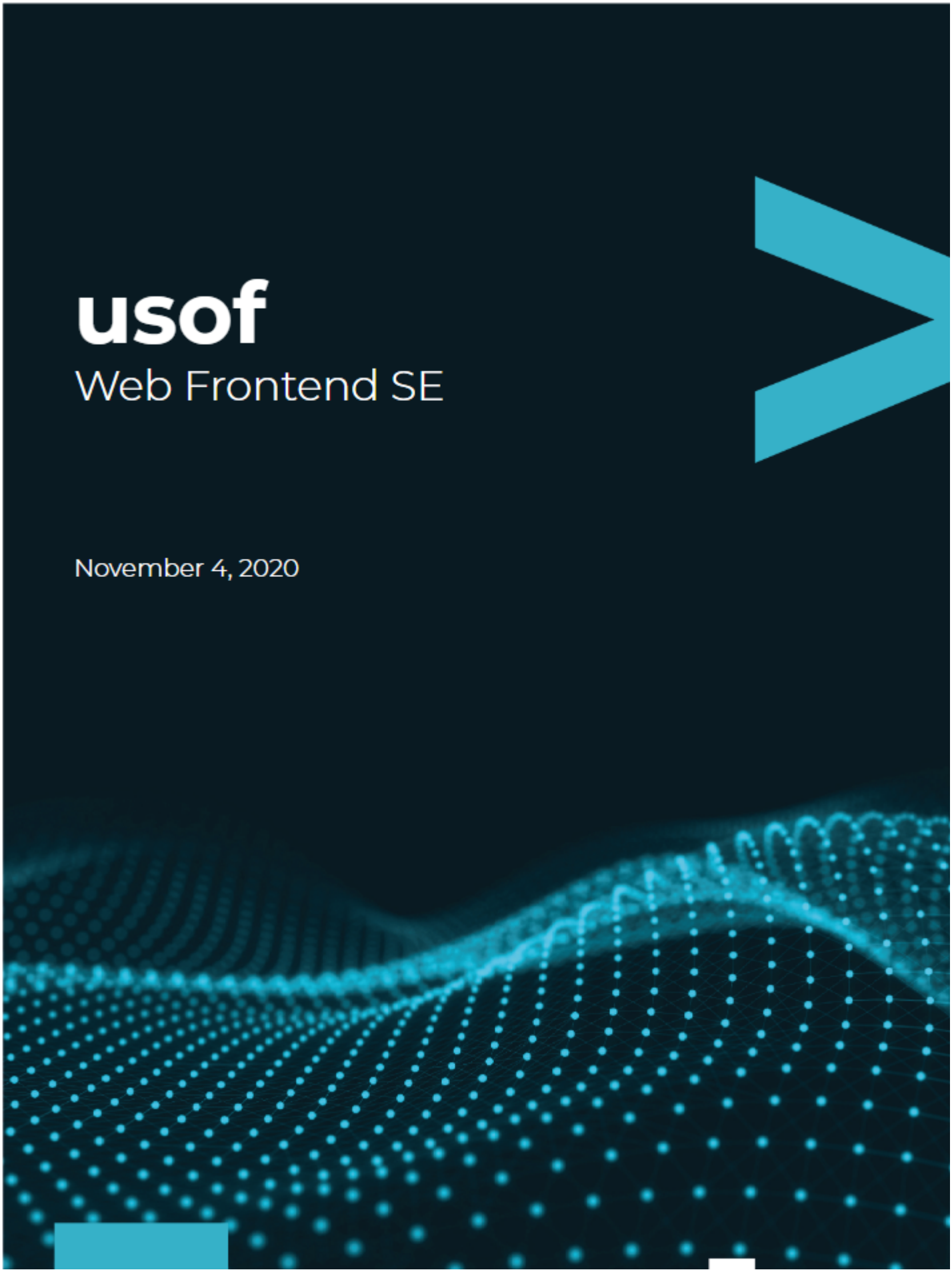

HALLENGE
MEDIA
SLOTS
CLUSTER
TATISTIC


usof
Web Frontend SE
November 4, 2020

Contents


ucode

Engage

DESCRIPTION

Hi, developer!

