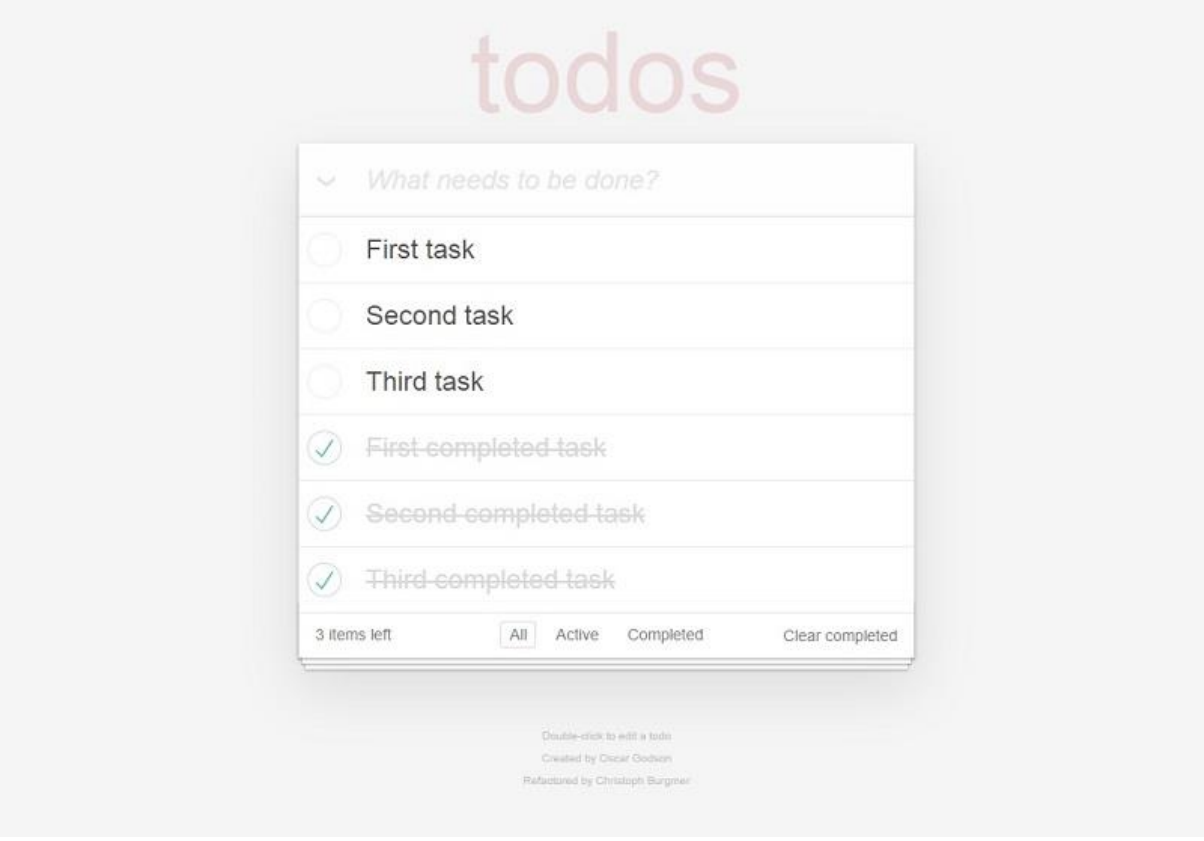
OpenClassrooms Project 8 Technical Documentation and Audit Writeup

1. Introduction

Todo List App is a simple to use application which allows user to:

- add new todos
- modify existing todos
- delete todos
- mark every separate todo as active or completed
- mark all todos as active or completed
- display only all, active or completed todos
- clear all completed todos



