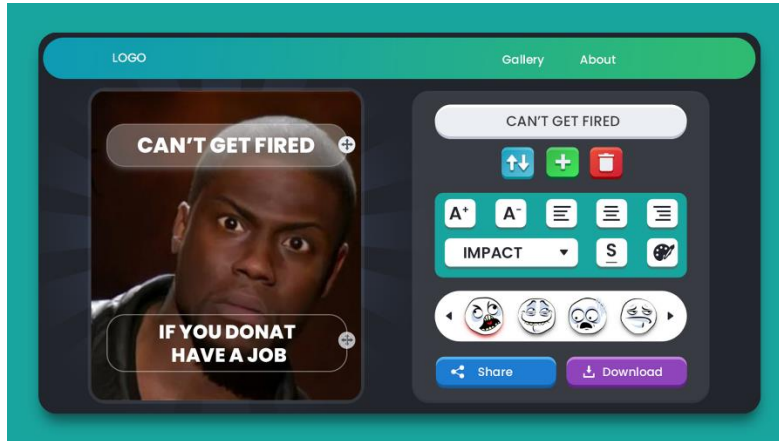


Ultimate Meme Generator

Sprint 2 Challenge

Let's code a super-duper Meme Generator

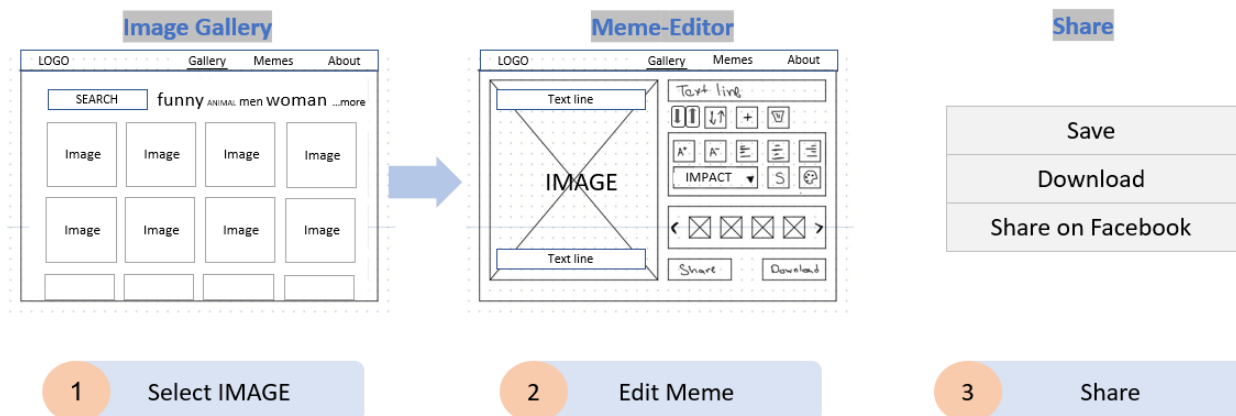


Product Definition - flow of the app

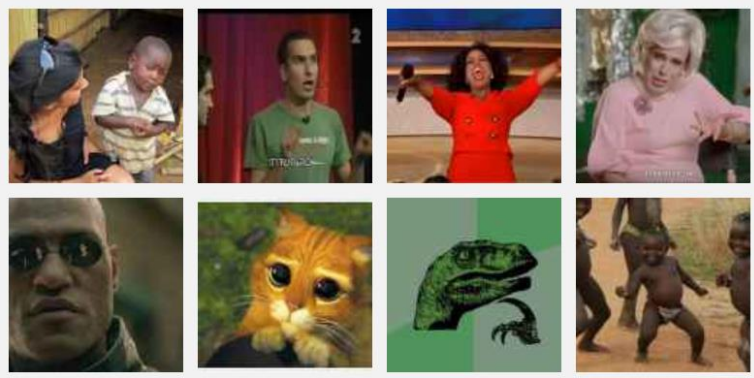
User selects an image, adds some text and downloads the picture to his device.

The app has two main UI areas: Image-Gallery and Meme-Editor (both are implemented on the same web-page).

Once the user selected an image on the Image-Gallery, the image is presented on the Meme-Editor then the user may edit that meme and once ready - download it.

