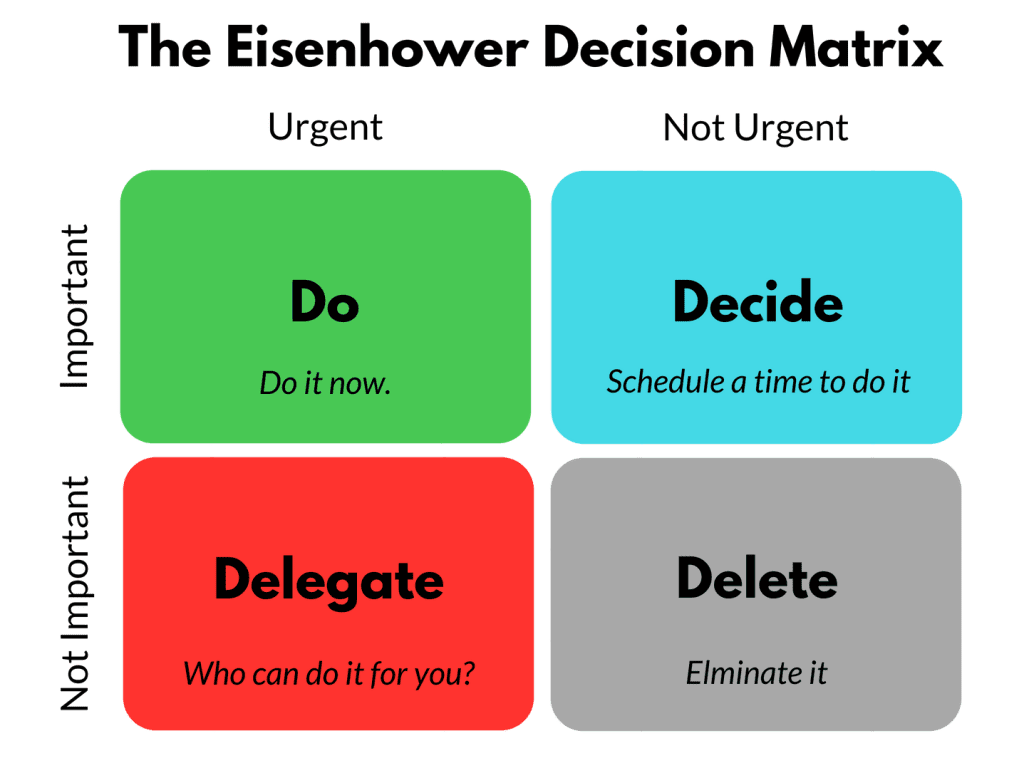
- Doing a deep dive and a map of what/where you want  
- Exploring and testing the viability of solutions

START TO FINDING WHICH TOOLS TO USE:  
- look for the metrics of a project

- make sample solutions to test them  
 - Use these as argument for tools/concepts

**Common Focuses** (What you might see code spikes in)

- Stack decisions   
- Technology evaluation (ex: UI library, other tools)  
- Code refactoring

- API integration (Social media APIs tend to be rough)

- New code concept or paradigm (object oriented or functional) to learn

- Optimization (Best to have multiple heads on it to try finding the optimal solution)

**Deliverables**:

- Do not attempt to achieve mastery, get it enough to work  
- MAKE OBJECTIVES AND A MAP TO WHAT YOU WANT

- If you don’t have a proof of concept, it didn’t happen

- Create Prototype implementation

- Create Performance Reports (Could be a readme)

- Create integration demo  
 - Create a data model  
 - Create security assessment

- Algorithm comparison  
  
  
**Code Spike Candidates**  
- List 2-5 tasks that would benefit from being a code spike  
- Importance list:   
 1) Essential feature and is an essential dependency  
 2) Essential feature but not an essential dependency  
 3) Not an essential features but is an essential dependency

4) Not an essential feature and also is not an essential dependency   
- What would be a useful deliverable(s)?  
- How long will it take you? (estimate)

**Understand the problem**  
- review official documents  
 - Get a feel for the field   
 -Record your notes  
- Use search terms   
- Give yourself mental space to explore (Lots of jumping around)  
 - Gathering links and information  
 - Record what you read  
- pay attention to common bugs  
- Don’t get into details  
  
**Plan the code spike** (*Knowing when to stop is important)*  
- Set your goals and objectives  
- Define success (for the code spike, not exactly for the project)  
- Allocate time for:  
 - Research  
 - Experimentation  
 - Documentation!!!!!!!!!!!  
 - Make notes; create notes and/or transcribe audio and give it to chatgpt for a summary  
- Have a review plan (with a colleague)  
 - Just call them and tell them your plan (just saying it will help you refine it)

**Documentation**  
- Use headings and point form (be concise and direct)  
- Make it easy to access  
- Include links with clear descriptions  
- Make sure there is a clear line between the objective and the summary  
  
  
**Share Results** *(The best research ever is meaningless if nobody knows about it)*  
- Start with the problem you’re trying to solve  
- Highlight constraints  
- Share the “so what”  
- Avoid self-depreciation and/or putting yourself in the work  
- Be concise and back up words with proof of work  
  
  
  
**Plan a code spike (30mins)**  
- Pick a topic that intersects priority and interest  
- Conduct **background research** and document  
 - Points of interest  
 - Things you don’t know  
 - Support resources  
 - Potential problem areas  
- Re-evaluate your objective to reflect your research  
- Break down what you need to learn in small elements (ordered if possible)

- Re-evaluate your time estimation