

PROJECT 1

ØYSTEIN AUNE SVERRE - S326287@OSLOMET.NO

Contents

Section 1.....	3
Case.....	3
Solution	3
Section 2.....	4
Design.....	4
Technical layout	4
Client page	5
Panel page.....	6
Additional feature	6
Relevance	7
Future implementation.....	7
Section 3.....	8
Approach.....	8
Complications.....	10
Solution	10
Feasibility	11
Revisiting the design	11
Section 4.....	12
Analysis	12
Positives	12
Negatives.....	12
Conclusion.....	13
Appendix	14
Code	14
index.html – front page	14
index.js – front page	15
index.html – control panel.....	16
index.js – control panel.....	18
get_features.js	19
set_features.js.....	19
get_features.php.....	21
set_features.php.....	21
features.json	21
get_sales.js.....	22
get_status.js	22

login.js	24
search.js	24
Assets	24
main.css.....	24
banner.jpg	27
favicon.ico	27
Screenshots	28
Front page	28
Control panel.....	29
Configuration files	29
Nginx (sites-enabled)	29

Section 1

Case

Catapult Hosting is a web hosting company. They are hosting a big website and accountable for its functionality. The website is in constant development and is delivered new code on a regular basis due to a competitive environment. The company uses a complex Docker system to host the website with many different features and components.

With constant, high speed development comes big risks. Catapult hosting is concerned a bug in new code will take down the entire webpage. This has happened in the past and it takes considerable effort to track down the responsible engineer to fix the problem. The only quick solution is to roll back the page to a previous version, but this is not desirable as this would slow down progress, cause irritation for the customer and possibly disable working features.

Solution

Catapult Hosting wanted to move their system towards a more “fine grained” mechanism where features can be controlled using feature-flags. Feature-flags inform the webpage of whether to enable certain features or not, based on the implementation. If a feature misbehaves it can be disabled and the site should function without it. For example, a malfunctioning search function could be disabled until the problem is resolved, allowing the site to still be functional. This would remove the need to roll back what version is running.

This resolution was still in an early phase and an investigation into this solution was desired to show engineers how this would work in practice. The Swiss Cheese model proposes a point of view that all systems are flawed, but with sufficient layers of precautions, these flaws are dealt with. This applies even though all these layers have weaknesses themselves. The layers just need to be accurate enough to prohibit certain scenarios, not allowing the “holes in the swiss cheese” to line up, causing the least amount of damage.

The feature-flag system would not automatically prohibit mistakes, but act as a defense against their repercussions. This would mean a task force could disable features and develop fixes to broken code, while the site functions as normal and development is continuing. The goal is to investigate if this is a valuable and feasible solution for Catapult Hosting.

Section 2

Design

The goal is to create a working prototype that shows the benefits feature-flags can have on Catapult Hosting's system. To display this concept, the prototype should have two interfaces:

1. A webpage that is being hosted by Catapult. This should have features available for the user to interact with or read information from. This is necessary to display a change on the webpage when Catapult use the feature-flags to alter how the page operates.
2. An interactive panel where Catapult Hosting can configure the feature-flags for the webpage they are hosting. It should be easy to use and include enough information for the administrator to understand what needs to be done when interacting with the panel.

In this prototype Catapult is hosting an internal, employee webpage for "Component LLC" that has several features for the customer to use. This is for example automatically updated sales figures from the last 24 hours, a search engine, a log in system and service status. These systems are maintained by Catapult and can be disabled by Catapult when they find it necessary for the stability of the webpage. This webpage receives feature-flag information from an API. The features in the prototype does not need to be complex but need to show how they are changed by the feature-flag system.

Catapult has their own administrator page that can interact with the feature-flag API through a panel. This is also a webpage, because of ease and flexibility of HTML when they need to expand or change the panel. Being a webpage, they can choose if it should be accessible outside their employee network. In its fully operational shape this panel should have a login system for administrators, but this is not planned to be operational in the prototype. The panel should show all features, their status and have a form to change features easily, causing Component LLC's site to change.

Technical layout

The API is in this prototype a CouchDB database delivering JSON statements to a PHP API. The API uses PHP for increased security by executing database-calls on a local server instead of a client's webpage and hiding connection information from local running code. The HTML webpages uses JavaScript to request and send information to the PHP API. The PHP then connects to the CouchDB database, receiving or changing information.

As can be seen in the figure 1 below, information is sent between the database and the two pages using the API, but only the admin panel has the option to send data to the database to change it. Component's website can only read information from the API so it can decide what features to enable on the webpage, it cannot change anything on the database.

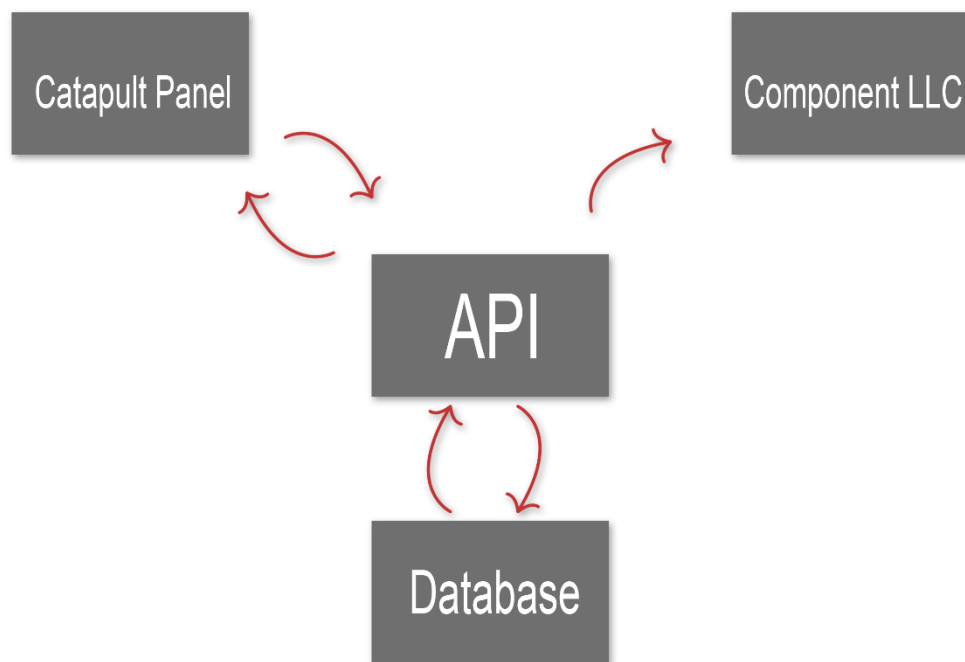


Figure 1 – A diagram showing how information is exchanged between the database, Component LLC and the Catapult admin panel. Component is only receiving information; however, all other parties can receive and send information through the API.