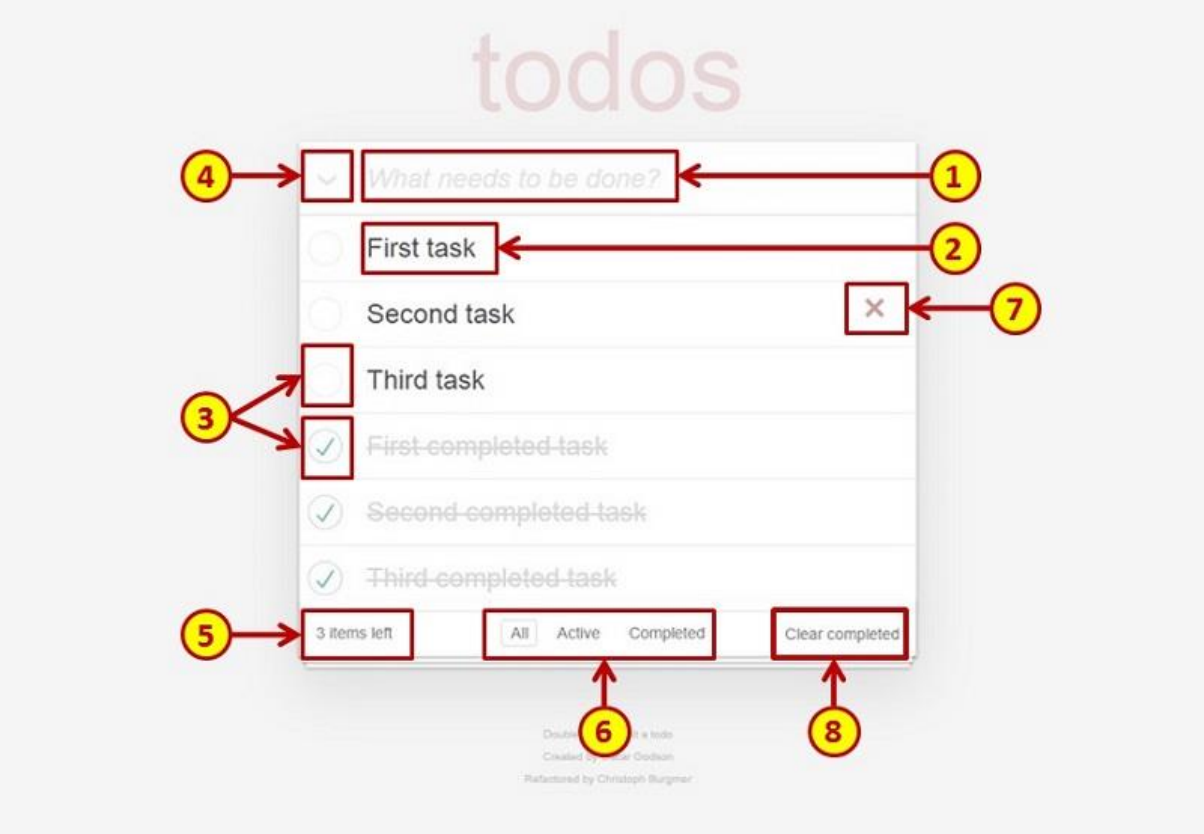
2. How to use

1) Installation and starting application

To install application simply copy application **todo-list-app** main folder on your hard drive.

To start application simply start **index.html** file located in main **todo-list-app** folder.

2) How to use



1 - To add new todo simply click onto field with text ***What needs to be done?*** type name of new todo and press enter.

2 - To change name of existing todo double left click this name and press enter after finish editing.

3 - To change status of active todo to completed or completed todo to active simply click this box.

4 - To change status of all active todos to completed or all completed todos to active (when completed) click this field.

5 - Over here you will see how many todos is still active (not completed).

6 - This filter buttons will let you change view to display all todos or only active or completed todos.

7 - To delete one active or completed todo hover over its name and click this X when appear.

