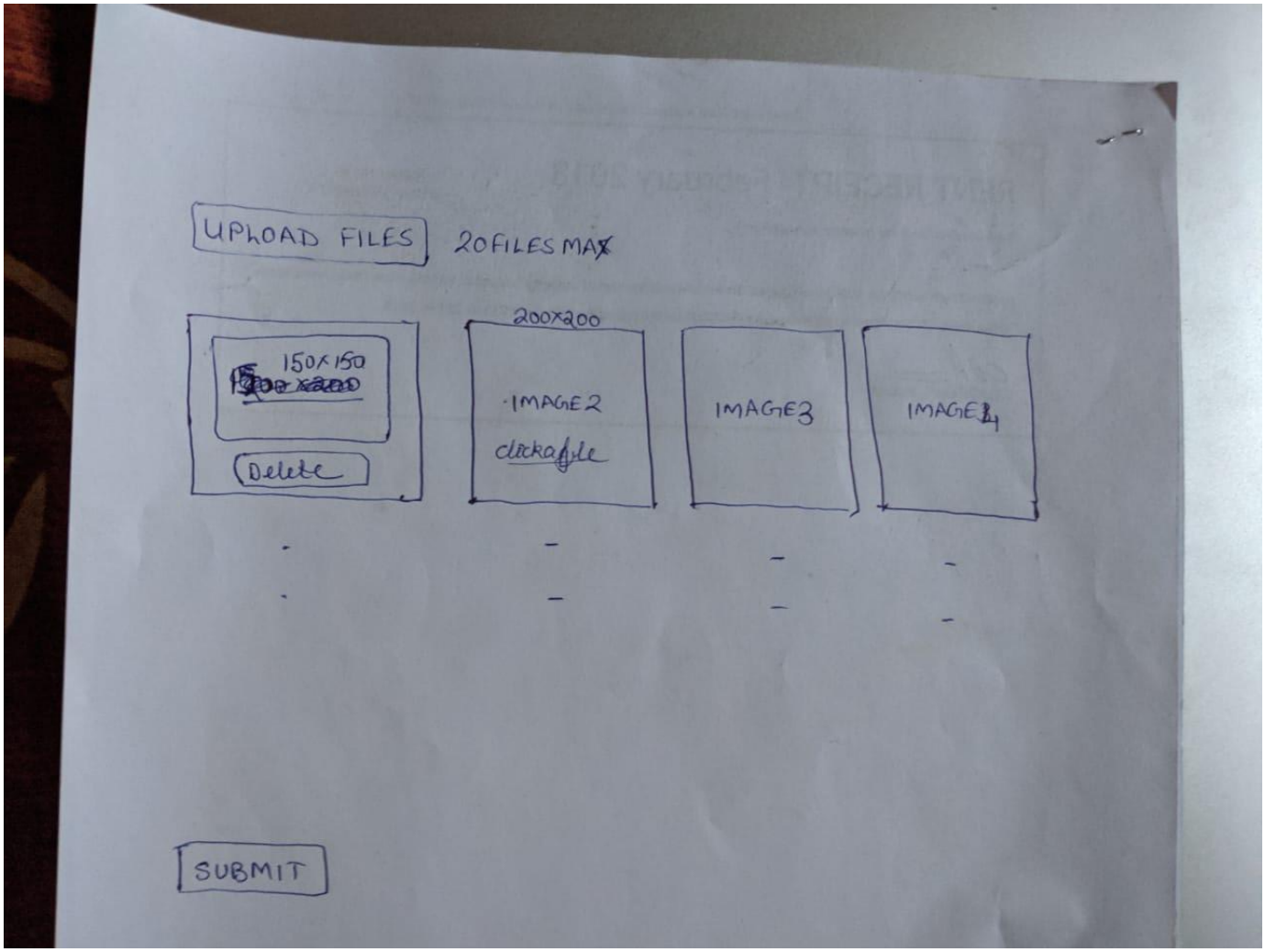
Framework problems:

1. File upload

Design and code a simple form which is able to upload images either 1 or multiple at a time.
The max limit of images to upload will be defined by a configuration variable(IMAGE_COUNT)
The UI for this should look be as close to the below image as possible



Details:

Link to refer for candidate: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/file
Candidate can only use this link and navigate to dependent links from this webpage if needed

1. The view should render cards for each image entry equal to **IMAGE_COUNT**
2. All cards are clickable and can be used to upload images(multiple)
3. Should be able to implement a functionality to upload and render 1 image at a time by clicking on upload button
4. When you click on any card it will open a popup to select images and the selected set of images will be displayed within the cards. You need not render the image in the card you clicked on but in actual order of display. For example, you had previously uploaded 1 image and then you click on card 4 to upload 2 images, post the upload the image will show in card2 and card3.
5. Also each card post the upload will show a delete button which can be used to delete that particular image

