Online Multimedia WS2022/23 – Final Project

Task: Your task is to implement a meme generator web application, consisting of a Node.JS based backend and Javascript based frontend. We provide a template that runs your project. Below we specify a list of features, each feature is subdivided into 3 levels. The amount of bonus points that you gain depends on the amount and level of features that you implemented (see *Grading*).

Clarification: There will be a final written exam. In the exam, we will ask questions about the theoretical lecture content as well as the practical aspects of this course. You can reach 100 points in the exam, which equals a 1.0. Further, you can achieve up to 30 bonus points by completing this project with sufficient features as described below.
The bonus points cannot be used to pass the exam.

Project Submission: For the final submission of the project, you need to hand in **until the 24th of February**:

1. All source code and materials. The project must be runnable via the provided template, without any further steps being required (e.g. importing some data somewhere, …). Do NOT submit the *node_modules* folder! This is a local folder that should neither be checked in a version control system nor be sent to other computers. **Submissions including the *node_modules* folder will be deduced 20% of the reached points.**
2. A document that lists used third-party contents (e.g. graphics, icons, …). Libraries listed in npm's *package.json* do not have to be listed here again
3. A list of all implemented features (we will provide a template)
4. **A video (maximum 1 minute per implemented feature)** going through all implemented features. (The video just serves as orientation for us where to find and how to use features. It can be a plain screen recording while using the website.)

Everything should contain all necessary information, but not extend too much above that, e.g., in the video to demonstrate that hand drawing works can be done by just drawing a line and it is not necessary to draw a really nice picture for 10 minutes.
Please register your group in Moodle until the 23rd of December!

Group Size: The allowed group size is 1-4 students, but we highly recommend 3-4 students. Please use the Moodle forum, Discord, and the group assignment to specify or find a group.

Allowed Software: You should use technologies that are close to the ones teached in this course. The backend has to be Node.JS based, the frontend implemented with Javascript. However, if you use anything other than the recommended MERN stack, we cannot provide support in case of problems or answer question regarding that software/framework. Your project has to be able to run in an isolated infrastructure (i.e. offline) and must not be depended on third party online services (like Firebase or MongoDB Atlas).

	Examples that are allowed	**Examples that are not allowed**
Backend	Express, Nest.js, Loopback.io	Java Spring, PHP, Go
Frontend	React, Angular, a plain html webpage	Android app

However, we highly recommend you to stick with the course's technologies!

Grading: A feature will count as implemented, if it is demonstrated in the video (and marked in the list), and the code is readable and not plagiarized. **Unfinished or not properly working features will not be counted.** In individual cases, e.g., one mostly working / nearly finished feature, which is needed for the next higher number of bonus points, we might consider an exception. In these cases explain that feature (what works and what does not) in detail including showing and explaining your code at the **end** of your video.

Plagiarism: Unless specified otherwise, you are allowed to use libraries or code snippets from the internet. In case a third party library is used, you need to note the source, a (short) explanation, and the corresponding license in the code. For code snippets, you have to note the source as well as explain the code in a comment (for longer code snippets please explain every main section).
In cases where you worked together with other groups or asked for help, please also specify that in a comment, e.g., "We were stuck with feature XX and asked group 2 for help. Our implementation follows their explanation."
In case we find any plagiarism, we will not count the corresponding feature and discuss further consequences.

Final Project - Task

Your task is to create a website to generate, download and display memes, i.e., [funny] pictures, gifs, or videos with text. The site should allow a user to upload (or select) an image, add text and download the created image. This should be possible using either an API call or a GUI. Other users then should be able to view the created images.

This document is structured into three parts. First, the general structure. Second, A list of suggested features and third, an appendix explaining each feature in more detail.
Each feature has three different difficulty levels: basic, intermediate, and advanced. In some cases, a higher level means to add functionality to a previous stage, i.e., you need to complete all stages for advanced. In other cases, functionality has to be replaced, i.e., only the highest stage has to be implemented.
Therefore, check out all features briefly before you start with the project and look at the different levels of the feature in detail before you implement it to prevent doing work twice.