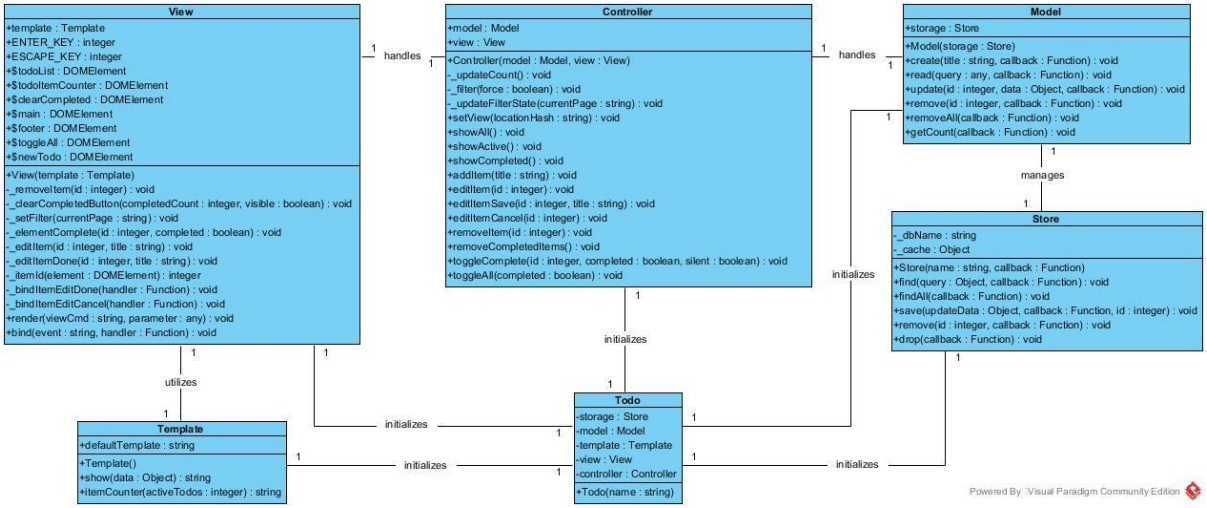
8 - To delete all completed todos click this element.

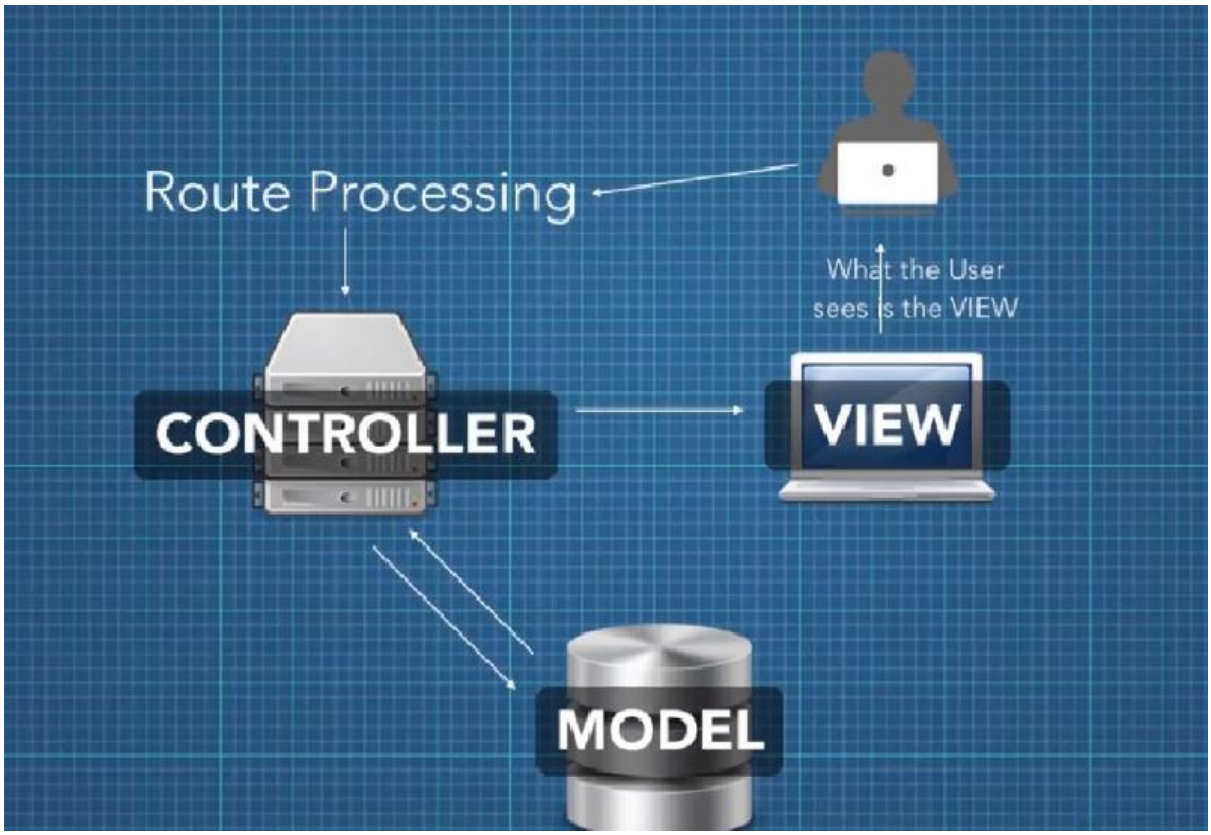
3. How it works (technically)