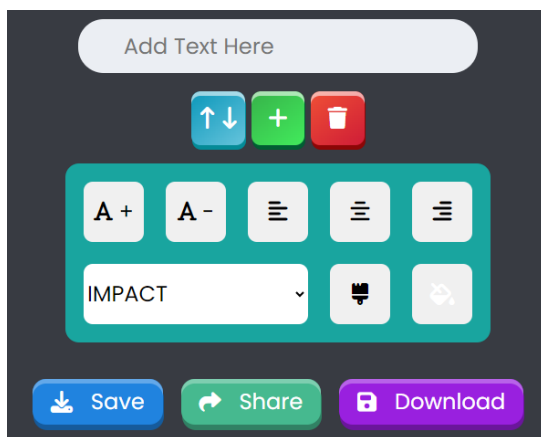


Initial Functionality: Image-Gallery



Show a gallery of images, when the user clicks an image, it is selected for edit

Initial Functionality: Meme-Editor



Start by showing a canvas, an input for entering the text and a color picker for setting the color

- Text (of line) shall be updated while typing
- Color should change when selecting a color
- Link to download the meme

Reference implementation

Play with some of the following apps on both desktop and mobile and pay attention to details.

Here is a [Reference](#) application

Here are some other references:

- <http://g.hazfalafel.com/index.php> (line draggable)
- <http://memebetter.com/generator> (very basic. no line move)
- <https://imgflip.com/memegenerator> (line draggable, side-line editing)
- <https://www.iloveimg.com/meme-generator> (line draggable, inline editing)
- <https://play.google.com/store/apps/details?id=com.zombodroid.MemeGenerator&hl=en> (Android. Look for more)
- <https://itunes.apple.com/us/app/meme-generator-by-zombodroid/id645831841?mt=8> (iOS. Look for more)

Data Model

Here is a proposed initial data structure (managed by a meme-service):

```
var gImgs = [{id: 1, url: 'img/1.jpg', keywords: ['funny', 'cat']}]  
var gMeme = {  
  selectedImgId: 5,  
  selectedLineIdx: 0,  
  lines: [  
    {  
      txt: 'I sometimes eat Falafel',  
      size: 20,  
      color: 'red'  
    }  
  ]  
}  
var gKeywordSearchCountMap = {'funny': 12, 'cat': 16, 'baby': 2}
```

Recommended order of implementation

Here is a recommended order of the development phases, make sure to commit and push regularly with good comments.

Phase1 – basic flow (~4-8 hours)

- 1) Build an initial home page (index.html, main.js, main.css)
 - a) Create an empty section for the gallery and for the editor
- 2) Commit and Push to Github, setup Github pages
- 3) Create a [memeController](#), code a function [renderMeme\(\)](#) that renders an image on the canvas and a line of text on top
- 4) Add a [memeService](#) with a [gMeme](#) variable and a function [getMeme\(\)](#), the function [renderMeme\(\)](#) can now render that meme
- 5) Add a text input and when it changed by the user –
 - a) update the [gMeme](#) using the [memeService](#) function [setLineTxt\(\)](#)
 - b) then [renderMeme\(\)](#)
- 6) Create a [galleryController](#), with a [renderGallery](#) presenting two images.
- 7) [onImgSelect](#) – call the [memeService's setImg\(\)](#) and then [renderMeme\(\)](#)
- 8) Commit and Push
- 9) Try your app from mobile

Phase2 – download meme:

- 10) Add a download link

Phase3 – add line operations:

- 11) Add a color picker button
- 12) Add the button “increase/decrease” font

Phase4 – multiple lines:

- 13) Add (to gMeme) a second line
- 14) Render the lines on the canvas
- 15) Add the button "add line"
- 16) Add the button "switch line", that switches the selected line
- 17) Draw a frame around the selected line (so the user can see which line is selected)
- 18) Use a single set of control-boxes to handle all the different lines

Phase5 – selectable lines:

- 19) When drawing text on the canvas, keep the location and size on the line object and use it to determine clicks on a line
- 20) When a line is clicked – it is selected for edit, so editor should reflect the current line

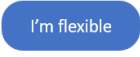
Phase6 – responsive layout:

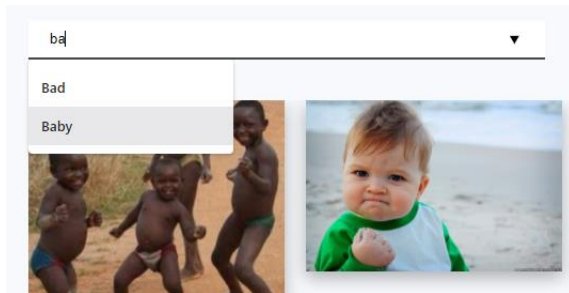
- 21) Build the page layout
- 22) Make it look good on mobile (both gallery and Editor)

Phase7 – More features

- 23) As a default, Use the common meme font "**Impact**" white with black **stroke**.
- 24) User can also click the text on the canvas to select the line
- 25) Set of controls: font family, font size, left-right-center alignment.
- 26) Up-down arrows for positioning the text line
- 27) Add a Delete-Line button.