There are 22 features in total. You will receive:

1. 7 bonus points, if you implemented 10 or more features (level basic or higher)
2. 11 bonus points (in total) if you implement 10 or more features (level intermediate or higher)
3. 15 bonus points (in total) if you implement 10 or more features (level advanced)
4. 20 bonus points (in total) if you implement 15 or more features (10 advanced, 5 intermediate or higher).
5. 25 bonus points (in total) if you implement 20 or more features (15 advanced)
6. 30 bonus points (in total) if you implement all features (15 advanced, the others intermediate or higher).

Examples: 5 basic, 3 intermediate and 2 advanced features -> 7 bonus points
10 basic, 0 intermediate and 0 advanced features -> 7 bonus points
0 basic, 7 intermediate and 3 advanced features -> 11 bonus points
0 basic, 0 intermediate and 13 advanced features -> 20 bonus points
6 basic, 0 intermediate and 15 advanced features -> 25 bonus points

Each feature counts only once (with the highest level implemented).

In general, you should implement the website in a coherent way, i.e., the features should be usable and useful. For example, it does not make sense to implement a filter function but it is not possible to apply that filter to the images.

Submissions that do not run out-of-the-box by launching the provided template will be deduced 20% of the reached points!

General Structure

The general structure of the website should be as following, where each point is a separate page unless specified otherwise. Either “Popup” refers to an actual popup with a separate URL, or a layer added over the current page. You are free to deviate from this scheme if that will increase the usability or design of your webpage. **Do not put everything on one page!**

Navigation (on all pages):
You should have a navigation element that shows links to all pages mentioned below.

Index:
You can either have a dedicated start page, or use one of the other pages as landing page.

Editor:
Here, the user should be able to create memes. The Editor itself can be one page or split into multiple pages, i.e., uploading new images for templates could be a separate page or popup and the result is saved and made available on the main editor page.

Account:
If you implement account management, you should have a profile page with personal information, own memes, likes, comments, etc. (depending on what features you implement). Drafts/history of created memes can also be integrated into the editor. Further, you need login/logout/(forgot password) pages or use popups for this.

API:
An API-functionality should work without any GUI elements and output should be machine-readable. For example, instead of going to “example.com/registration” entering my name there, clicking okay and receiving a new HTML page with “Welcome #name”, we send a request “example.com/registration?name=#name” and receive text/json “{success=true, addedName=’#name’}”. **The API itself should not return a website but instead only just plain text or JSON and it should not be reachable over the navigation!**
However, there should be a page serving as documentation, i.e., which API calls are available under which URL, which are the required parameters that need to be added to the call and by which method (GET/PUT/PUSH/DELETE). This page should contain an example call for each API, which opens the result in a new window.

Overview / Endless scroll:
Here, all created (and uploaded) memes are shown in an endless scroll, i.e., X memes are shown, and if you reach the bottom, the next X new ones are loaded.
Memes should be presented one per row on top of each other, with minimal information like title, votes, vote buttons, screen reader, etc.

Single view:
This should be the detail view for a meme, including detailed information, comments, option to add comments, etc. (depending on what features you implement). It should also contain left right buttons to go through the memes or an auto play button. Each meme should have a dedicated URL that opens that meme in single view. **This is not the direct image URL.**
Instead of a dedicated page, you can show this as a layer over the Overview.

Appendix

1. Editor - Upload templates

Here, the user should be able to choose their template for the meme. You should present previously uploaded (or manually added by you) templates, which are locally stored on the server, to the user. Further, implement some of the following upload methods:

a) uploaded by the user via file upload
b) uploaded by the user through providing an image URL
c) Images downloaded from a third party service, e.g., imgflip and selected by the user
d) Screenshot from a user provided (non image) URL
e) Photo from a connected camera
f) Hand- (or rather mouse) drawn.