Our **todo-list-app** is an application using **MVC** architecture.

MVC stands for **MODEL - VIEW - CONTROLLER**.

MODEL, **VIEW** and **CONTROLLER** are three different entities separate from each other.





MODEL and **VIEW** never interacts together. They can do it only through **CONTROLLER**. **MODEL** and **VIEW** can interact only with **CONTROLLER**. **CONTROLLER** is only one entity which can interact with **MODEL** and **VIEW** and it acts like connection between them.

1. ***MODEL***
   Model is responsible for managing the data of the application. It receives user input from the controller. He is responsible for CRUD (create, read, update and delete) operations. Our model is using local storage to save our todos.
2. ***VIEW***
   View is a presentation of the model in a particular format - in our case displaying All, Active or Completed todos (**route**). It also let user to interact with displayed data.
3. ***CONTROLLER***
   Controller responds to the user input from View and performs interactions with Model. It also reading data from Model and passing it to the view.

By using MVC architecture our application works like Single Page Application (SPA) - that means user can interact with todos without reloading webpage.

Files and methods

.HTML FILES

index.html - Our application starting file (in other words - It is the application entry point.)

.CSS FILES

index.css - Defines our application CSS styles

.JS FILES

app.js - Sets up a brand new Todo list.

model.js - Creates a new Model instance and hooks up the storage.

Methods:

- create - Creates a new todo model
- read - Finds and returns a model in storage
- update - Updates a model
- remove - Removes a model from storage
- removeAll - Removs all data from storage
- getCount - Returns a count of all todos

controller.js - Takes a model and view and acts as the controller between them.

Methods:

- setView - Loads and initialises the view.
- showAll - Will get all items and display them in the todo-list.
- showActive - Renders all active tasks.
- showCompleted - Renders all completed tasks.
- addItem - Adding new item to our todos list.
- editItem - Triggers the item editing mode.
- editItemSave - Finishes the item editing mode.
- editItemCancel - Cancels the item editing mode.
- removeItem - Remove item from the DOM and also remove it from storage.
- removeCompletedItems - Will remove all completed items from the DOM and storage.
- toggleComplete - Toggles item between completed and not completed (active).
- toggleAll - Will take all todos and make them complete or incomplete.
- _updateCount - Update number of todos remaining as incomplete (active).
- _filter - Re-filters the todo items, based on the active route.
- _updateFilterState - Simply updates the filter nav's selected states.

helpers.js - It functions are:

- Getting element by CSS selector and attaching event listener to it.
- Attaching a handler to event for all elements that match the selector.
- Finding the element's parent with the given tag name.
- Allowing for looping on nodes by chaining forEach method.

store.js - Creates a new client side storage object.

Methods:

- find - Finds items based on a query given as a JS object.
- findAll - Will retrieve all data from the collection.
- save - Will save the given data to the DB.
- remove - Will remove an item from the Store based on its ID.
- drop - Will drop all storage and start fresh.

template.js - Sets up defaults for all Template methods such as a default template.

Methods:

- show - Creates an <li> HTML string and returns it for placement in our app.
- itemCounter - Displays a counter of how many to dos are left to complete.
- clearCompletedButton - Updates the text within the "Clear completed" button.

view.js - View that abstracts away the browser's DOM completely. It has two simple entry points:

- bind(eventName, handler) - Takes a todo application event and registers the handler.
- render(command, parameterObject) - Renders the given command with the options.

4. Bugs fixing

Four bugs has been found in application code:
1) Bug which not allows adding new todos to the list (simple typo bug).