More features

- A. Add button  at the Gallery. Clicking the button randomly generates Meme and present it at the editor: start with an image and a single text line.
- B. Use a memeService to save the created memes (to localStorage) and display them in a "Saved Memes" section. The saved memes can be re-edited.
- C. Image gallery filter (use [<datalist>](#))
Gallery Filter: The user should be able to filter the images and also clear the filter



D. Add stickers (Those are lines that have emojis characters such as: 🤔🤔)

E. Support "Drag&Drop" of lines and stickers on canvas.

F. Share on Facebook (use the sample code provided)

G. Let the user pick an image from his device instead of the gallery

H. Add "search by keywords" to Image-Gallery Page.

Each word size is determined by the popularity of the keyword search - so each click on a keyword makes that keyword bigger

TIP: use an initial demo data so it will look good when loads



I. Inline (on Canvas) text editing

J. Resize / Rotate a line.

K. Support using various aspect-ratio of images

L. Website theme: celeb-meme, politic-meme, ani-meme, kid-meme, Mondial-meme

M. Use the new Web Share API to share your meme

N. i18n for Hebrew

Design and Responsivity

- Both the Image-Gallery and the Meme-Editor shall look good on both Desktop and Mobile.
- Use pure CSS – do not use CSS Libraries and/or jQuery.
- Test your app on your mobile devices by accessing the app hosted in Github pages
- Keep correct proportion of images on both Canvas and Gallery
- Note that it is possible to measure the text size on the canvas
- Select one of the UI designs given in Appendix3, you may also allow yourself some creativity but focus on implementing the app.
- The UX (User Experience) definition of the app is given in MemeGenUX.PDF
- The UI (design of the app) definition is given in Appendix2 below.

Delivery Instructions

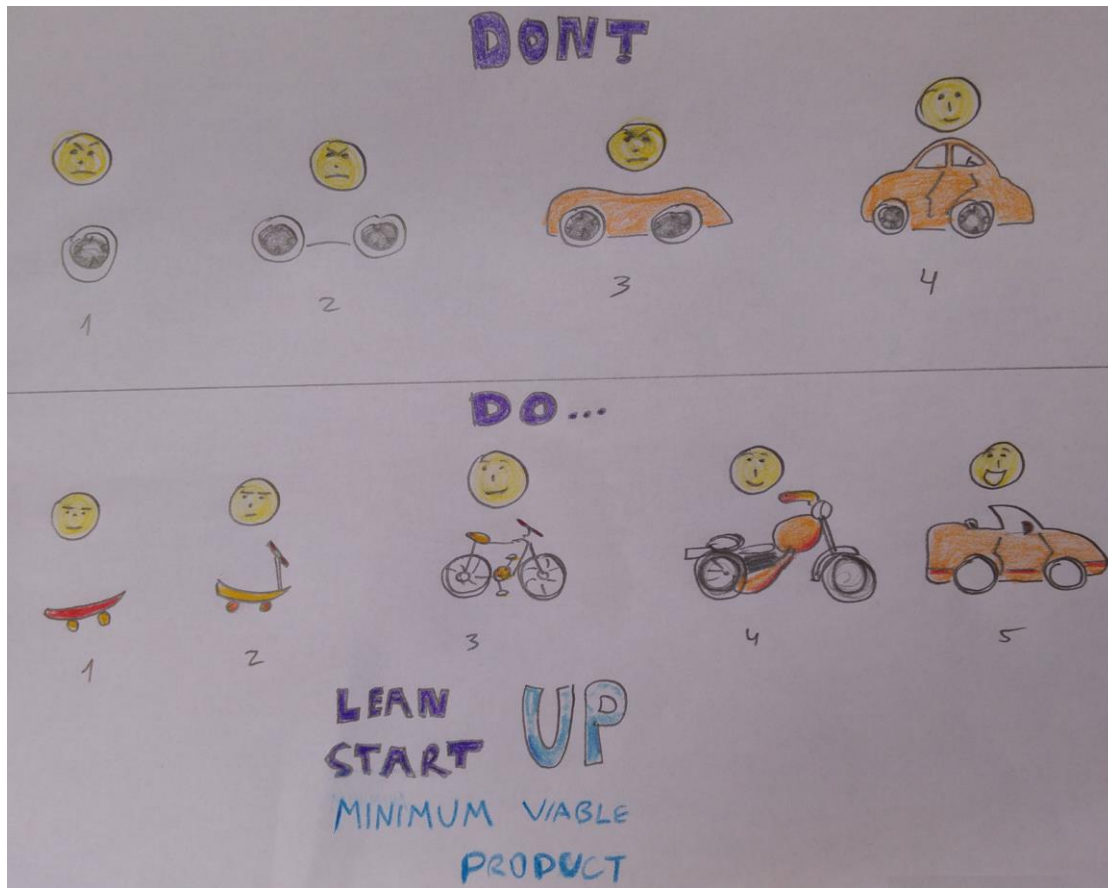
Use a git repository on Github, your tech leader will monitor this repository to help you get the maximum out of this sprint.

