# Echo Functional Analysis/Tasks

## Echo web service

* + Create endpoint structure
  + Implement echo functionality
  + Add a binary/base64 endpoint

## Response emulation web service

* + Create endpoint structure
  + Implement uploading templates
  + Implement deleting templates
  + Implement showing a list of all templates
  + Implement returning a response based on template

## Template based request frontend UI

* + Do research on existing technologies, tools and libraries that already exist, so no wheel is accidentally reinvented.
    1. Look into creating or generating json schemas
    2. Are there existing tools, project or libraries that allow the user to generate an html form using a UI? Make a list of do’s and don’ts, and apply to own project. Both, with how the function and structure as well as their UI
    3. UI frameworks such as bootstrap and how I can apply them to easily style the webapp.
  + Create endpoint structure (name)
  + Decide on existing framework, libraries, etc
  + Create a UI/UX wireframe in Adobe Xd
  + Implement the frameworks (static page)
  + Implement generating form from internal json
  + Implement loading from external templates
  + Implement generating request from template
  + Implement changing values

## Template based response frontend UI

* + Work off #3, and implement changing key names
  + Implement changing key data, such as its type and whether it’s required
  + Implement removing and adding items
  + Add button to load an “empty” template so you can start from scratch
  + Research server-side version control.
    1. Necessary? How will it be structured? (branch per template-name, or work in master/develop, …)

## Optional request-structure validation

* + Research json validation
    1. Are there existing libraries that will validate json (and/or xml) for me, without having to write my own?
    2. How strict does the comparison have to be?
  + Implement json validation into #3 and #4, and always do it. There’s no good reason not to.
  + Add a toggle that when turned on will perform a comparison between the new request and the template, and in case something is missing, show where and what.
  + Implement the actual comparison functionality.   
    Make sure to implement it as a separate class so you can reuse it in #6.

## Optional response-structure validation

* + Json validation already implemented in #5
  + Work from #5. Nothing even really needs changing, you’re just saving the generated code as a template instead of sending it off as a new request, so perform the same comparison as in #5.