2. Editor - Text input

How can the user add text to the selected template?
Basic: Two simple textboxes with unformatted text, at specific locations in the image. The user has some option of entering the text somewhere and it will appear on the generated image.
Intermediate: The text should be movable somehow (extra buttons are fine) and the text inside should be formatable, e.g., size, color, font, background etc. Further, the text should already be visible in the image and changes should be applied immediately.
Advanced: Like “intermediate”, but with three or more freely movable and resizable textboxes.

3. Editor - Canvas and images

Intermediate: The user can append other images (from a set of templates or upload, etc.) left, right, top, or bottom from the current meme. Aspect ratio of the original and the added image should be conserved.
Advanced: You can freely adjust the canvas size and place images anywhere inside the canvas.

4. Editor - Generation and Download

By default the generated meme should be added to the website. Additionally the creator receives immediate feedback.
Basic: The finished meme is generated and shown to the user.
Intermediate: The user can choose to either create the meme locally on their device or on the webserver. Afterwards they are shown the image and provided with a download button.
Advanced: In addition to intermediate, the user can specify a target file-size before the image creating and share buttons are presented below the image. The generated meme should have the specified (or lower) file-size. Providing a button that copies the “singleview-URL” of the image into the clipboard suffices as a share button.

5. Editor – Navigation

Intermediate: Provide buttons to switch to the next and previous template. Keep already entered text or other settings, but provide a “clear” button.
Advanced: Full navigation through other templates should be possible (sidebar, dropdown or similar). Keep already entered text or other settings, but provide a “clear” button.

6. API - Meme Creation

Here the task is to create an API call, that takes different parameters (GET or POST) and returns an image or an URL to an image or an URL to the SingleView of that image. The used template can be specified by name, URL or as file. (Only one Variant has to be provided.)

Basic: Given a template and bottom as well as top text the meme is returned. The positions of the texts are chosen by the server.
Intermediate: Given a template and a set (more than two possible) of texts the meme is returned. For each text at least two of the following should be customizable: size, position, color, font, transparency, background.
Advanced: Same as Intermediate, but for one template a list of sets of text is given and as many memes are returned (list of URLs or *.zip file). (You may also use a list of (maybe) different templates instead of just specifying one.)

7. API - Meme Retrieval
Here the send request should contain information to identify a meme and again return an image or an URL to an image or an URL to the Single view of that image.
Basic: Either specify a specific meme by identifier (e.g. single view URL) or request a random meme.
Intermediate: Same as Basic, but also provide meta-data (e.g., title, creator, creation-date, image captions as plain text, votes, used template, etc.)
Advanced: Specify constraints, e.g., search parameters, newest, oldest, and a maximum amount of requested memes to request a set of memes, i.e., return a list of URLs or *.zip file including meta-data.

8. View – Overview
Here, all created (and uploaded) memes are shown in an endless scroll, i.e., 40 memes are shown, and if you reach the bottom, the next 40 are loaded.
Memes should be presented one per row on top of each other, with minimal information like title, votes, vote buttons, screen reader, etc.
Basic: Only show the title and the image
Intermediate: Also show other information like number of comments, up/down votes, etc.
Advanced: Allow for interaction, i.e., up/down vote memes from the overview page.

9. View – Single view
This should be the detail view for a meme, including detailed information, comments, option to add comments, etc. It can also contain left right buttons to go through the memes or an auto play button. Each meme should have a dedicated URL that opens that meme in single view.
Instead of a dedicated page, you can show this as a layer over the Overview.
Basic: A simple slide show with left/right buttons.
Intermediate: Basic with an added random button, which jumps to any randomly selected meme.
Advanced: Intermediate, with an added autoplay button. Activating this button displays a new meme every x seconds (either choose a fix number or let the user choose x).

10. View - Sort and Filter
Here, the idea is to offer the user options to sort or filter the amount of memes displayed in either overview or Single view. In single view a sort should influence the left/right buttons and auto play, a filter should additionally influence random.
Basic: Sort after creation date, or title, or similar.
Intermediate: Filter after some numerical value, e.g., creation date after/before, number of votes/views, etc.
Advanced: Filter after / search for other meta data like used template, file format, (parts of) the title, (parts of) the captions, or creator/uploader, etc.