LOCATION: **controller.js** *line 95*

```
Controller.prototype.adddItem = function (title) {
```

has been changed into

```
Controller.prototype.addItem = function (title) {
```

2) Bug which may leads to potential conflict between duplicate IDs (ID for new todos has been generated randomly which could leads to create duplicated ID's).

LOCATION: **store.js** *starting from line 84*

```
var newId = "";
var charset = "0123456789";
for (var i = 0; i < 6; i++) {
  newId += charset.charAt(Math.floor(Math.random() * charset.length));
}
```

has been changed into

```
var newId = Date.now();
```

3) Console log displayed message when user delete todo (unnessesary code and console log).

LOCATION: **controller.js** *starting from line 165*

```
items.forEach(function(item) {
  if (item.id === id) {
    console.log("Element with ID: " + id + " has been removed.");
  }
);
```

Piece of code above has been removed

4) Missing id in input tag for toggle-all label (label is binded together with an id of HTML element) which prevent toggle all todos to completed works properly.

LOCATION: **index.html** *line 16*

```
<input class="toggle-all" type="checkbox">
```

has been changed into

```
<input id="toggle-all" class="toggle-all" type="checkbox">
```

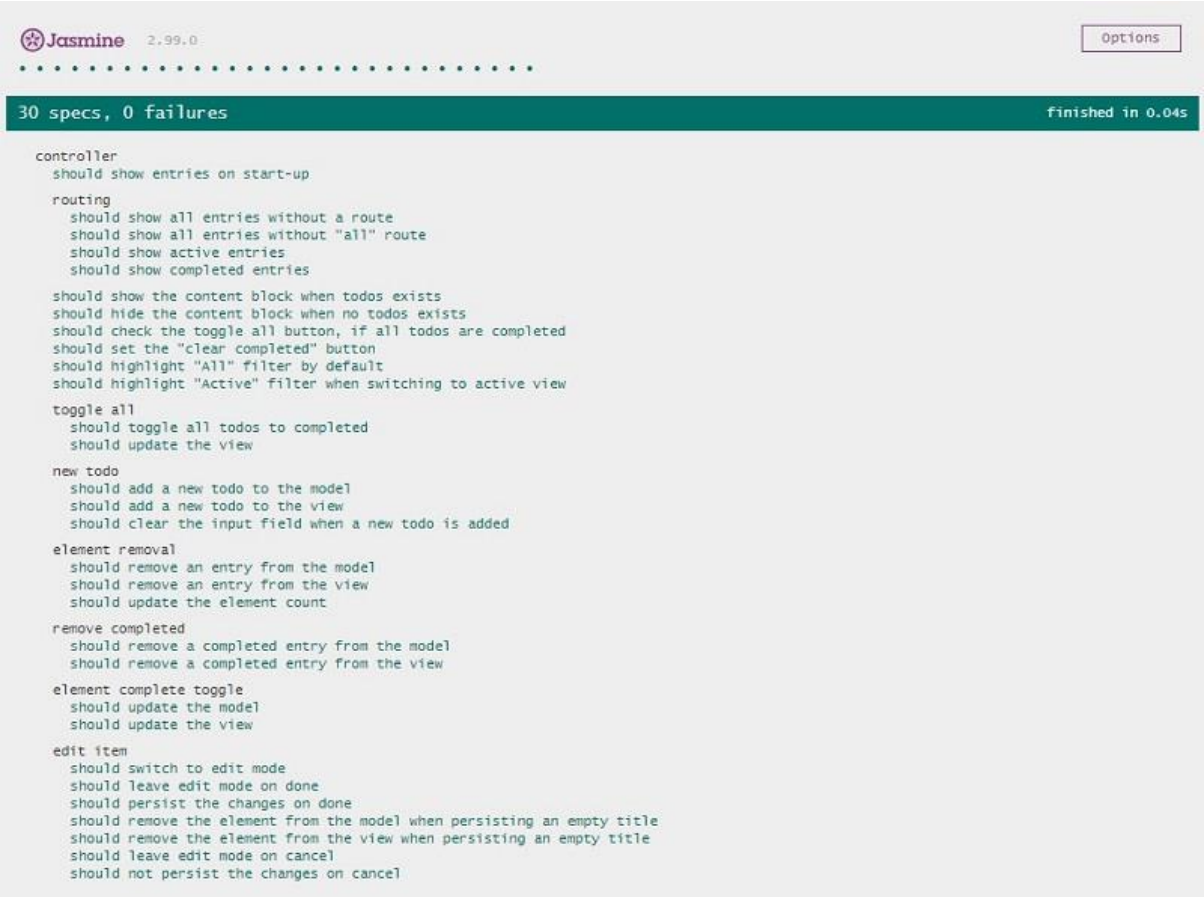
5. Adding tests

Tests has been created by using **Jasmine**.
Jasmine is an open source testing framework for JavaScript.
To start test simply start **SpecRunner.html** file located in application main directory in **test** folder.
To create or modify tests open and modify **ControllerSpec.js** file located in the same directory.

9 tests has been added to the existing tests:

- should show entries on start-up
- should show active entries
- should show completed entries
- should highlight "All" filter by default
- should highlight "Active" filter when switching to active view
- should toggle all todos to completed
- should update the view
- should add a new todo to the model
- should remove an entry from the model

All tests were successful (see image below).



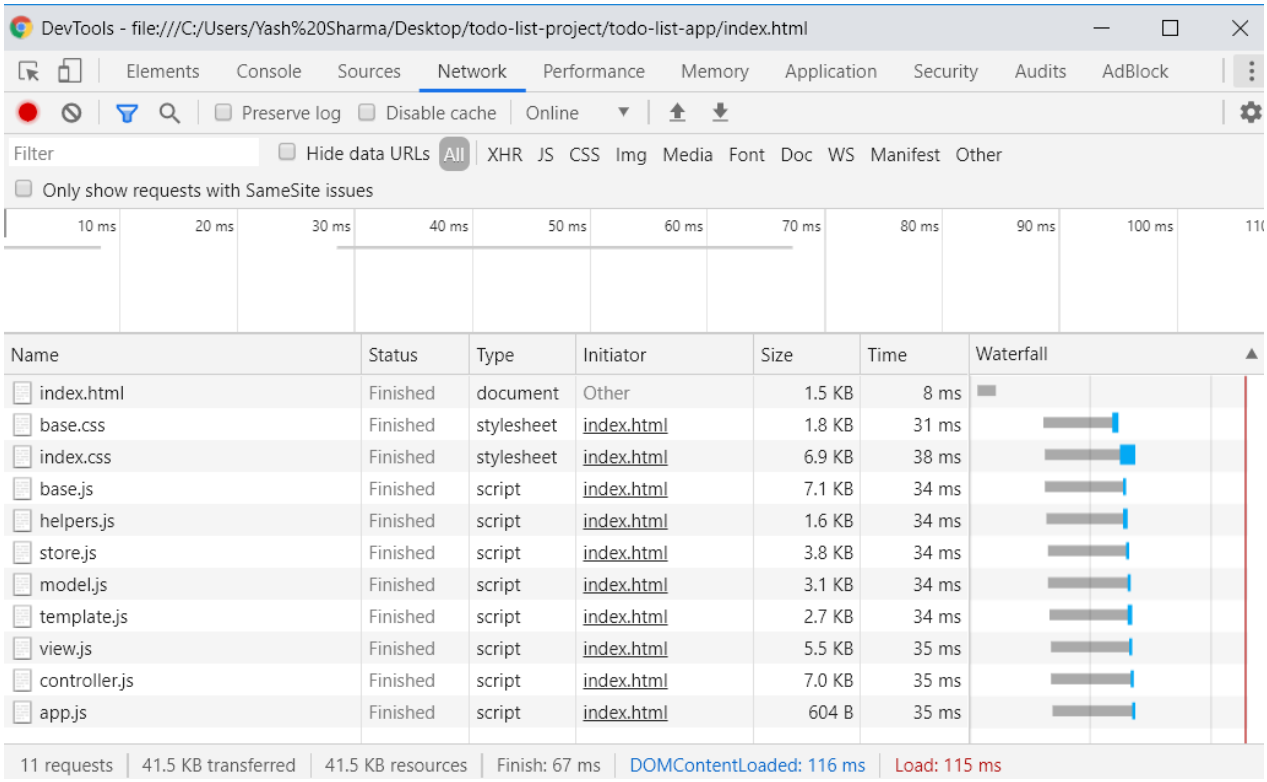
6. Our Todo List App performance audit

Audit was performed by using Chrome browser

Version 80.0.3987.163 (Official Build) (64-bit)
It has been done by using Developer Tools.
System - Windows 10