6. Once you have selected any number of images you can click on submit button and trigger a mock api to upload details of all files
7. Show an upload success Screen or toast
8. Able to drag and reorder images within the view(can use any library for it)(For react use react-dnd if not familiar with something else)
   The mock API will have the following info

   Url: /files/upload

   payload: {
   data: [{
   filename,
   size,
   contentType,
   disposition
   }]
   }

   Expectations:

   0-2 years: Complete points 1-3 with proper implementation
   2+: Complete points 1-6

   An outstanding candidate will be able to implement this completely

---

2. Movie Finder App(0-2) - Design

Create an application that will allow users to search for movies/TV Shows(or both) by entering a search term. User will be allowed to choose whether he wants to search for only movies, or only TV shows or both via a dropdown. Display the results as a list on the page.
Each result will have the name of the show/movie, its poster, vote count and one of its genres (name).

Things candidate will have to think about

- TV and movies have separate APIs. Wait for both in case user wants both in search
    - Can let the candidate just start with Movies in case seems confused
- Genres are stored as id in movies/TV shows results. Will have to ensure genres have been fetched from the api before rendering the results to get their names
- Think about debouncing while searching for recommendations

Features that be further added

- Allow user to filter based on the original_language/genre
- Allow sorting of results based on release date/first-air-date (they are different terms in the api for movies and shows), sort by popularity
- Clicking on an item will open a new page with details of the show/movie (routing)

Refer https://developers.themoviedb.org/3/getting-started/introduction for documentation
API Key : 635905955fd14b6379568443ce2949cd

Sample App

- Search Movies is a drop down which will have search movies, search series, search all
- 2nd image is for filter and sort dropdowns

iron

Search Movies

Iron Man
Up-votes : 999
Year of Release : 2008
Genre: Action

Iron Man 3
Up-votes : 765
Year of Release : 2013
Genre: Action

iron

Search Movies

Filter

Sort

Expectations

- 0-2 year experience candidate
    - Complete the task and add filter functionality. Anything above that can be considered as a bonus
- 2-4 year experience candidate
    - Expected to add filter and sorting functionality. Routing a bonus
- 4+ experience candidate
    - Can be expected to implement routing, filter and sort.

---

3. Card Memory Game(2+) Design + Coding

Card Memory is a game where you have to click on a card to see what image is underneath it and try to find the matching image underneath the other cards.

User can see a grid with n x n cards. All cards are faced down initially (hidden state). User clicks on a button to start the game, which starts a timer.
User can click on any image to unveil the image that is underneath (visible state). The image will be displayed until the user clicks on the 2nd card.
If there is a match, both the cards will be eliminated from the game (either remove them or leave them in visible state). If there isnt a match, the 2 cards will flip back to their original state(hiden state)

When all matches have been found, the user will see a dialog box showing a Congratultaions message with a counter displaying the time it took to finish the game.

Features that be furthur added

- User can choose between bdifferent multiple levels of difficulty(Easy, Medium, Hard). Increased difficulty means either decreased time available to complete the game or increasing the number of cards (or both)
- User can see the game stats (and persist in LS)
    - Number of games won/lost
    - Best time

Note: After showing the link, below ensure to tell the candidate that animation and css is not a priority here. Also do inform that the images are not required here and he can simple use any characters or numbers in their place.

Refer https://codepen.io/zerospree/full/bNWbvW

Expectations

- 2-4 year experience candidate
    - Candidate can be first asked to create a game with no timer involved. Just flipping of cards and eliminating same cards.
    - Ask to start a timer with keep decreasing at the top
    - Add congratulations dialog
    - Add a difficulty drop down which decreases the time available

- 4-6 year experience candidate
    - Expected to maintain a scorecard tab which has best timings and statistics which persists on refresh

