Introduction

This documentation describes the project, how it works (technically) and includes performance analysis based on audits.

App Description

The app is a simple to-do list, the user can add, edit, mark as completed, mark as active and delete tasks. The app loads with an empty list initially.

A screenshot of a cell phone

Description automatically generated

Once a todo has been added, the appearance is changing and the task is being displayed in a list.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedApp Structure

The structure of the app can be found in the image.

App Architecture

The app is built using MVC architectural pattern.

MVC stands for Model, View and Controller. MVC separates application into three components - Model, View and Controller.

* **Model** represents shape of the data and business logic. It maintains the data of the application. Model objects retrieve and store model state in a database.
* **View** is a user interface. View display data using model to the user and also enables them to modify the data.
* **Controller** handles the user request. Typically, user interact with View, which in-turn raises appropriate URL request, this request will be handled by a controller. The controller renders the appropriate view with the model data as a response.

The following figure illustrates the interaction between Model, View and Controller.

A picture containing game

Description automatically generated

Storage

The app is using local storage API to store the tasks on the list. Therefore, the tasks will not be lost on page refresh.

Clearing the storage from the browser will delete the tasks and the app will load no tasks present.

The storage is being handled by store.js

Rendering

The app is rendering the newly added elements by creating and appending an <li> element within the <ul class=”todo-list”>, currently existing in index.html, when the create event is triggered.

Testing

Automated unit testing is a great way to check that code is working and continues to work as intended. As the app grows in terms of features and complexity, more and more testing is required to assure the successful completion of the projection.

For testing, Jasmine Behaviour Driven Testing was implemented, to ensure that all critical functionalities and all user events are correctly triggered and provide the desired outcome.

All unit testing has been wrriten in ControllerSpec.js .

Performance

1. Performance audit [http://todolistme.net](http://todolistme.net\)

The app performance is at 72% with the following aspects:

* First contentful Paint : **2.1s** (measures the time from navigation to the time when the browser renders the first bit of content from the DOM)
* Speed Index: **3.1s** (shows how quickly the contents of a page are visibly populated)
* Time to Interactive: **7.9s** (measures how long it takes a page to become interactive)
* First Meaningful Paint:**2.3s** (measures the time at which the user feels that the primary content of the page is visible)
* First CPU Idle : **5.6s** (measures when a page is minimally interactive)
* Max Potential First Input Delay : 200ms (measure your user's first impression of your site's interactivity and responsiveness.)

The app can improve by :

* Reduce JavaScript execution time (2.1s)
* Serve static assets with an efficient cache policy (27 resources found)
* Preconnect to required origins (0.3s)
* Minimize main-thread work (4.3s)

A screenshot of a video game

Description automatically generated

1. Comparison

Comparing our todo app with <http://todolist.me> we can see there is a significant difference.

|  |  |  |
| --- | --- | --- |
| Metrics | My App | Competitor |
| Performance | 100 | 74 |
| First contentful paint | 1.2s | 1.9s |
| Speed Index | 1.2s | 2.9s |
| Time to interactive | 1.6s | 7.8s |
| First Meaningful paint | 1.2s | 2.1s |
| First CPU Idle | 1.6s | 5.7s |
| Max potential First Input Delay | 150ms | 220ms |
| Passed audits | 22 | 15 |

A screenshot of a video game

Description automatically generated

1. Optimize Performance

In order to increase the overall performance of an app, it is important to :

* Minify and compress javascript files
* Remove unused CSS rules
* Defer script files
* Serve images in next-gen format (WebP)
* Compress images.