Client page

The Component webpage code need to take feature flags into consideration when building the webpage. Much like making JavaScript functions, different features will be put into different JavaScript files and loaded based on information from the API delivery. This should decrease data transfer and system impact when features are disabled as big or complicated JavaScript code is simply not loaded by the page. The steps are the following:

1. The HTML page, index.html, loads index.js. This is the main JavaScript code that has basic functions and automatically loads feature flags from the API. This is all the code index.html needs to be functional. All other code that is not yet loaded is relevant to features alone.
2. Then if/else statements load different JavaScript files with features based on information given from the API. The main information this is based on is feature status. If the feature status for the sale figures feature is “false” then index.js will skip loading the get_sales.js entirely.

As can be seen in the sketch on figure 2 below, the page design is simple, but modern, with a focus on features as their relevance to the feature flags is the most important part of the prototype. All features are not always as separated into modules in the real world, but for a proof of concept, this should easily show how the admin panel can alter the features on the connected page.

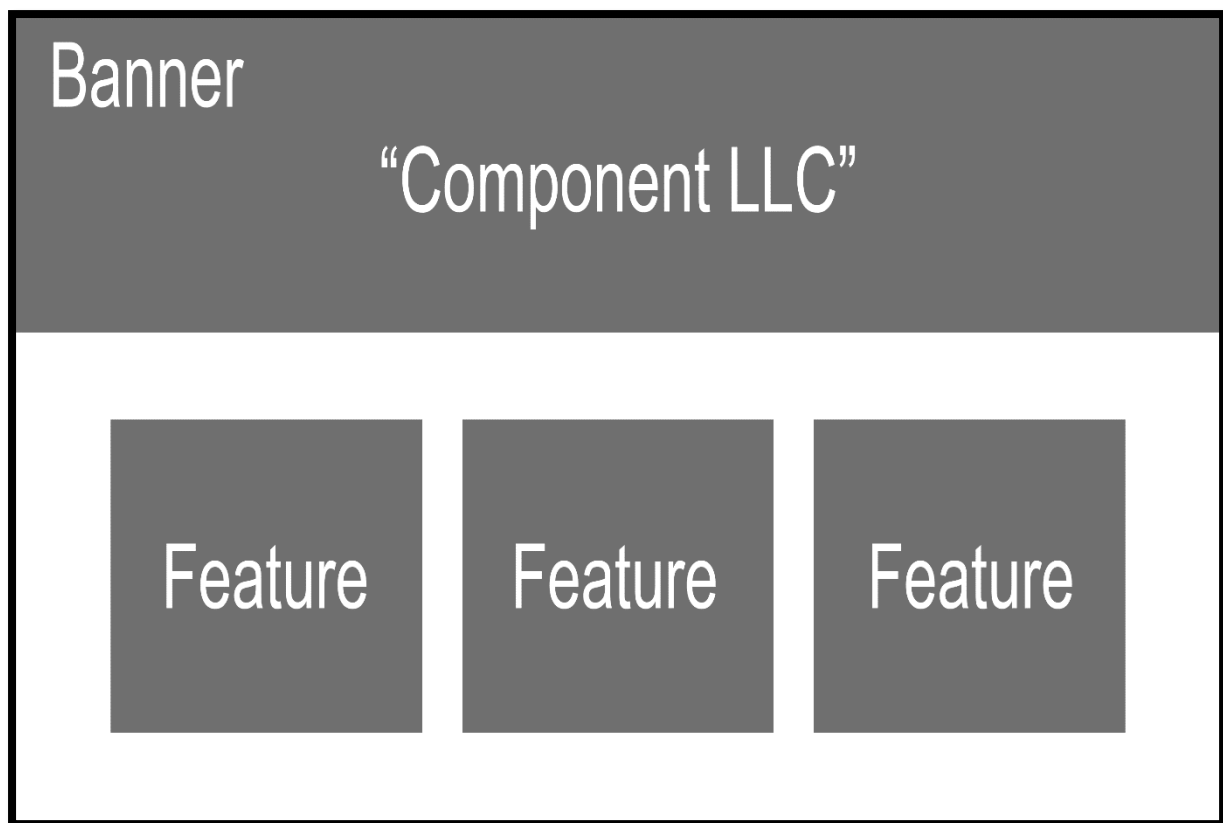


Figure 2 – A sketch of the website. The top has a banner with the company name. The space underneath has three modules on a line, reserved for features.

Panel page

Much like the client page, the admin panel loads feature-flags from the API. Unlike the client page, the panel loads every feature and their status to show their current state to the administrator interacting with the panel. This also includes more information like feature name, error messages, status, and comments. This information is shown in modules like how features are displayed on the connected site.

At the end of the page is a form that can be used to change feature status in the database. This form is in its own module and allows the administrator to change the features and their settings. The panel has additional PHP API that can send information to the database, allowing the panel to change how the connected sites operate.

Additional feature

A feature-flag system comes with an additional possibility that Catapult could make use of. In the database a feature can be on, off or a third option. This is a “shadow” option that is used to test performance on clients and servers when new features are created. When a feature status is set to “shadow” it will load the relevant JavaScript file and execute the code, but the results will not be shown. To the client the page will look like the feature is not enabled.

This is useful for testing performance on clients and servers when new features are implemented. When something is having a heavy impact on the system it can be discovered and dealt with without having to break or interrupt anything.

Relevance

An approach the like one in this design would in theory greatly increase Catapult Hosting's flexibility when constantly developing the webpage for their client. Errors should still be avoided as much as possible, but the frustration of downtime will be limited to the affected features and not the whole webpage. It does not affect other solutions, like version rollback, so it would not limit Catapult's approaches to unexpected issues.

The shadow feature will also be useful for avoiding the necessity of disabling features, as it implements some easy testing into the platform. It is also optional so if it feels unnecessary or too time consuming it can be ignored in favor of other solutions.