11. View - Sort Integration
Basic: Integrate the sort/filter/search implemented in 10.) into both Single-, and Overview.

12. Profile – Authentication
Here, you should implement some possibility for the user to identify themselves, e.g., the user can always upload under the same username.
Basic: Anything that does the above. Does not need to be secure, i.e., it is okay if users could upload under a different name.
Intermediate: Make it secure, i.e., require a password to login with a specific username/email address.
Advanced: Integrate a third party service(s) to authenticate users, e.g., google/Facebook/steam/etc.

13. Profile – Privacy
Basic: Allow users to upload memes as unlisted (i.e. can be accessed if the single view URL is known, but does not appear in overview) or private (i.e. can only be viewed by the creator, when logged in)

14. Profile – History
Basic: Show a history of one's own created memes.
Intermediate: N/A
Advanced: Allow users to safe drafts and edit those later.

15. Profile – Social
Here, implement some features for social networking.
Basic: Allow for account bound votes (i.e., every user can only up-/down vote each meme once)
Intermediate: Account bound comments.
Advanced: N/A

16. Template Formats – Gifs
If you allow gif as meme templates, then:
Basic: Convert them to still images on upload, i.e., use one of the frames of the gif.
Intermediate: Create the meme as a gif and display it accordingly. The meme then moves. The caption can be static (constant over all frames).
Advanced: Integrate an option into you editor for placing temporal textboxes, i.e., text appears/disappears/changes over the duration of the gif.

17. Template Formats – Videos
Analogue to 16.) gifs.

18. Video stream
Basic: N/A
Intermediate: Implement a video stream similar to the auto play slideshow (but as an actual video stream and not a series of images). Can be rendered live or be automatically precomputed, i.e., every hour/day a new video file is generated.
Advanced: Encrypt the video stream (in some form, e.g., https)

19. Accessibility - text to speech
For people with visual impairment, offer a screen reader.

Basic: Read textual information, e.g., the title.
Intermediate: Read previously textual information, e.g., the captions or the name of the used template.
Advanced: Read a more detailed description of the image, e.g., what can be seen. For know template, these could be predefined for others user entered. Alternatively, use a third party API like google Lens to get a description for any image.

20. Accessibility - speech to text
Implement the option to use one's own voice to control the website. It is fine, if these featured work only in certain browsers (please specify if that is the case).
Basic: N/A
Intermediate: Dictate text, e.g., when a text field for captions or the title is selected.
Advanced: Offer voice control, e.g., to press a button, select a text field directly dictate captions, etc.

21. Offline Capabilities
Modern progressive web apps (as studied in tutorial 9) allow webapps to be "installed" locally on a device. Thereby an webapp can be used when the device is offline (i.e. the backend applicaiton is not running / reachable). As soon as the device is back only, the local actions and remote changes are synced again.
Basic: The meme generator webapp can be opened also when the device is offline (i.e. the backend application not running). A custom error message should be prompted, that informs the user that without internet connection neither contents can be viewed nor edited.
Intermediate: In the offline mode, formerly created memes can be viewed
Advanced: Memes can also be created while the device is offline. They are persisted in some browser storage and will be synced with the server as soon as the webapp has a server connection again.

22. Statistics
Here, the task is to integrate dynamically generated graphs into the website. That means, that you provide up to date information and generate a graph from that.
Basic: One simple graph anywhere. For example, a global graph showing uploads over time or a simple graph on the single view like views over time.
Intermediate: Provide one (or more) complex graphs for either the single view page (meme views, up votes, down votes, comments, etc.) OR the template selection in the editor (how often the template is used, likes the created memes get, etc.).
Advanced: Provide one (or more) complex graphs for the single view page (meme views, up votes, down votes, comments, etc.) AND the template selection in the editor (how often the template is used, likes the created memes get, etc.).