High quality frontend is quite a challenge.
A good interface is born at the junction of many technologies and approaches. It is an aesthetic design built on the preferences of the target audience and a call to action, a well-thought-out interface and an understanding of the specifics of interaction with the UI/UX software product.

Functionality should serve customer needs and meet customer expectations. When it comes to features, more does not always equal better. Good web design has no features that customers could find annoying. In many cases, a product with few features that do exactly what the user needs can be perceived as higher quality than a product packed with features, some of which have little real value.

A web page that offers good user experience is obvious.
Users have no questions or misunderstandings: every element on the page works as they expect it to.

Of course, a web application needs to have a server side and not just the client side. The beautiful thing about web development these days is that you are not bound by the need to build your own backend. There are numerous trustworthy third-party APIs that frontend developers can use to add content and functionality to their own websites!

BIG IDEA

Interaction with an API.

ESSENTIAL QUESTION

How does the implementation of a frontend for an API differ from a regular website's frontend?

CHALLENGE

Develop an interface for interaction with a third-party API.


ucode
usof | Web Frontend SE > 2

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What is the difference between a single-page app and a multi-page app?
- How does frontend interact with backend?
- Why do many people prefer using frontend frameworks instead of writing pure HTML/CSS/JS code?
- What frameworks and libraries are useful for layouts?
- Why does page load speed matter?
- How can we protect a website from overloads?
- What is github pages? How can you use it?

GUIDING ACTIVITIES

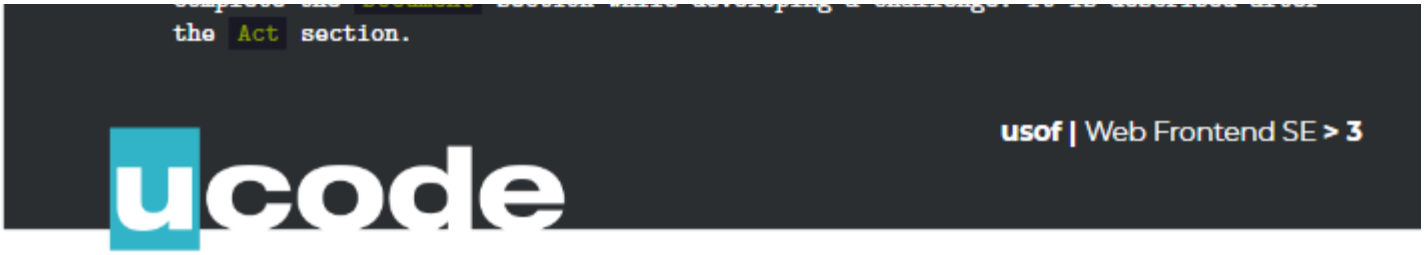
Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Read the story.
- Clone your git repository issued on the challenge page.
- Investigate which tasks take the most time during development.
- Investigate how planning may be helpful for saving time and making error-free web services.
- Compare the frameworks you know. What advantages does React have over others?
- Make sure you have `npm` and the latest version of `React`. Update if you need it.
- Create a new project.
- Run the app to get the default homepage opened in your browser. Complete the configuratio
- For a better understanding of the basics of website design, check out 'The design of everyday things' by Don Norman.
- Start to develop the solution. Strive for excellence. Test your code.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story.
- Analyze all information you have collected during the preparation stages. Try to define the order of your actions.
- Complete the `Document` section while developing a challenge. It is described after


the Act section.
ucode
usof | Web Frontend SE > 3

- The challenge must be performed in `HTML, CSS, JavaScript` with the `React library`.
- Submit all necessary files to your repository with one of the recommended structures.
- If you use a package manager, `package.json` must be at the root of the repository.
- Pay attention to what is allowed. Use of forbidden stuff is considered a cheat and your challenge will be failed.
- Usage of third-party frameworks (apart from React) is considered a cheat and your challenge will be failed.
- You can always use the `Console` panel to test and catch errors.
- Complete tasks according to the rules specified in the following style guides:
  - HTML and CSS: Google HTML/CSS Style Guide. As per section 3.1.7 Optional Tags, it doesn't apply. Do not omit optional tags, such as `<head>` or `<body>`
  - JavaScript:
    * JavaScript Style Guide and Coding Conventions
    * JavaScript Best Practices
    * Airbnb React/JSX Style Guide
- All input fields must be properly validated with relevant error messages.
- Your web app must be responsive and correctly displayed on different screen resolutions (mobile, tablet, etc.).
- Pages must represent real and current data from the API.
- The solution will be checked and graded by students like you. `Peer-to-Peer learning`.
- If you have any questions or don't understand something, ask other students or just google it.