Future implementation

In this design the HTML on the admin page is hardcoded and must be updated according to features being added, but if the output from the feature flag API is standardized this could be dynamically built around existing feature flags. This increases effectiveness when coding and reduces risks of mistakes when constantly adding and removing features to the panel. Less time is used administrating and more is used staying competitive.

The shadow feature is useful, but further testing tools could be implemented. Currently all visitors of connected sites receive the same settings, but a more complex integration would mean more reliable testing. A slow rollout of features could provide useful data that could prevent big problems. This can be done by sending certain feature settings to certain users, either using percentages, regions or account-type.

The system itself would need built in a more secure environment when implemented in the real world. A user system is especially necessary for the admin panel when changing information in the database and the panel does not need to be a public URL even though it is a webpage. This page's access could be restricted to a local internal network at Catapults worksite to increase security, in addition to a user system for logging changes to the database.

The API design is using a simple CouchDB database, but for safety this should be a suitable database with proper authorization to reduce security risks. It is important the feature-flag feature does not add more obstacles to the platform, causing more harm than good.

Section 3

Approach

The prototype starts out by adding the essentials. A webserver using Nginx running on a Docker container. Then Nginx is configured to use PHP 7.4 FPM to process PHP code for the website, and a folder in the container is chosen as a directory for files. When the webserver is showing files in the directory and correctly processing PHP scripts, the API is ready to be created.

API scripts created using PHP functions as a delivery system for premade JSON outputs for testing the JavaScript implementation on the HTML page. JavaScript send calls to the PHP script and receives a JSON message with information. The JavaScript then uses the JSON information to build the website appropriately. These premade JSON outputs allow the codes to create the website without having to finish the database first.

To create the right order of execution, JavaScript is split into three different parts, loaded based on necessity. The first JavaScript file is named index.js and is automatically loaded by the HTML. This includes essential code that the site needs to function. These are not effected by feature-flags and declares methods and variables needed later. This script decides, when ready to load get_features.js.

The JavaScript file get_features.js commits a call to the PHP API using AJAX and processes the return. The features, received in the JSON format, is then saved to be used later. This allows the index.js script to continue based on what features it should enable.

All features are separated and saved in individual JavaScript files. These files are called by the index.js script based on the enabled features from the get_features.js script. The feature loading is shown by the if/else statement shown in line four. As can be seen on line five, they are executed when the feature is enabled.

```
1. $.getScript("get_features.js", function(){
2.     if(features.error == "false") {
3.
4.         if(features.feature[0].status != "false") {
5.             $.getScript("get_sales.js", function(){
6.                 });
7.         } else {
8.             replace('sales', feature_error);
9.             console.log("sales not loaded.");
10.        }
```

Figure 3 – Part of the JavaScript executed when the site starts to load. On line one the features are loaded. Then an if/else statement on line four determines if the feature should be loaded on line five.

The different features located in separated files also use information from the feature flags to determine how they execute. If the feature is set to shadow it will execute all code, calculating and loading all relevant information, but the final line that displays the result is not completed. In the example below, on line four, it instead tells the user that the feature is disabled. This allows the script to complete before the results are cancelled.

```

1. if(features.feature[0].status == "true") {
2.     replace(id, text);
3. } else if(features.feature[0].status == "shadow") {
4.     replace(id, feature_error);
5. } else {
6.     replace(id, error);
7. }

```

Figure 4 – On line three, in this example from a feature script, we can see that if the feature is set to “shadow” it will not display the results of the code on line four.

The result for the user is modules that have one of two states visually. Even though the feature could be in shadow mode, that is never told to the user except for in the developer console. In the screenshot below you can see the sales module in enabled and disabled mode, going left to right. The third option, a shadow feature, would be displayed just like the example on the right, but it will also cause a bigger load on the system because it executes all code in the feature.

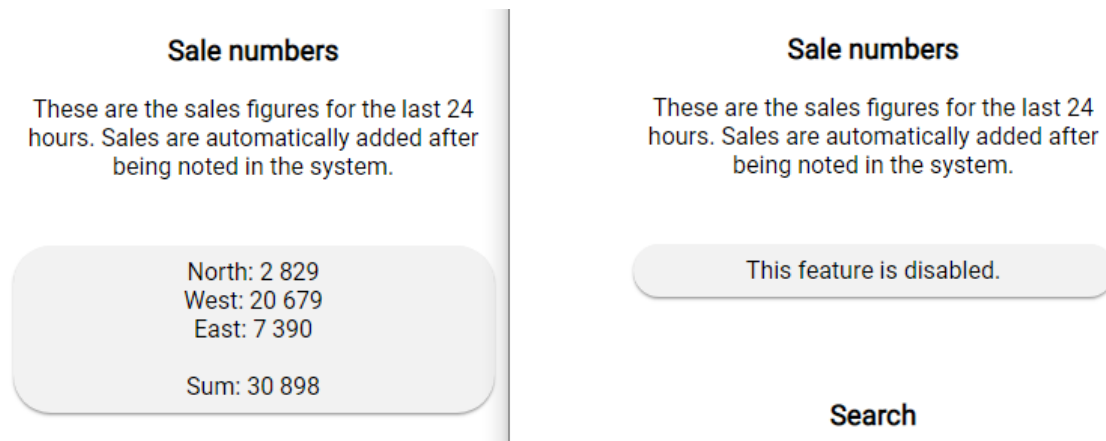


Figure 5 – The same module on the website displayed in two different modes. The left example has sales displayed, the one on the right says the feature is disabled. The shadow mode would also display the right option.

The administration panel also loads the features, but displays all information received in modules. This makes it possible for the administrator to read necessary details about the features. At the end of the web page is a form that informs the API to change the configuration of the features. This allows the admin to change feature status when needed. In the screenshot below it is demonstrated by two modules. The one on the left displays a feature-module, showing different details like feature-name, status, and comments. The module on the right has a form for the same information where the dropdown input allows the administrator to change different integrated features and status-modes.

Log in status

The log in function on the website.

Name:

Error:

Message:

Status:

Comment:

Changes feature status

Use this for to change the status of features.

Feature:

Error:

Message:

Status:

Comment:

Figure 6 – This screenshot shows two modules. The first one indicates the status of the `log_in` feature, which is currently in shadow mode. The second one is the form for changing feature-status. It has the same details as module, but option to change them using inputs and dropdown menus.

