Fullstack Engineer Coding Challenge

Objectives

This coding challenge is designed to assess a candidate's level of experience with writing a single full stack application. If you have any questions about the coding challenge or any requirements, please feel free to reach out to Alexander at alexander.maricich@lightfeather.io. Make sure to give yourself a generous amount of time to complete the challenge, especially if you wish to implement the optional portions.

Rules of Submission

1. This challenge must be completed in under 72 hours.
2. Your code and any assets must be version controlled with Git.Your submission must include a hyperlink to a github.com repository that you have pushed your project to.
3. Entire deliverable application must include any Dockerfiles or docker-compose.yml files required to launch your application.
    a. Note - If there are no Dockerfiles or docker-compose.yml files, your submission will not be considered.
4. The github project should have a README file that includes instructions on how to launch your dockerized application.
5. Submit your code by the google form provided with this challenge.

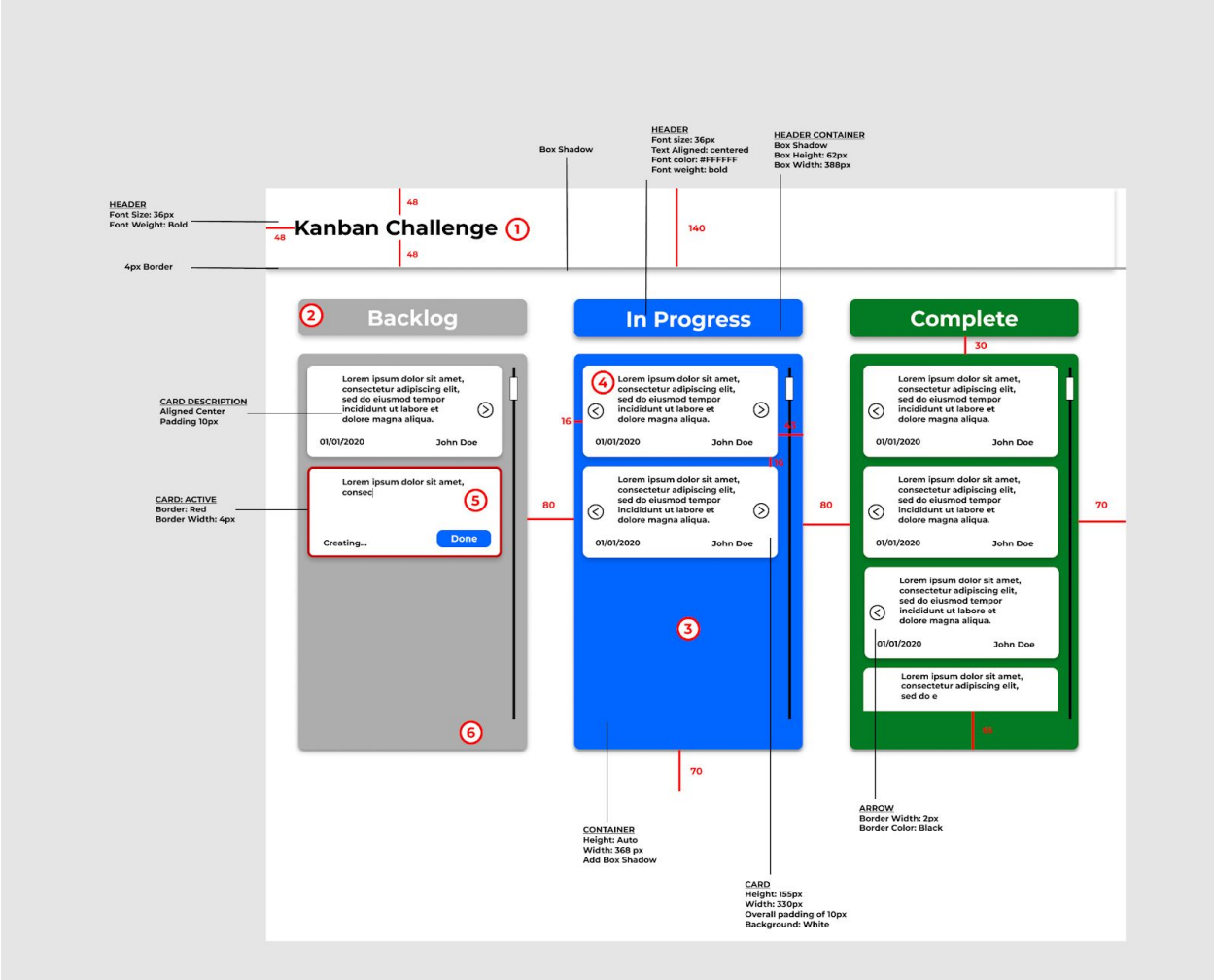
Overview

This application you will be building is a kanban board. Both client and server have specific requirements as to how the application should look and function. Below are a list of specifications that must be adhered to. Please note that the items are split into **required** and **optional** components. If you easily complete the required components, feel free to add optional components for extra credit.

You will be provided with the following assets that are required in completing this challenge:

1. Encrypted data file named **encoded_file.json**
2. UI Mocks

Client Specifications



The overall Mock can be found in this [Link to Figma Mock](). **If you have any issues viewing this mock, please contact the email stated above..**

- ❏ **Required:**
  - ❏ Your Kanban board must be able to perform the following functions:
    - ❏ Retrieve your Kanban board state from the API endpoint GET /api/kanban.
    - ❏ Capability to update your Kanban board state from the API endpoint POST /api/kanban
    - ❏ Render all tickets to the correct corresponding swimlanes based on the provided data file served through the above API calls
    - ❏ You must be able to move the tickets between any lanes and retain the state upon client refresh.
  - ❏ You must adhere to all the CSS requirements stated above.
- ❏ Optional:
  - ❏ Please implement a drag and drop functionality of the Ticket Cards.

Server Specifications

The main function of the server is to retain the current state of the kanban board in server memory. First, your application must read in the provided, encrypted JSON file. Then your application must serve two API endpoints, used to read and update the front end's state.

- ❏ There exists an in-memory datastore that holds the current state of the kanban board. This should not be a database, but a simple memory allocation.
- ❏ On application boot, the provided file is read into memory and decrypted. The contents of the file establish the base state of the kanban board.
    - ❏ The provided file is encrypted using a simple shift cypher. You must decode this file in order to retrieve human readable information that will be displayed on the front end.
        - The shift cipher works by shifting each character by a given value. For example, if your encoded string is `Gdg$` and the characters have been shifted by three, your decoded string would be `Dad!`
        - The provided file has been shifted by a value of `8`.
        - The maximum valid character is `~`.
        - If a character would be shifted below the minimum possible value, simply begin again at the maximum valid character and continue shifting.
- ❏ There exists a RESTful API that:
    - ❏ Runs on port `23456`.
    - ❏ Has an endpoint at `/api/kanban`.
        - ❏ A `GET` to this endpoint retrieves the current state of all tasks.
        - ❏ A `POST` to this endpoint updates the state of an individual task.