MAGEBIT

Web Developer Test

The purpose of this test is to evaluate the knowledge of the candidate in various areas that are related to the daily work of a Web Developer at Magebit. Please try to use the latest technologies, and be as accurate as possible with the formatting.

Even while the test task has been split into multiple sections, that's only the suggested order of how you should implement it. The final result is a **single** application rather than 3 separate ones.

Contents

Rules:

1. As the tasks listed match with the job responsibilities you would have to fulfil at work on a daily basis, **we encourage you to complete the test by yourself without asking anyone for help**.
2. As we are willing to test your knowledge, please avoid using PHP frameworks or Bootstrap as it will be harder to determine your skills in PHP, HTML, CSS, however, you are more than welcome to use CSS preprocessors, Vue.js, React or other similar technologies.
3. In each task, you will find "Requirements" that list points mandatory for completion, and "Advantages" that are not mandatory but can be completed by candidates who can and want to get some extra points.
4. ReadMe file has to be in place with instructions on how to run the project locally.
5. Once the test is completed, please upload the results to GitHub and share the link with us by filling-out the Google form.

Good luck!

Task 1 - HTML / CSS

Create an HTML/CSS page based on the design provided below.
In this task, you must show your knowledge of HTML standards and proper structure. We will also test your skills on writing rendering efficient CSS.

Requirements:

- The page is as pixel-perfect as possible. You can use this Chrome extension to test it.
- Bootstrap or similar frameworks are not used.
- The page is responsive for all devices.
- All URLs on-page is "#". They do not lead anywhere.
- The form submission functionality is not part of Task 1.
- Social icons have hover effects as per style guides.
- Hover effects are present on the site as per style guides.

Advantages:

- A font with custom icons is used instead of separate icon files.
- LESS, SASS or any other CSS preprocessor is used.

Design:

- Design link on Figma
- Style guide link on Figma

Task 2 - JavaScript

Based on Task 1 you have to create subscription input validation.
In this task, you must show your knowledge about writing fast and memory-efficient JavaScript.

Requirements:

- There is an error message under input that shows validation messages if:
    - Invalid email is added - "Please provide a valid e-mail address"
    - The checkbox is not marked - "You must accept the terms and conditions"
    - No email address is provided - "Email address is required"
    - Provided email is ending with .co - "We are not accepting subscriptions from Colombia emails".
- Once validation has passed, the error disappears.
- The button is disabled if the form is not valid.
- On successful validation, a success message appears in the place of the form, as per design.

Advantages:

- You can use Vue.Js, React or similar technologies.

Design:

- Design link on Figma
- Style guide link on Figma

Task 3 - PHP / MYSQL

Based on Tasks 1 and 2, you have to create BE functionality for an email subscription.
In this task, you must show your knowledge about PHP OOP standards, PHP best practices, and security.

Requirements:

- Data that is submitted is saved in a MySQL database.
- Adjust the functionality from previous Task so that success message appears only after form submit instead of successful validation.
- Data is validated also in PHP, and if JavaScript is disabled it displays errors in the same place and style as it is with JavaScript enabled (exceptions apply to React and Vue headless implementations).
- A page is created where all saved data can be seen:
    - No login for this page is needed.
    - No styling for this page is needed. Basic HTML table is enough.
    - Sorting by name and date is added (by default it sorts data by date)
    - It is possible to filter email addresses by email providers:
        - Example: If under subscriptions there are 3 Gmail, 5 Yahoo and 2 Outlook email addresses, 3 buttons should appear: Gmail, Yahoo and Outlook. Once new subscriptions from a different provider appear, for example, HubSpot, this leads to the automatic appearance of a new button which will allow to filter out all the email addresses with the email provider that have @hubspot.com at the end.
    - It is possible to search for email addresses and use filters and sorting at the same time.
    - It is possible to delete email addresses.

Advantages:

- It is possible to select multiple emails using checkboxes and export them as CSV.
- Pagination is present on the list with 10 emails per page.

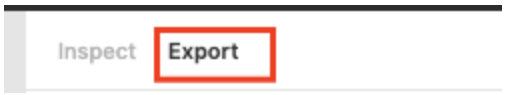
After test completion, please remember to fill in the form & set up a working test site on your test instance (was shared with you per email together with this test).

FAQ

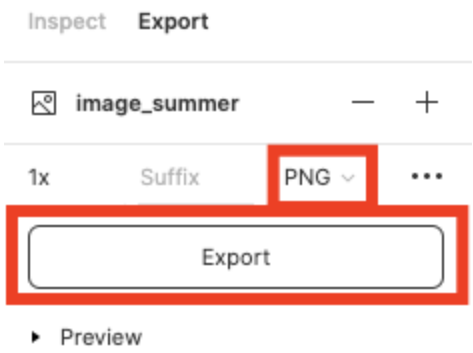
Q: How to extract files from Figma?
A:

1. Login in Figma. If you don't have an account you can create it. It's free.
2. On the left side, you need to find the file that you are willing to export -

3. On the right side, you can find now the Export tab.



4. Now you can change the file format and export the file



Q: Should I send each task separately or all of them together?
A: Goal of the test task is not to have multiple applications. Result at the end is just a single application that contains all the steps/tasks.

Q: Can I add one more option in the nav bar for the link to easily access the data display page?
A: Can be added but it's not necessary, it's fine if the route should be visited manually.

Q: Can I use react routing in order to navigate to the data display page?
A: React routing can be used, and should be if you have chosen to implement a task in React.

Q: Can I use Angular even if it's not mentioned in the list of libraries that can be used?
A: You can but if you have chosen to use a frontend library we'd prefer to see Vue or React as those are the ones we tend to use in projects we work on.

Q: What should I write in the README file?
A: There are no strict guidelines on what can and what can not be written in the README file, but as long as it contains information on how to run the project locally it would be fine. Keep in mind that not everyone is using the same OS or software, so the more generic the guide is the better.

Q: How should the application be structured?
A: Most common and simplest way to build the applications is to use the MVC pattern. You don't need to reinvent the wheel, yet some form of basic view and business logic separation would be enough.

Q: How to access the server where the test should be set up?
A: You have received the htpasswd details in the email together with the test task.

Q: How to connect to the server to upload and run my test task?
A:

- One way would be to use the terminal and connect to the server via SSH. SSH connection details were sent together with the test task. Then using GIT you can clone your repository on the server. Afterwards you'll have to adjust the Nginx (or if you prefer - Apache) configuration for the web server to function correctly.
- You can use SFTP client to upload code from your local machine, yet you'll still need to connect to the server via SSH afterwards for web server configuration (Nginx or Apache). If you are using an SFTP client make sure you use port 22.

Q: Where should the project be uploaded on the server?
A: Usually web projects are located in the /var/www directory. For your own project, you can create a new directory within.

Q: What IDE/code editor should be used for the completion of the task?
A: It doesn't matter, can be any editor as long as you feel comfortable using it. Most of the new programmers tend to use VSCode as it's free of charge. More advanced IDE would be PHPStorm but for that you have to pay the license fee.