Complications

The plan was to have an API that calls to a database for ease of use and increased security for the system. The database was created, but there was not enough time to implement functionality for the prototype caused by poor planning. Creating the system was prioritized and not enough was set aside to learn CouchDB and therefore the functionality of the database was not fulfilled. Databases require some discretion so if this solution is implemented it must be taken into consideration. This was the result of creating the platform first using premade API outputs, instead of creating a functional database first.

Solution

The complexity of the database is easily avoided by using a local configuration file. As the JSON message were already defined for testing purposes, this just needed to be read from a JSON file. The API for changing feature settings is then changed to write to that JSON file. This avoidance of the intricate database solution does lose some features like increased security and availability, but in turn does simplify the approach. The configuration file is local, so would not be affected by a database issue and requires less upkeep.

In the example seen below you can see the PHP API read a local file called `features.json`. This is the dedicated file storing feature settings and it is shared by the website and the administration panel.

```
1. $features = utf8_encode(file_get_contents('features.json'));
```

Figure 7 - PHP code that reads a local JSON file containing feature-settings.

In the figure underneath you can see how the PHP API processes data from the administration panel and saves it to the local file. The first lines (one through five) replace the targeted data in the current JSON configuration, and then saves it on line eight. This then replaces the original file which is shared by the connected site.

```
1. $features->feature[$index]->name = $data->f_name;
2. $features->feature[$index]->error = $data->f_error;
3. $features->feature[$index]->message = $data->f_message;
4. $features->feature[$index]->status = $data->f_status;
5. $features->feature[$index]->comment = $data->f_comment;
6.
7. $fp = fopen('./features.json', 'w');
8. fwrite($fp, json_encode($features));
9. fclose($fp);
```

Figure 8 - PHP script changing local feature-settings file. Lines one through five changes the current configuration and this is then saved on line eight.

Feasibility

This solution is feasible, as shown in this report. It is demonstrated to be functional using a local configuration file and it is theoretically possible using a database. The feature-flags gives the website more flexibility if issues appear and the shadow mode is useful for testing, meaning turning off features is even less likely. This prototype is built from scratch, with no insight into Catapult Hosting's work process, current setup, or programming language preferences. The feasibility of its implementation is very dependent of how hard it would be to integrate. It could be a long process, but we know the result is increased stability and testing opportunities.

Revisiting the design

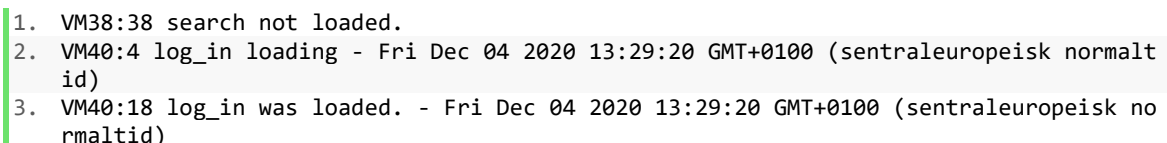
The final prototype does accurately follow the design, even if the approach for saving feature-flag settings was changed. Designing a website that can change the settings of another website is very doable and control panels are very common on the world wide web. This knowledge meant the decision to use JavaScript to interact with a PHP API was very approachable and resulting in no issues. The final API is simple with few security measurements, but the design in this prototype did not take this into account.

Section 4

Analysis

The prototype built in this report clearly shows how a solution like this can benefit Catapult Hosting if used correctly. The goal is to increase stability and allow more time to be spent staying competitive. A feature-flag system does all this and comes with the additional option of the “shadow”-mode, meaning more testing opportunities.

Multiple tests of the feature-flag system were performed, resulting in changing the connected website based on the settings applied to the control panel. The connected website only showed the two different states, but the developer console displayed logs of what was executed on the client side. In the example below you can see the site disabling the search function, not loading the script at all. The second line starts the loading of the log_in feature, executing the code within the function. The feature itself takes into consideration that it is in shadow mode and it ultimately also shown as disabled when finished on line three.



```
1. VM38:38 search not loaded.  
2. VM40:4 log_in loading - Fri Dec 04 2020 13:29:20 GMT+0100 (sentraleuropeisk normaltid)  
3. VM40:18 log_in was loaded. - Fri Dec 04 2020 13:29:20 GMT+0100 (sentraleuropeisk normaltid)
```

Figure 9 - Log from developer console. Line one displays a feature being skipped, line two and three display a feature being loaded, the time it takes to load it, and it is ultimately not shown because the feature is in shadow mode.

Positives

This is an efficient way to improve reliability and opens for more time spent staying competitive. Standardizing coding methods and increasing control of features could also mean the code is cleaner, resulting in more effective coding and possibly less errors.

Even though the planned database was scrapped, and the prototype ended up utilizing a local configuration file, this could be argued as a better approach. The file is saved locally, and the security measures needed to change files on the webserver applies to the configuration. This results in the file only being editable by the administration panel if webserver folder-access is restricted. If there is a database outage, the websites would still work as before, able to receive feature settings from the local file.

Negatives

A solution like this prototype is functional, but it should be taken into consideration what changes to workflow could follow. When writing features and creating a platform the coder would need to write according to feature flags and even if it is beneficial, it could be slower depending on the current process. Existing features would need rewritten to take advantage of the new system and the whole process would take time. To know if it is worth the effort one must evaluate the current system and the need.

The feature flag system is useful when determining how the site should behave, but it is important to keep in mind that this is itself a system and could break. If the feature-flag system goes offline and

for some reason this scenario is not considered, that could result in all customers losing access to all features. The feature-flags create increased reliability but does not remove the possibility of mistakes.

Using a local configuration file to keep the settings also has the downside of being harder to sync across websites. If the feature is provided by Catapult themselves to multiple customers, it would be important to sync the availability of the features on all sites. This could mean there are multiple administration panels needed to turn of certain features for different sites.

This could also apply to features that are co-dependent. It is common to build features and methods that use each other to increase efficiency. This can become problematic if not considered when creating the feature-flag system. Disabling a feature could result in two being disabled if they need each other. This would not appear as an intentional action and can destabilize the platform.

Conclusion

The solution is feasible to implement but has many ways of being designed. The layout in this report is valuable, working and could be of great benefit to Catapult Hosting, but is not necessarily functional depending on their current setup. This comes with the additional choice of choosing a local configuration file versus a database. Regarding the Swiss Cheese model, this solution does have its own flaws, but it also blocks flaws in the system already in place by Catapult Hosting, resulting in a system that is less likely to fail. If added correctly it will benefit the system.