Please make sure you commit and push regularly, at minimum - make sure your code is updated every few hours.

Publish and test your app on Github Pages.

Try to complete all functionality at least a few hours before the final delivery time and then concentrate on UI finalizations.

Appendix1 – MVP – Minimum Viable Product

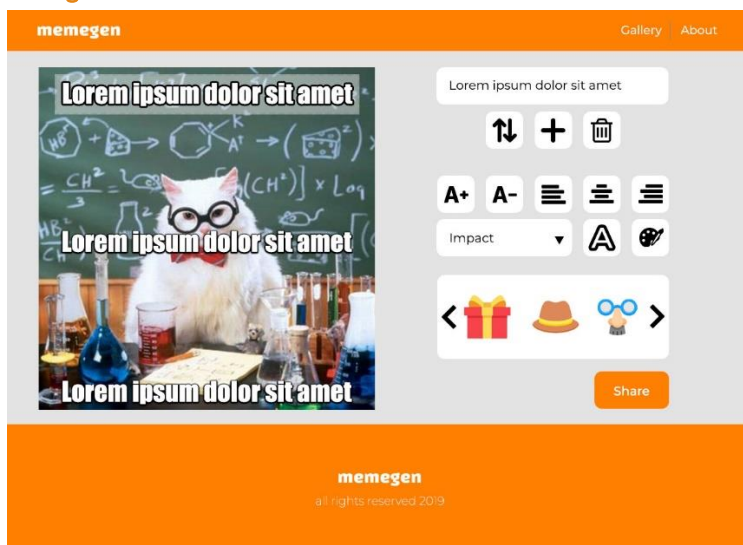


Appendix2: UI

Please note that the designers may have not completed the entire design. They are also not available, so you will need to take some UI decisions, please make good ones.

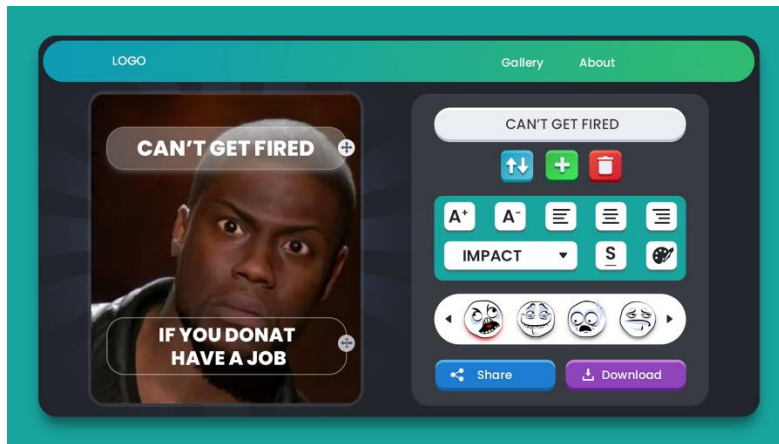
Select one of the following 3 UI designs:

Design1 - Sergei



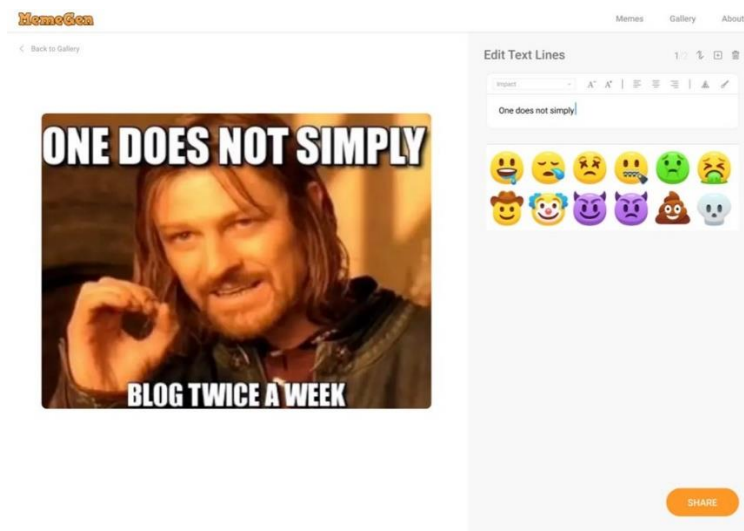
PSD file: Located in the "DESIGN/Meme1 - Sergi/"

Design2 – Game Style



PSD file: Located in “DESIGN/Meme2 – Game Style/” folder

Design3 - Alex



Figma: <https://www.figma.com/file/LTqroiQHhQwDMU8VD5bjI6/MemeGen?node-id=0%3A1>

Link to Inspect: <https://inspect.ceros.com/design-upload>