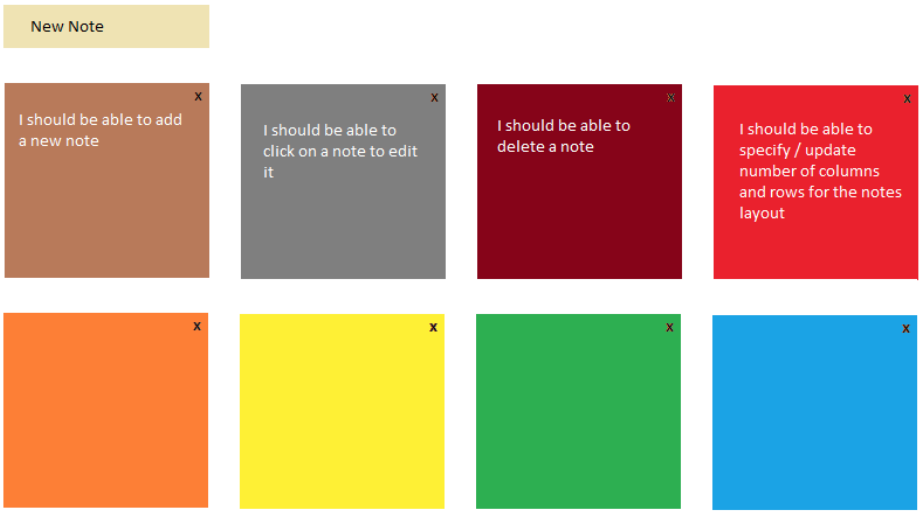
---

4. Sticky Notes (0-2)

Create a simple to do list which would be displayed as a grid on the screen. User can create new lists, edit previous ones, mark todos as complete and delete todos.

Essential Features

- The todo grid size would be configurable (The number of rows and columns can be entered as input)
- Todos should be persistent



As seen in the above image, clicking on the new note, creates an empty new note in which data can be entered. This can be edited/deleted. User will have an option to choose the background color of the item or can be a randomized color if nothing is chosen.

Here the config of the sticky app is 4 columns. There should be an option to configure this.

Bonus Features

- Allow user to filter todos based on status (complete/ incomplete)

- Ask the candidate to implement filtering based on colour and notice how much change in the code did it take to implement this

---

5. Image Collage (Design) (Design + Coding for 4+ yr experience)

collage of images (2-4 yrs - 90 minutes)
- candidate could refer mdn site for dom api references
- select/read local images and specify its dimensions in collage
- place image on a page on screen

2-4 yrs expectations
- Apis to be hinted by interviewer
- Draw a page on screen and accept file input
- Complete in 60 minutes
- Accept dimensions for each image, and place on page (move to new row when a row is filled up)

4+ expectations
- mouse interaction to rearrange image on page
- download collage as a single image

6. News Feed(0-2) - Shubham

Your task is to create a newsfeed App.
A News feed could be of two types:
1) New news feed (with new newsfeed Id)
2) Edited news feed(old newsfeed id).

A news feed must contain
1) User profile image
2) User name
3) News feed text
4) Intial timestamp/updated timestamp,

5) Mention edited if a news feed is updated
6) Feeds should be sorted by lastEdited property(latest should be above)

You have a FlockDupe global object which has 2 methods:
1) getUsers - to get all the users data
2) getPosts - to fetch initial posts
You also have to listen to an event called "post" which is hooked up on window object. It's event handler will receive an event object, you could access its data by event.detail.

To read this instructions again just type FlockDupe.help()

Expectations:

1. The candidate should be able to finish this task within 40 minutes with above average quality code. Only then should we go for extra things like persistence etc.
2. Basic component architecture, Feed, FeedItem, state modification logic and a decent, if not scalable way of data organization are expected.

Non framework problems

1. Alex the Hungry Dog (Design, 4+ Years)

Candidates can use the starter code.
Reference: http://g.recordit.co/1Phd7gjKEV.gif (Add bones and rotten meat)

Problem Statement

Alexander a.k.a. Alex is an ever-hungry dog and wants to eat **randomly** hidden bones in an N X N grid, where N is the number of rows and columns. Alex will always start its hunt for a bone by sniffing from the 0th cell,i.e (0,0). At the same time, some grid will have **rotten meat that** might kill Alex, and you wouldn't want that to happen :(

Alex can move around when you use the arrow keys to reach a bone and after every bone he eats, another bone will appear somewhere in the grid in an empty cell.

List of requirements:

1. An N X N grid. To start the game, press any arrow key to make Alex move in that direction. Candidate can hard-code at first but we can ask him to make it configurable later.
2. Alex will start moving in that direction and will reverse his direction once he reaches the end of the grid. Alex can also change his direction of movement whenever an arrow key is pressed. Alex will initially move at 1000ms per grid.
3. Add a bone and rotten bones in grids randomly. If Alex eats a bone, he increases his speed by 100ms and if he eats rotten meat, the game ends.

Tech Stack:

Please use Vanilla JavaScript. Usage of Frameworks isn't allowed.