The approach used in this report should be implemented with a couple of questions in mind.

- Should the features be accessed through a configuration file or a database. Both options come with benefits and downsides.
- The feature-flag method is a system that needs maintenance and supervision. This is especially true if there is an API using a database to change multiple sites.
- It could be the whole platform currently in use must be rewritten with this new system in mind.
- Whatever approach, security must be taken into consideration. Allowing outsiders to change features could be dangerous and a proper login system is necessary.

Appendix

Code

index.html – front page

```
1. <!doctype html>
2.
3. <html lang="no">
4.
5.     <head>
6.         <!-- Required meta tags -->
7.         <meta charset="utf-8" />
8.         <meta name="viewport" content="width=device-width, initial-scale=1, shrink-
to-fit=no" />
9.
10.        <title>Component LLC</title>
11.
12.        <!-- Bootstrap 4 CSS and custom CSS -->
13.        <link rel="stylesheet" type="text/css" href="assets/css/main.css" />
14.        <link rel="shortcut icon" href="assets/favicon.ico" />
15.        <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" r
el="stylesheet">
16.    </head>
17.
18.    <body>
19.        <div id="alert"></div>
20.        <div class="banner">
21.            <h1>Component LLC Internal Site</h1>
22.            <h2>A company that does stuff</h2>
23.        </div>
24.
25.        <div class="box">
26.
27.            <div class="unit">
28.                <div class="info">
29.                    <h3>Log in</h3>
30.
31.                    <p>Log into your company user to view the items on this page.</
p>
32.                </div>
33.                <div class="log_in" id="log_in">
34.                    Not loaded...
35.                </div>
36.
37.            </div>
38.
39.        </div>
40.
41.        <div class="box">
42.
43.            <div class="unit">
44.                <div class="info">
45.                    <h3>Sale numbers</h3>
46.
47.                    <p>These are the sales figures for the last 24 hours. Sales are
automatically added after being noted in the system.</p>
48.                </div>
49.                <div class="sales" id="sales">
50.                    Not loaded...
51.                </div>
52.            </div>
53.
54.            <div class="unit">
55.                <div class="info">
```

```

56.         <h3>Search</h3>
57.
58.         <p>Search items on company database</p>
59.     </div>
60.     <div class="search" id="search">
61.         Not loaded...
62.     </div>
63. </div>
64.
65.     <div class="unit">
66.         <div class="info">
67.             <h3>Service status</h3>
68.
69.             <p>Here you can view the current status of our services. If the
re is a problem it will be visible here. You can also read the newest updates on cu
rrent situations.</p>
70.         </div>
71.         <div class="status" id="status">
72.             Not loaded...
73.         </div>
74.     </div>
75.
76. </div>
77.
78. <div class="footer">
79.     <h4>Component LLC<br>All rights reserved and stuff</h4>
80. </div>
81.
82. <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
83. <script type="text/javascript">
84.
85.     $(document).ready(function() {
86.
87.         $.getScript("index.js", function(){
88.             });
89.
90.     });
91. </script>
92.
93. </body>
94.
95. </html>

```

index.js – front page

```

1. function replace(div, string) {
2.     $('#'+ div).html(string);
3. }
4.
5. function number_format(number) {
6.     var format = number.toString().replace(/\B(?=(\d{3})+(?!\d))/g, " ");
7.     return format;
8. }
9.
10. var feature_error = "This feature is disabled.";
11. var api_error = 'Could not reach API.';
12. var error = "There was an error.";
13.
14. $.getScript("get_features.js", function(){
15.     if(features.error == "false") {
16.
17.         if(features.feature[0].status != "false") {

```



```

18.         $.getScript("get_sales.js", function(){
19.             });
20.     } else {
21.         replace('sales', feature_error);
22.         console.log("sales not loaded.");
23.     }
24.
25.     if(features.feature[1].status != "false") {
26.         $.getScript("get_status.js", function(){
27.             });
28.     } else {
29.         replace('status', feature_error);
30.         console.log("status not loaded.");
31.     }
32.
33.     if(features.feature[2].status != "false") {
34.         $.getScript("search.js", function(){
35.             });
36.     } else {
37.         replace('search', feature_error);
38.         console.log("search not loaded.");
39.     }
40.
41.     if(features.feature[3].status != "false") {
42.         $.getScript("log_in.js", function(){
43.             });
44.     } else {
45.         replace('log_in', feature_error);
46.         console.log("log_in not loaded.");
47.     }
48.
49. } else {
50.     alert("Website could not reach feature flag API. No features will work. Con-
    tact your administrator.");
51. }
52. });

```

index.html – control panel

```

1. <!doctype html>
2.
3. <html lang="no">
4.
5.     <head>
6.         <!-- Required meta tags -->
7.         <meta charset="utf-8" />
8.         <meta name="viewport" content="width=device-width, initial-scale=1, shrink-
    to-fit=no" />
9.
10.        <title>Component LLC</title>
11.
12.        <!-- Bootstrap 4 CSS and custom CSS -->
13.        <link rel="stylesheet" type="text/css" href="assets/css/main.css" />
14.        <link rel="shortcut icon" href="assets/favicon.ico" />
15.        <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" r
    el="stylesheet">
16.    </head>
17.
18.    <body>
19.        <div id="alert"></div>
20.        <div class="banner">
21.            <div class="login">Logged in as: Admin</div>

```

```

22.         <div class="title">
23.             <h1>Component LLC Feature Flags</h1>
24.             <h2>Catapult Hosting admin page</h2>
25.         </div>
26.     </div>
27.
28.     <div class="box">
29.
30.         <div class="unit">
31.             <div class="info">
32.                 <h3>Sales API</h3>
33.
34.                 <p>The API transferring sales numbers to the website.</p>
35.             </div>
36.             <div class="sales" id="sales">
37.                 Not loaded...
38.             </div>
39.         </div>
40.
41.         <div class="unit">
42.             <div class="info">
43.                 <h3>Status API</h3>
44.
45.                 <p>Status panel showing what services is running for Component
LLC.</p>
46.             </div>
47.             <div class="status" id="status">
48.                 Not loaded...
49.             </div>
50.         </div>
51.
52.         <div class="unit">
53.             <div class="info">
54.                 <h3>Search status</h3>
55.
56.                 <p>The search function on the website.</p>
57.             </div>
58.             <div class="search" id="search">
59.                 Not loaded...
60.             </div>
61.         </div>
62.
63.     </div>
64.
65.     <div class="box">
66.
67.         <div class="unit">
68.             <div class="info">
69.                 <h3>Log in status</h3>
70.
71.                 <p>The log in function on the website.</p>
72.             </div>
73.             <div class="log_in" id="log_in">
74.                 Not loaded...
75.             </div>
76.         </div>
77.
78.         <div class="unit">
79.             <div class="info">
80.                 <h3>Changes feature status</h3>
81.
82.                 <p>Use this for to change the status of features.</p>
83.             </div>
84.             <div class="form_div" id="form_div">
85.                 Not loaded...
86.             </div>

