively.
- **Networking Strategies** - Build professional network through events, communities, and LinkedIn outreach.
- **Salary Negotiation** - Research compensation, negotiate offers, and evaluate total compensation packages.
- **Remote Work Preparation** - Set up for remote work success including communication and time management.
- **First 90 Days Planning** - Plan for success in your first role with learning goals and relationship building.



Individual CV Review

ISURU GAYASHAN
FULL STACK DEVELOPER - SYSTEM ANALYST - SEO SPECIALIST
Baku, Azerbaijan | 091 600 0000 | 091 600 0000 | isurugayashan@gmail.com
LinkedIn: isurugayashan

SUMMARY
Experienced Full Stack Developer, System Analyst, and SEO Specialist with a proven track record in building web-based applications and business systems. Skilled in Java, PHP, JavaScript, Angular, React, Node.js, Python, Django, Flask, and SQL. Strong background in database design and optimization. Proven ability to deliver high-quality solutions for performance, security, and user experience while managing data-driven strategies for digital growth.

TECHNICAL SKILLS

Java	Angular.js	PHP	Python	JavaScript	jQuery	React	Node.js	MySQL	PostgreSQL	Redis	MongoDB	Git	Webpack	CI/CD	Cloud	Security
Expert	Advanced	Expert	Advanced	Expert	Advanced	Advanced	Advanced	Advanced	Advanced	Advanced	Advanced	Advanced	Advanced	Advanced	Advanced	Advanced

PROFESSIONAL EXPERIENCE

System Analyst, Faculty of Software Studies, University of Malaya (Jan 2020 - Present)
• Improved student enrollment, attendance, and grading modules within the university's MIS.
• Conducted requirements analysis and provided system enhancements aligned with academic needs.
• Designed and implemented database structures to support various academic operations.

Junior MIS Specialist (Intern), Kuala Lumpur Education City (KL-EC) (Aug 2022 - Present)
• Developed and deployed MIS-based projects and technical MIS tools.
• Improved system reliability via MIS-related technical actions, testing, and updates.
• Conducted quality control and testing of MIS systems.

Summary
The video is a detailed review and feedback session on a resume design and content. The reviewer provides specific suggestions on formatting, alignment, and how to better quantify achievements to make the resume more impactful. The discussion also covers the structure and order of resume sections, emphasizing clarity and effectiveness.

Action Items

- Change the file name to a proper version name without italicizing.
- Quantify achievements with specific metrics like percentage gains or keyword rankings.
- Maintain the order of professional experience, projects, and education, and improve descriptions for impact.

Isuru Resume Review
shapeeq khan · 20 days ago

Concurrent Projects Running Throughout

Weeks	Project	Description
11-31	Tesla Trip Intelligence	Full-stack AI-powered platform: React, Next.js, Node.js, Express, PostgreSQL, Prisma ORM, Clerk auth, mapping and payment services, retrieval augmented generation, Vercel and Railway deployment
18-32	Externships Project	Real company project with assigned PM, Tech Leads, Architects. Team-based development using real SDLC, sprints, company visits. Final pitch & presentation deck

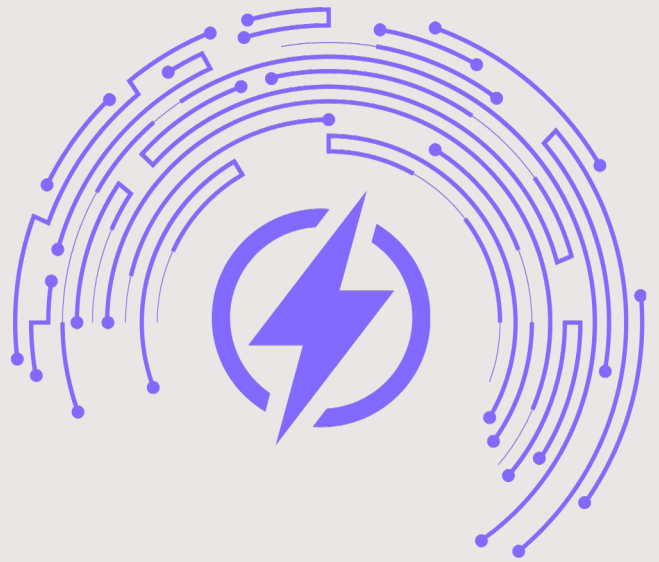
Program Final Evaluation

- Fully functional web application
- Complete documentation
- Test coverage
- CI/CD pipeline
- Production deployment

Externships Final Project

- The final working project approved by the provider
- Final Pitch and Presentation Deck

STEMLink Learning Experience



Hands-on Projects

Experiential projects are designed to reflect actual workplace challenges. Learners can return to lessons at any time during the course to refresh concepts.



Live mentorship

Search questions asked by other students, connect with technical mentors, and discover how to solve the challenges that you encounter.



Live-program Delivery

All sessions are conducted live and we're providing recordings for all the sessions.



Professional Certification

Providing a professional certificate to all the students who complete the program.



Learn from experienced engineers

Connect individually with our engineering team and discover how to solve the challenges that you encounter & tracking your progress.



Industry focused programs

Our course aligns with industry trends, providing real-world projects to ensure graduates are job-ready and competitive in the tech field.