Evaluation Criteria:

The problem is complex and completion will have a huge weightage. The following criteria will be used to judge candidates:

1. Usage of **Design Patterns** (Hint: Observer or MVC design pattern) in a meaningful way. (30%)
2. Code extendability/reusability/maintainability (30%)
- If there are enhancements, like adding two bones or more types of rotten bones, will the code allow that easily?
- Good programming practices and readability + modularity of code and functions.
3. Completion of requirements 1-2 (40%)

For Interviewers:

This problem is being set for 4+ experience. If you feel that a candidate is <4 Years Experience, but is good at JavaScript, you may proceed with this problem statement. Being a design question, this allows you to evaluate the candidate’s application architecture and design skills.

Expectations:

The time frame of the problem is within 60 minutes for Requirements 1, 2 and a good premise for 3.

3-4 Years Experience.

Hire if and only if:

1. The candidate was able to draw the grid.
2. The candidate was able to get Alex moving and handled the boundary conditions of the grid. We don't expect the candidate to have Requirement 3.
3. The candidate adhered to Evaluation Criteria 2.

Anything above increases the weightage of selection.

4+ Years Experience

Hire if and only if:

1. The candidate was able to draw the grid.
2. The candidate was able to get Alex moving and handled the boundary conditions of the grid.
3. The candidate adhered to Evaluation Criteria 1 & 2.
4. Requirement 3 might not be within the time limit, but the candidate should have a good approach and a premise set up for it in their code.

Alex the Hungry Dog - (Only Coding, 2+)

Download the starter kit from here.

Problem Statement

Alexander a.k.a. Alex is an ever-hungry dog and wants to eat **randomly** hidden bones in an N X N grid, where N is the number of rows and columns. Alex will always start its hunt for a bone by

sniffing from the 0th cell,i.e (0,0). At the same time, some grid will have **rotten bones that** might kill Alex, and you wouldn't want that to happen :(

Alex can move around when you use the arrow keys to reach a bone and after every bone he eats, another bone will appear somewhere in the grid in an empty cell. We will give you a starter kit to work with this problem and ask you to implement the rest of the features.

The following features are already completed for you in the starter kit:

1. A 10 X 10 grid. To start the game, press any arrow key to make Alex move in that direction.
2. Alex will start moving in that direction and will reverse his direction once he reaches the end of the grid. Alex can also change his direction of movement whenever an arrow key is pressed. Alex will initially move at 1000ms per grid.

Features to be completed:

3. At a time, there can only be **N randomly placed rotten bones** in an **N** X **N** grid. There will **only be one randomly placed bone** for Alex to eat.
4. Alex will initially move at 1000ms per grid and gains speed by 100ms for every bone he eats.
5. After Alex eats a bone, the positions of the rotten bones are changed randomly and a new bone is placed for him to find. If Alex eats a rotten bone, the game ends with the page displaying the user's score as an alert.
6. Please ensure that the rotten bone isn't kept within two grids of Alex's position (at game start or after he eats a bone).
7. Pause and Resume Alex's movements.
8. Alex should start from the same position and at the same speed in case the page is reloaded.

Evaluation Criteria:

4+ Years:
3, 4 & 5 are a must-have.

9. Exponential Backoff(2+) (Coding)

The problem is to implement a custom ApiManager and a Backoff scheduler.

Details

1. You are given an App which renders multiple accounts. The baseUrl for each of the accounts is different. However the endpoint for them will be the same. You need to implement a generic APIManager that is able to decide and route the request to the particular url based on combination of accountId and endpoint.
2. The API manager needs to have post, get and put methods. Each request will have some headers that are constant no matter which API is hit. Also there are certain common headers for each type of request. Ex. auth token
3. Also the API manager needs to provide a capability to retry request itself
4. Retry request needs to be scheduled with an Exponential backoff(Double)
5. Support a max number of retries before actually reporting an error
6. Support retires only for certain set of error codes
7. Support a custom timeDelay function for BackOff scheduler

Expectations:
2-4Years (Should be able to do points 1-4 in 1 hour)
4+ (implement 1-6 in 1 hour)

Example usage

```
ApiManager.post('/url', payload, accountId, { maxRetries  }).then().catch()
ApiManager.get('/url', params, accountId, { maxRetries }).then().catch()
ApiManager.put('/url', payload, accountId, { maxRetries }).then().catch()
```

Each request should execute the catch block only when the specified number of retries are exhausted

You can use any fetch request helper(browser fetch or axios). You are only allowed to refer docs of that helper

10. Promise internal implementation. (2+)

The problem is to emulate a design workable implementation of browser Promise API. Below is the stepwise description of functionalities that need to be implemented

MDN:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

1. Implement a Promise.all functionality first
2. Implement a Promise constructor that has .then and .catch functions
3. Implement a .then method in the manner of two callback functions like