```

```

87.         </div>
88.
89.     </div>
90.
91.     <div class="footer">
92.         <h4>Catapult Hosting<br>All rights reserved</h4>
93.     </div>
94.
95.     <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
96.     <script type="text/javascript">
97.
98.         $(document).ready(function() {
99.
100.             $.getScript("index.js", function(){
101.                 });
102.
103.             });
104.         </script>
105.
106.     </body>
107.
108. </html>

```

index.js – control panel

```

1. function replace(div, string) {
2.     $('#'+ div).html(string);
3. }
4.
5. function number_format(number) {
6.     var format = number.toString().replace(/\B(?=(\d{3})+(?!\d))/g, " ");
7.     return format;
8. }
9.
10. var feature_error = "This feature is disabled.";
11. var api_error = 'Could not reach API.';
12. var error = "There was an error.";
13.
14. $.getScript("../get_features.js", function(){
15.     if(features.error == "false") {
16.
17.         $.getScript("get_sales.js", function(){
18.             });
19.
20.         $.getScript("get_status.js", function(){
21.             });
22.
23.         $.getScript("get_search.js", function(){
24.             });
25.
26.         $.getScript("get_log_in.js", function(){
27.             });
28.
29.         $.getScript("set_feature.js", function(){
30.             });
31.
32.     } else {
33.         alert("Website could not reach feature flag API script. No features will work. Contact your administrator.");
34.     }
35. });

```

get_features.js

```
1. var features;
2.
3. $.ajax({
4.     url: "api/get_features.php",
5.     type : "POST",
6.     contentType : 'application/json',
7.     async: false,
8.     success : function(result) {
9.         features = JSON.parse(result);
10.    },
11.
12.    // show error message to user
13.    error: function(){
14.        replace('alert', api_error);
15.    }
16. });
```

set_features.js

```
1. var id = "form_div";
2.
3. var text = `
4.     <form id="feature_form" onsubmit="return false" action="" method="post"
5.     >
6.         <div class="form-group">
7.             <label for="f_name">Feature: </label>
8.             <select class="input" id="f_name" name="f_name" form="feature_f
9.             orm" required>
10.                 <option value=""></option>
11.             `;
12. var i;
13. for(i = 0; i < features.feature.length; i++) {
14.     text += '<option value="' + features.feature[i].name + '>' + features.feature[
15.     i].name + '</option>';
16. }
17. text += `
18.         </select>
19.     </div>
20.     <div class="form-group">
21.         <label for="f_error">Error: </label>
22.         <select class="input" id="f_error" name="f_error" form="feature
23.         _form" required>
24.             <option value="false">False</option>
25.             <option value="true">True</option>
26.         </select>
27.     </div>
28.     <div class="form-group">
29.         <label for="f_message">Message: </label>
30.         <input class="input" type="text" class="form-
31.         control" name="f_message" id="f_message" required />
32.     </div>
33.     <div class="form-group">
34.         <label for="f_status">Status: </label>
```

```

34.         <select class="input" id="f_status" name="f_status" form="featu
re_form" required>
35.             <option value="true">True</option>
36.             <option value="false">False</option>
37.             <option value="shadow">Shadow</option>
38.         </select>
39.     </div>
40.
41.     <div class="form-group">
42.         <label for="f_comment">Comment: </label>
43.         <input class="input" type="text" class="form-
control" name="f_comment" id="f_comment" required />
44.     </div>
45.
46.     <div class="form-group4">
47.         <button class="input" type="submit" class="btn btn-primary">
48.             <div id="change">Change </div>
49.         </button>
50.     </div>
51.
52.
53.     </form>
54. `;
55.
56. replace(id, text);
57.
58. $(document).on('submit', '#feature_form', function(){
59.
60.     var f_name = document.getElementById("f_name").value;
61.     var f_error = document.getElementById("f_error").value;
62.     var f_message = document.getElementById("f_message").value;
63.     var f_status = document.getElementById("f_status").value;
64.     //console.log(f_status);
65.     var f_comment = document.getElementById("f_comment").value;
66.
67.     var feature_form = { "f_name" : f_name,
68.                          "f_error" : f_error,
69.                          "f_message" : f_message,
70.                          "f_status" : f_status,
71.                          "f_comment" : f_comment
72.     };
73.
74.     var feature_data = JSON.stringify(feature_form);
75.
76.     var xhttp = new XMLHttpRequest();
77.     xhttp.onreadystatechange = function() {
78.         if (this.readyState == 4 && this.status == 200) {
79.             var result = JSON.parse(this.responseText);
80.
81.             if(result.error == "false") {
82.                 alert("Feature changed.");
83.                 location.reload();
84.             } else {
85.                 alert("Error! Not changed.");
86.                 location.reload();
87.             }
88.
89.
90.         }
91.     };
92.     xhttp.open("post", "api/set_feature.php", true);
93.     xhttp.send(feature_data);
94. });

