𝐌𝐚𝐬𝐭𝐞𝐫 𝐓𝐞𝐦𝐩𝐥𝐚𝐭𝐞 𝐅𝐨𝐫 𝐂𝐫𝐚𝐜𝐤𝐢𝐧𝐠 𝐀𝐧𝐲 𝐒𝐲𝐬𝐭𝐞𝐦 𝐃𝐞𝐬𝐢𝐠𝐧  
  
  
  
Establishing a robust foundation in system design is a crucial asset for advancing your career, regardless of your engineering specialisation. It serves as a key driver for success not only in interviews but also throughout your professional journey.  
  
Becoming an expert in system design is a gradual and enriching process, driven by hands-on experience. It's not just about reaching a destination; it's about appreciating the journey itself. Persistence, an open-minded approach, and an unwavering commitment to continuous learning will ultimately guide you to master the art of system design.  
  
To adeptly conquer any system design challenge, follow these essential steps:  
  
1. Define Functional and Non-Functional Requirements:  
  Identify and meticulously document both the functional requirements (what the system should accomplish) and non-functional requirements (how the system should perform).  
  
2. Understand Your Users and Their Behaviour:  
  Gain insights into the system's users, their behaviour, and their needs. Consider factors such as user demographics, types, usage patterns, and expectations.  
  
3. Shape the Data Model and Storage:  
  Create a comprehensive data model that encompasses the data schema. Choose the most suitable database type (SQL, NoSQL, distributed stores, or graph databases). Strategize how data will be stored, retrieved, and updated efficiently.  
  
4. Craft API Design and Communication:  
  Define the API endpoints and communication methods for interactions within the system and with external systems. Pay meticulous attention to API contracts, request/response formats, synchronous vs. asynchronous communication, and the selection of communication protocols (REST, SOAP, GraphQL).  
  
5. Identify Core Components:  
  Pinpoint the major components within your system, including core services, databases, load balancers, CDNs, DNS, caching layers, and any external services or APIs your system connects with.  
  
6. Envision the High-Level Architecture:  
  Develop a high-level architectural diagram that visually depicts the components and their interactions with core services. Understand how data flows throughout the system and how services communicate.  
  
7. Dive into Low-Level Design:  
  Delve deep into the low-level design specifics for each major component. Define data structures, algorithms, and implementation particulars. Contemplate optimisation techniques, trade-offs, and potential bottlenecks.  
  
𝐈𝐟 𝐲𝐨𝐮'𝐫𝐞 𝐢𝐧𝐭𝐞𝐫𝐞𝐬𝐭𝐞𝐝 𝐢𝐧 𝐚 𝐝𝐞𝐭𝐚𝐢𝐥𝐞𝐝 𝐠𝐮𝐢𝐝𝐞 𝐨𝐧 𝐚𝐩𝐩𝐫𝐨𝐚𝐜𝐡𝐢𝐧𝐠 𝐬𝐲𝐬𝐭𝐞𝐦 𝐝𝐞𝐬𝐢𝐠𝐧 𝐚𝐧𝐝 𝐬𝐨𝐟𝐭𝐰𝐚𝐫𝐞 𝐚𝐫𝐜𝐡𝐢𝐭𝐞𝐜𝐭𝐮𝐫𝐞, 𝐜𝐡𝐞𝐜𝐤 𝐨𝐮𝐭 𝐦𝐲 𝐥𝐢𝐯𝐞 𝐜𝐨𝐮𝐫𝐬𝐞 𝐬𝐭𝐚𝐫𝐭𝐢𝐧𝐠 𝐨𝐧 𝟏𝟕𝐭𝐡 𝐅𝐞𝐛 [𝐈𝐧 𝐭𝐡𝐞 𝐝𝐢𝐚𝐠𝐫𝐚𝐦 𝐢𝐭𝐬𝐞𝐥𝐟]

Activate to view larger image,