ucode
usof | Web Frontend SE > 4

Act

ALLOWED

HTML, CSS, JS, React, Stack Exchange API, Stack Exchange OAuth library

DESCRIPTION

General information

Create a web app based on the Stack Exchange API using data from stackoverflow.
This means that you must visualize and implement full functionality for some Stack Exchange endpoints.

First steps

Create a simple login/logout system by using the library from Stack Exchange.
Also, setup a repository on GitHub and host your web app using GitHub Pages.

Basics

Every page must contain a header and a menu bar.
The `header` must have the following elements:

- the name of your service
- a search bar for fast navigation across the site
- a login and an avatar of the current user

You can add the menu bar to the header or implement it in a different way.
However, it must be present.
If the user is not signed in, a login button must appear.
Profile image is also a button leading to the personal profile page.

On every page, there must be links for navigating to:

- Main page
- Users page
- Tags page

Main page

This is the first page that every user will see. So, pay maximum of your attention to its quality.
The main page is dedicated to questions. Questions must be shown with the following info:

- title
- tags
- owner's name
- owner's reputation
- publish date and time
- number of votes
- number of answers


• number of views
ucode
usof | Web Frontend SE > 5

- mark if the question has an accepted answer

Everyone must be able to sort questions by

- activity
- votes
- creation

in ascending or descending orders.
Also, every question must have a preview with the information you want.

Users page

Users page must contain... users. And information about them:

- name
- image
- location
- reputation
- favorite tags

Also, a preview of a user's page must be available. In it, more information must appear.

- amount of badges
- user's `about_me` section

Of course, it must be possible to navigate to personal pages by clicking on users' names or images.
And there must be a possibility to sort them by

- reputation
- creation date
- names

in ascending or descending orders.

Tags page

There must be tags and a possibility to sort them by

- popularity
- activity
- name

in ascending or descending orders.

Every tag must contain:

- name
- total number of questions with this tag
- tag's wiki


ucode
usof | Web Frontend SE > 6

A tag's name is also a button leading to the main page with the tagged questions.

Profile page

Implement a `user profile page(s)`. There must be the user's:

- history of reputation changes
- posts
- questions
- answers
- comments
- favorites
- notifications
- privileges
- tags
- badges
- associated accounts
- actions on timeline

Also, every `user` can manage sorting and filtering parameters.

Question page

Every question page must contain:

- title
- body
- score
- tags
- comments
- buttons for upvoting or downvoting
- time and date of creation
- closed date and reason, if a question is closed
- owner's reputation
- owner's name
- owner's profile image
- number of answers
- answers

On such pages, it must be possible to sort answers by

- creation


• activity
ucode
usof | Web Frontend SE > 7

- votes

in ascending or descending orders.

Answer
For every answer, there must be present:

- scores
- body
- comments
- a mark if the answer was accepted
- buttons for upvoting or downvoting
- time and date of answer
- details about the user who answered:
  - reputation
  - name
  - image

An authorized user can:

- accept votes on answers
- undo accept votes on answers
- cast downvotes on answers, questions
- undo downvotes on answers, questions
- cast upvotes on answers, comments, questions
- undo upvotes on answers, comments, questions
- get questions the site considers unanswered within a user's favorite or interesting tags on a special page
- log out on any page of your service

Additions
You must implement `content pagination.`
It can be endless scrolling or pagination with the definition of a specific page.
In any case, search for pagination types and decide which one is more compatible with your needs.

Nesting of your threads must be logical - it must be clear to which questions or answers comments belong.

By default, `comments` must be sorted by creation date in ascending order.