```

get_features.php

```
1. <?php
2.
3. /* Premade output used for testing
4. echo json_encode(array(      "error" => "false",
5.                             "message" => "No error.",
6.                             "sales" => array(
7.                             "error" => "false",
8.                             "message" => "No error.",
9.                             "status" => "true",
10.                            "comment" => "Turned on 05/10/20 by Larry David."
11.                            ),
12.                            "status" => array(
13.                            "error" => "false",
14.                            "message" => "No error.",
15.                            "status" => "true",
16.                            "comment" => "Turned on 02/10/20 by Jerry Seinfeld."
17.                            )
18.                        ));
19. */
20. $features = utf8_encode(file_get_contents('features.json'));
21. echo $features;
```

set_features.php

```
1. <?php
2. $features = utf8_encode(file_get_contents('features.json'));
3. $features = json_decode($features);
4.
5. $data = json_decode(file_get_contents("php://input"));
6.
7. $index = 0;
8.
9. for($i = 0; $i < count($features->feature); $i++) {
10.     if($features->feature[$i]->name == $data->f_name) {
11.         $index = $i;
12.         break;
13.     }
14. }
15.
16. $features->feature[$index]->name = $data->f_name;
17. $features->feature[$index]->error = $data->f_error;
18. $features->feature[$index]->message = $data->f_message;
19. $features->feature[$index]->status = $data->f_status;
20. $features->feature[$index]->comment = $data->f_comment;
21.
22. $fp = fopen('./features.json', 'w');
23. fwrite($fp, json_encode($features));
24. fclose($fp);
25.
26. echo json_encode(array("error" => "false"));
```

features.json

```
1. {"error":"false","message":"No error.","feature":[{"name":"sales","error":"false","message":"No error.", "status":"false","comment":"Search disabled. Did not show coor
```

```
ect results, must be reworked. Turned off 05.10.20"}, {"name": "status", "error": "false", "message": "No error.", "status": "true", "comment": "Turned on 02\10\20 by Jerry Seinfeld."}, {"name": "search", "error": "false", "message": "No error.", "status": "false", "comment": "Search disabled. Did not show correct results, must be reworked. Turned off 05.10.20"}, {"name": "log_in", "error": "false", "message": "No error.", "status": "shadow", "comment": "New feature being tested. Shadow started 07.10.20."}]}
```

get_sales.js

```
1. var xhttp = new XMLHttpRequest();
2. xhttp.onreadystatechange = function() {
3.     if (this.readyState == 4 && this.status == 200) {
4.         var sales = JSON.parse(this.responseText);
5.         var id = 'sales';
6.         var t = new Date();
7.         t.getTime();
8.         console.log(id + " loading - " + t);
9.
10.        var sum = Number(sales.north) + Number(sales.west) + Number(sales.east);
11.        var text = "North: " + number_format(sales.north) + "<br>West: " + number_format(sales.west) + "<br>East: " + number_format(sales.east) + "<br><br>Sum: " + number_format(sum);
12.
13.        if(features.feature[0].status == "true") {
14.            replace(id, text);
15.        } else if(features.feature[0].status == "shadow") {
16.            replace(id, feature_error);
17.        } else {
18.            replace(id, error);
19.        }
20.        var t_2 = new Date();
21.        t_2.getTime();
22.        console.log(id + " was loaded. - " + t_2);
23.
24.    }
25. };
26. xhttp.open("post", "api/get_sales.php", true);
27. xhttp.send();
```

get_status.js

```
1. var xhttp = new XMLHttpRequest();
2. xhttp.onreadystatechange = function() {
3.     if (this.readyState == 4 && this.status == 200) {
4.         var status = JSON.parse(this.responseText);
5.         var id = 'status';
6.         var t = new Date();
7.         t.getTime();
8.         console.log(id + " loading - " + t);
9.
10.        status = JSON.parse(this.responseText);
11.        var text = "";
12.
13.        text += "<div class='item'>";
14.        text += "<div class='server'>Search: </div>";
15.        if(status.feature[2].status == "true") {
16.            text += "<div class='true' title='Server was reached'></div>";
17.        } else {
```

```

18.         text += "<div class='false' title='Server was not reached'></div>";
19.     }
20.     text += "</div>";
21.
22.     text += "<div class='item'>";
23.         text += "<div class='server'>Log in: </div>";
24.         if(status.feature[3].status == "true") {
25.             text += "<div class='true' title='Server was reached'></div>";
26.         } else {
27.             text += "<div class='false' title='Server was not reached'></div>";
28.
29.         }
30.         text += "</div>";
31.
32.     text += "<div class='item'>";
33.         text += "<div class='server'>Service status: </div>";
34.         if(status.feature[1].status == "true") {
35.             text += "<div class='true' title='Server was reached'></div>";
36.         } else {
37.             text += "<div class='false' title='Server was not reached'></div>";
38.
39.         }
40.         text += "</div>";
41.
42.     text += "<div class='item'>";
43.         text += "<div class='server'>Sale numbers API: </div>";
44.         if(status.feature[0].status == "true") {
45.             text += "<div class='true' title='Server was reached'></div>";
46.         } else {
47.             text += "<div class='false' title='Server was not reached'></div>";
48.
49.         }
50.         text += "</div>";
51.
52.     /*
53.     text += "<div class='item'>";
54.         text += "<div class='server'>Webstore: </div>";
55.         if(status.webstore == "true") {
56.             text += "<div class='true' title='Server was reached'></div>";
57.         } else {
58.             text += "<div class='false' title='Server was not reached'></div>";
59.
60.         }
61.         text += "</div>";
62.     */
63.
64.     if(features.feature[1].status == "true") {
65.         replace(id, text);
66.     } else if(features.feature[1].status == "shadow") {
67.         replace(id, feature_error);
68.     } else {
69.         replace(id, error);
70.     }
71.
72.     var t_2 = new Date();
73.     t_2.getTime();
74.     console.log(id + " was loaded. - " + t_2);
75. }
76. };
77. xhttp.open("post", "api/get_status.php", true);
78. xhttp.send();

```


login.js

```
1. var id = 'log_in';
2. var t = new Date();
3. t.getTime();
4. console.log(id + " loading - " + t);
5.
6. var text = "You are logged in as:<br>George Costanza<br>Log out";
7.
8. if(features.feature[3].status == "true") {
9.     replace(id, text);
10. } else if(features.feature[3].status == "shadow") {
11.     replace(id, feature_error);
12. } else {
13.     replace(id, error);
14. }
15.
16. var t_2 = new Date();
17. t_2.getTime();
18. console.log(id + " was loaded. - " + t_2);
```

search.js

```
1. var id = 'search';
2. var t = new Date();
3. t.getTime();
4. console.log(id + " loading - " + t);
5.
6. var text = "Search<br><input type=text>";
7.
8. if(features.feature[2].status == "true") {
9.     replace(id, text);
10. } else if(features.feature[2].status == "shadow") {
11.     replace(id, feature_error);
12. } else {
13.     replace(id, error);
14. }
15.
16. var t_2 = new Date();
17. t_2.getTime();
18. console.log(id + " was loaded. - " + t_2);
```

