Generated Notes

Building Projects in the Age of AI: A Simple Guide

1. Learning Phase:

* **Goal**: Understand the basics. Like learning to ride a bike, you need to know how the pedals
* **How**:
  * **Documentation/Crash Courses**: Start with official guides or quick video tutorials to get fa
    * *Analogy*: Think of this as reading the instruction manual or watching a quick demo.
  * **Basic Project**: Build a simple project using what you've learned.
    * *Example*: If you're learning to bake, start with a simple cookie recipe.
    * *Tip*: Ask ChatGPT for project ideas based on the topics you've covered.
  * **Stack Overflow**: If you get stuck, see how others have solved similar problems.
    * *Analogy*: It's like asking your friends for advice on how to fix your bike. You get different
  * **LLMs (ChatGPT, etc.)**: Use AI as a last resort.
    * *Why?* You want to understand the fundamentals yourself before relying on AI.
    * *Analogy*: Don't ask a robot to ride your bike for you before you've even tried it yourself.

  Key Takeaway: Don't jump straight to AI. Learn the basics first.

2. Building Projects Phase:

* **Goal**: Create well-designed, maintainable projects. Like building a house, you need a bluep
* **Steps**:
  * **Design First**: Plan your project before writing any code.
    * *Analogy*: Imagine building a Lego castle. You need to plan the layout before you start s
    * *Tip*: Watch videos on project design.
  * **Choose Your Stack**: Decide which technologies (databases, languages, etc.) to use.
    * *Think about*: Pros and cons of each option.
  * **Database Design**: Plan how your data will be stored.
  * **Test-Driven Development (TDD)**: Write tests before writing code.
    * *Analogy*: Imagine you're building a bridge. You need to test each section to make sure i
    * **AI-Assisted TDD**:
      1. Generate code using an LLM.
      2. Ask the LLM to write test cases for the code.
      3. Check if the test cases are comprehensive (covering all possible scenarios).
      4. Refactor the code and test cases as needed.
  * **Refactor the Code**: Make sure the code is clean and easy to understand.