   .then(function success() {}, function error {})
   Basically candidate should be able to understand how instances and prototypes work and how is he going to know which function has been used on the promise
4. Modify the above implementation to implement the .catch functionality
5. Implement chainable promises. Ex: myPromise.then().then().catch()
6. Implement .finally functionality
7. Implement .then functionalities post .catch(). Ex myPromise.then().catch().then()

Expectation:

0-2 Years: Candidate should be able to implement points 1, 2 and should have a approach for point 3
2-4Years: Candidate should be able to implement 1-4 and have a clear approach for point 5
4+: Candidates should be able to implement 1-5 completely (1 hour)

An outstanding candidate will be able to implement at least 6 and have an approach for point 7

11. Implement a Virtual DOM(Coding + Design 3+)

Ask the candidate if he has heard of Virtual DOM. If he hasn't we must explain him what it is and proceed with the problem

Details

1. Come up with an architectural representation of the DOM. Basically decide how he/she is gonna represent it with an object
   ForEx: const a = (
   <ul class="list">
   item 1
   item 2

   );

Is represented as

```
const a = (
  { type: 'ul', props: { className: 'list' }, children: [
    { type: 'li', props: {}, children: ['item 1'] },
    { type: 'li', props: {}, children: ['item 2'] }
  ] }
);
```

2. Once the candidate arrives at this, the expectation is that he/she should be able to implement a function which is able to convert the above into a DOMNode and then render it
3. The third step here is to arrive at how someone is going to update the elements
   Following conditions needs to be checked
   1. An Element was added
   2. An Element was deleted
   3. An element was updated
      This is following subparts
      1. The DOM node type got changed
      2. The DOM Node attributes got changes
4. Once all the above functionalities are completed, we must throw in the concept of keys and that if keys are specified we need to be able to compare respective keys and only compare those DOM elements at that nesting level
5. Should be able to do a workable demo

Expectations:

3+ years: Candidate should be able to implement 1,2 and has an approach for 3.1,3.2 points from above
4+ years: Candidate should be able to complete 1-3.1, 3.2 points and arrive on an approach to implement 4

Outstanding candidates will be able to implement 4 and arrive close to a workable demo.

Sample post for our reference.
https://medium.com/@deathmood/how-to-write-your-own-virtual-dom-ee74acc13060

Async Result With Throttle 2+ (Coding)

1 implement function throttle(callback, duration), where throttled function returns async result
2 callback is expected to return some dynamic result
3 If not throttling currently, then callback invoked synchronously (leading edge)
4 For all calls within interval, queue the requests until throttle duration and resolve all with latest return value from callback at the end of duration (when next callback is called - trailing edge)

bonus:
5. disable leading/trailing edge using config params
6. callback could accept arguments (of latest trigger)

Eg usage:

```
function heavyComputation(p) {
        return Math.random()*p;
}
const doit = asyncThrottle(heavyComputation, 10000);
const p1 = doit(1);; // resolved with doit(1)
const p2 = doit(2); // resolved with doit(4)
const p3 = doit(3);; // resolved with doit(4)
const p4 = doit(4);; // resolved with doit(4)
```

Expectation
0-2 yrs: 1 & 2
2-4yrs: 1-4yrs
4+ : complete solutions for 1-6

Throttled Fetch 2+ (Coding)

1. implement a util, that makes a fetch (or xhr) call (accepts fetch params)
2. ensures that only one **distinct** request is in flight
3. subsequent requests are throttled (wait) until call in flight is complete - make single fetch for all throttled calls
4. Util returns a Promise that resolves with fetch response body
5. for calls that were throttled, make a single fetch and resolve all these calls with this new fetch's response (any new invocations while this call is in flight will be throttled - will wait)

0-2 yrs: complete upto 3 (make fetch calls without returning promise/response)
2-4yrs: complete 1-5

gotchas:
* several of fetch result apis are also async
* need to identify requests as distinct (can limit util to GET requests only)

Eg. this set of statements should only make total of 4 requests (2 for each url)
myfetch('https://url1');
myfetch('https://url2');
myfetch('https://url1');
myfetch('https://url2');
myfetch('https://url1');
myfetch('https://url2');
myfetch('https://url1');
myfetch('https://url2');
myfetch('https://url1');
myfetch('https://url2');
myfetch('https://url1');
myfetch('https://url2');

Rate-limitter with Throttle 2+ (Coding)

- The utility takes as input a function and returns a function returning a Promise
- Automatically Delays calls to the original function such that the rate is kept below the specified number of calls in the specified number of milliseconds.

Expectations:

2-4yrs

- Creates queue

- solution:: https://github.com/rhashimoto/promise-throttle/blob/master/index.js