Assets

main.css

```
1. body {
2.     font-family: 'Roboto', serif;
3.     font-weight: normal;
4.     padding: 0;
5.     margin: auto;
6.     position: relative;
7.     width: 100%;
8.     text-align: center;
9. }
10.
11. .banner {
12.     width: 100%;
13.     height: 25em;
```

```

14.     margin: auto;
15.     padding: 0;
16.     display: inline-block;
17.     vertical-align: text-bottom;
18.     background-image: url("../banner.jpg");
19.     background-position: bottom;
20.     background-size: cover;
21.     box-
        shadow: 0 2px 2px 0 rgba(0,0,0,0.14), 0 1px 5px 0 rgba(0,0,0,0.12), 0 3px 1px -
        2px rgba(0,0,0,0.2);
22. }
23.
24. .footer {
25.     width: 100%;
26.     height: 5em;
27.     margin: auto;
28.     padding: 0;
29.     vertical-align: bottom;
30.     text-align: center;
31.     position: relative;
32.     bottom: 0;
33.     left: 0;
34. }
35.
36. .box {
37.     width: 100%;
38.     margin: auto;
39.     padding: 0;
40.     display: inline-block;
41.     vertical-align: top;
42.     text-align: center;
43. }
44.
45. .unit {
46.     width: 20em;
47.     display: inline-block;
48.     margin: 1.5em;
49.     position: relative;
50.     vertical-align: top;
51.     text-align: center;
52. }
53.
54. .info {
55.     display: inline-block;
56.     vertical-align: top;
57.     text-align: center;
58.     height: 10em;
59. }
60.
61. #sales, #status, #search, #log_in {
62.     box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.2), 0 1px 1px 0 rgba(0, 0, 0, 0.19);
63.     background-color: #f2f2f2;
64.     border-radius: 25px;
65.     padding: 0.5em;
66. }
67.
68. h1,h2,h3,h4,p {
69.     text-align: center;
70. }
71.
72. h1 {
73.     margin-top: 8em;
74.     color: white;
75.     text-shadow: 2px 2px 4px #000000;
76. }
77.

```

```

78. h2 {
79.     color: white;
80.     text-shadow: 2px 2px 4px #000000;
81. }
82.
83. h4 {
84.     font-size: 0.75em;
85.     text-shadow: 0;
86.     font-weight: normal;
87. }
88.
89. .false {
90.     background-color: red;
91.     box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.2), 0 1px 1px 0 rgba(0, 0, 0, 0.19);
92.     width: 0.7em;
93.     height: 0.7em;
94.     border-radius: 50%;
95.     display: inline-block;
96.     margin: 0.25em;
97.     float: right;
98. }
99.
100.     .true {
101.         background-color: lightgreen;
102.         box-
shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.2), 0 1px 1px 0 rgba(0, 0, 0, 0.19);
103.         width: 0.7em;
104.         height: 0.7em;
105.         border-radius: 50%;
106.         display: inline-block;
107.         float: right;
108.         margin: 0.25em;
109.     }
110.
111.     .item {
112.         width: 100%;
113.         display: inline-block;
114.         margin: auto;
115.         padding: 0;
116.     }
117.     .server {
118.         width: 75%;
119.         margin: 0.25em;
120.         padding: 0;
121.         display: inline-block;
122.         text-align: left;
123.         float: left;
124.     }

```

banner.jpg



favicon.ico



Screenshots

Front page



Log in

Log into your company user to view the items on this page.

This feature is disabled.

Sale numbers

These are the sales figures for the last 24 hours. Sales are automatically added after being noted in the system.

This feature is disabled.

Search

Search items on company database

This feature is disabled.

Service status





Here you can view the current status of our services. If there is a problem it will be visible here. You can also read the newest updates on current situations.

Search:	●
Log in:	●
Service status:	●
Sale numbers API:	●

Control panel

Component LLC Feature Flags

Catapult Hosting admin page

<p>Sales API</p> <p>The API transferring sales numbers to the website.</p> <p>Name: <input type="text" value="sales"/></p> <p>Error: <input type="text" value="false"/></p> <p>Message: <input type="text" value="No error."/></p> <p>Status: <input type="text" value="false"/> </p> <p>Comment: <input type="text" value="Search disabled. Did not show correct results, must be reworked. Turned off 05.10.20"/></p>	<p>Status API</p> <p>Status panel showing what services is running for Component LLC.</p> <p>Name: <input type="text" value="status"/></p> <p>Error: <input type="text" value="false"/></p> <p>Message: <input type="text" value="No error."/></p> <p>Status: <input type="text" value="true"/> </p> <p>Comment: <input type="text" value="Turned on 02/10/20 by Jerry Seinfeld."/></p>	<p>Search status</p> <p>The search function on the website.</p> <p>Name: <input type="text" value="search"/></p> <p>Error: <input type="text" value="false"/></p> <p>Message: <input type="text" value="No error."/></p> <p>Status: <input type="text" value="false"/> </p> <p>Comment: <input type="text" value="Search disabled. Did not show correct results, must be reworked. Turned off 05.10.20"/></p>
<p>Log in status</p> <p>The log in function on the website.</p> <p>Name: <input type="text" value="log_in"/></p> <p>Error: <input type="text" value="false"/></p> <p>Message: <input type="text" value="No error."/></p> <p>Status: <input type="text" value="shadow"/> </p> <p>Comment: <input type="text" value="New feature being tested. Shadow started 07.10.20."/></p>	<p>Changes feature status</p> <p>Use this for to change the status of features.</p> <p>Feature: <input type="text" value=""/></p> <p>Error: <input type="text" value="False"/></p> <p>Message: <input type="text" value=""/></p> <p>Status: <input type="text" value="True"/></p> <p>Comment: <input type="text" value=""/></p> <p><input type="button" value="Change"/></p>	

Configuration files

Nginx (sites-enabled)

```
1. server {
2.     listen 80;
3.
4.     root /var/www/featureflags;
5.     index index.html index.php;
6.     server_name 128.39.121.85 www.128.39.121.85;
7.
8.     location / {
9.         try_files $uri $uri/ =404;
10.    }
11.
12.    location ~ /\.php$ {
13.        include snippets/fastcgi-php.conf;
14.        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
15.    }
16.
17.    location ~ /\.ht {
18.        deny all;
19.    }
20. }
```