All dates and times must be in a human readable format, not in unix time.


ucode
usof | Web Frontend SE > 8

SEE ALSO

- Framework:
  - React
- Package managers:
  - npm
  - yarn
- Boilerplate:
  - Wiki
  - yarn
- Other:
  - Flexbox
  - TinyMCE


ucode
usof | Web Frontend SE > 9

Document

DOCUMENTATION

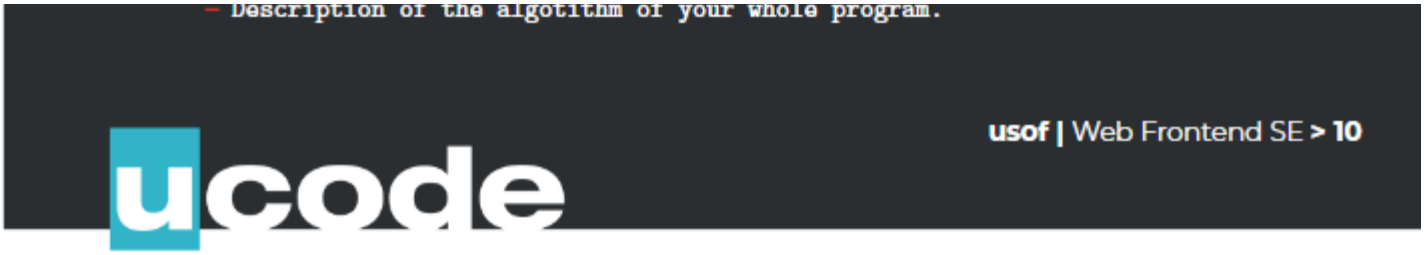
One of the attributes of Challenge Based Learning is documentation of the learning experience from challenge to solution. Throughout the challenge, you document your work using text and images, and reflect on the process. These artifacts are useful for ongoing reflection, informative assessment, evidence of learning, portfolios, and telling the story of challenge. The end of each phase (Engage, Investigate, Act) of the challenge offers an opportunity to document the process.

Much of the deepest learning takes place by considering the process, thinking about one's own learning, analyzing ongoing relationships with the content and between concepts, interacting with other people, and developing a solution. During learning, documentation of all processes will help you analyze your work, approaches, thoughts, implementation options, code, etc. In the future, this will help you understand your mistakes, improve your work, and read the code.

At the learning stage, it is important to understand and do this, as this is one of the skills that you will need in your future job. Naturally, the documentation should not be voluminous, it should be drawn up in an accessible, logical, and connected form.

So, what must be done?

- a nice-looking and helpful README file. In order for people to want to use your product, their first introduction must be through the README on the project's git page. Your README file must contain:
  - Short description. This means, that there must be some info about what your project actually is. For example, what your program does.
  - Screenshots of your solution. This point is about screenshots of your project "in use".
  - Requirements and dependencies. List of any stuff that must be installed on any machine to build your project.
  - How to run your solution. Describe the steps from cloning your repository to the first launch of your program.
- a full-fledged documentation in any forms convenient for you. By writing this, you will get some benefits:
  - you have an opportunity to think through implementation without the overhead of changing code every time you change your mind about how something should be organized. You will have very good documentation available for you to know what you need to implement
  - if you work with a development team and they have access to this information before you complete the project, they can confidently start working on another part of projects that will interact with your code
  - everyone can find how your project works
- your documentation must contain:
  - Description of progress after every competed CBL stage.


- Description of the algorithm of your whole program.
ucode
usof | Web Frontend SE > 10

Keep in mind that the implementation of this stage will be checked by peers at the assessment!

Also, there are several links that can help you:

- Make a README
- How to write a readme.md file?
- A Beginners Guide to writing a README
- Google Tools - a good way to journal your phases and processes:
  - Google Docs
  - Google Sheets
- Dropbox Paper - a tool for internally developing and creating documentation
- Git Wiki - a section for hosting documentation on Git-repository
- Haroopad - a markdown enabled document processor for creating web-friendly documents
- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio
- code commenting - source code clarification method. The syntax of comments is determined by the programming language
- and others to your taste


ucode
usof | Web Frontend SE > 11

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Telegraph - create a post that you can easily share on Telegram
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.


ucode
usof | Web Frontend SE > 12