**Designing and Implementing a Solution for the Course Project**

1. Understand what needs to be done
   1. Read the Project Requirements, top-to-bottom
   2. Reread the spec
2. Create a ToDo list
   1. Everything that needs to be designed
   2. Everything that needs to be created
3. Prioritize the ToDo list
   1. Consider alternative strategies
      1. Plan everything, then implement
      2. Interleaved plan-implement
   2. Consider alternative approaches
      1. Easiest-first (low-hanging fruit)
      2. Hardest-first (get the big stuff out-of-the-way)
4. My approach
   1. High-level plan for everything
   2. Build easiest-first
   3. Interleaved low-level plan/implementation
   4. Develop “technologies” for small components, use later
5. Date/Time module (just to get started)
   1. A (reusable) component that returns current date/time
   2. A driver that formats the date/time information
6. Next-step alternatives
   1. Could design/build a quote module
   2. Could design/build a country module
   3. Could design/build the Menu-driven application  
      Since we’ll need drivers for 6a + 6b, why not let the   
      App handle this so that we don’t build (extra) drivers?
7. The App design
   1. Menu-driven
   2. Commands
      1. Display date/time
         1. Short form
         2. Long form
      2. Quote-of-the-Day
         1. Like/not like
      3. Country lookup
         1. By country code, e.g. US or GB
         2. By name, e.g., United States or United Kingdom
         3. By continent, e.g., NA or EU
         4. Short display
         5. Long display
         6. Handle “not found” situations
      4. Quit
   3. Can specify details re what display looks like
      1. Now
      2. Wait until implementation
      3. Regardless, must document
8. Protocols
   1. User-App
      1. Menu items – a-d, 1-4
   2. App-Client
      1. Need specific messages, or
      2. Need specific function calls
   3. Higher-level consideration: Stateful or not?
      1. Best design is RESTful, not stateful
      2. So, App deals with state
      3. And, Client/Server are RESTful
      4. (Representational State Transfer (REST))
      5. E.g., Quote followed by Not-Like command  
         Which quote is retained by the App, not C/S
   4. Client-Server
      1. Collection of messages is the easiest approach
      2. True, regardless of encrypted or not
      3. Go to 7b to see what messages are needed
      4. Specify precise syntax for each message
      5. Both C+S will need to parse/understand all messages
      6. Look at PR, Step 4
9. Build the App
   1. Use stubs for each of the actions/commands
   2. Can hook up date/time module to see how well   
      the implementation works
10. Check Client/Server functionality
    1. Add UDP support early or late?
    2. My suggestion: late
    3. Implement just one command: date/time  
       But, use TCP for now
11. Initial integration
    1. Hook up App to Client/Server
    2. Make date/time command work
    3. Note: no encryption yet; we’re looking for functionality
12. Alternative strategies
    1. Can add encryption now, or later
    2. Can add additional commands now, or later
    3. Can add UDP now, or later
13. Fundamental Principles
    1. KISS – keep it simple and straightforward
    2. KIW – always keep it working
    3. ID – incremental development – design/build/test/refactor if necessary
    4. Always work on the easiest things first;   
       then address harder issues (they may not be so hard then)
    5. Test thoroughly
       1. Expected input
       2. Unexpected input, boundaries, limits, off-by-one
       3. Missing/inaccessible things: files, servers, etc.
       4. Provide mechanisms to keep the system running
    6. Reread the Project Requirements to ensure that you have covered  
       all your bases