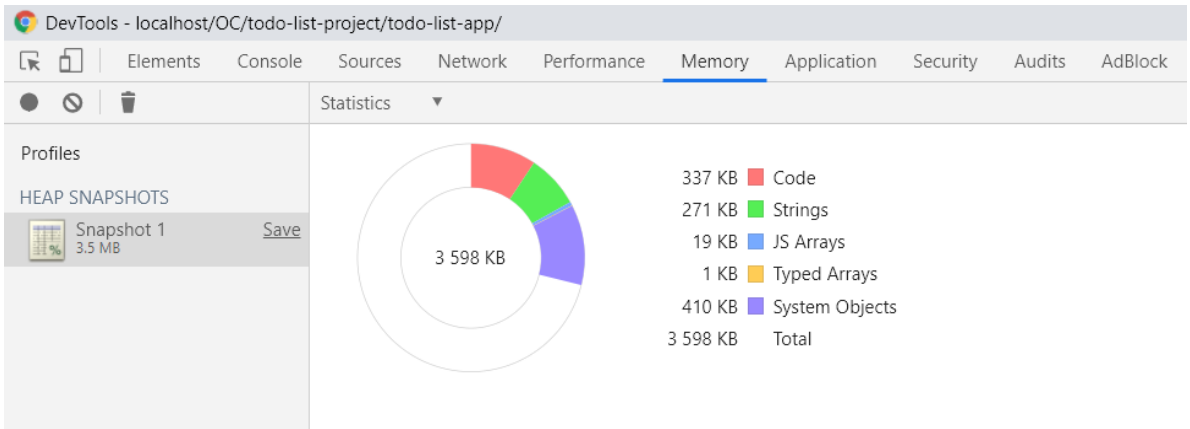
Loading

- PAGE LOADING TIME - **67 ms**
- DOM CONTENT LOADED TIME - **116 ms**
- NUMBER OF REQUESTS SENT - **11**
- AMOUNT OF DATA TRANSFERRED - **41.5 KB**



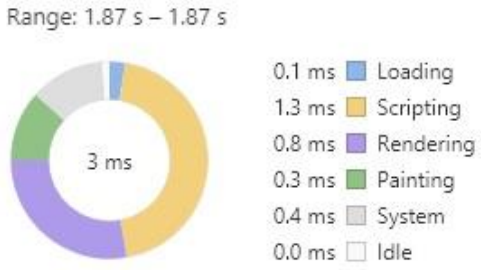
Memory

- TOTAL AMOUNT OF MEMORY USED - **3.5 MB**

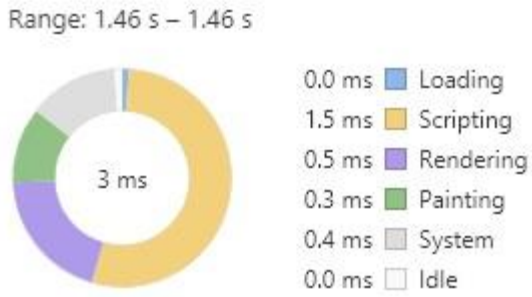


User actions

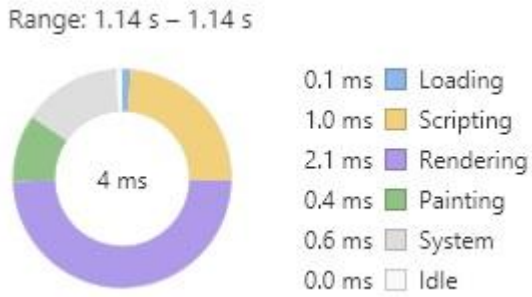
ADDING NEW TASK TO LIST - **3 ms**



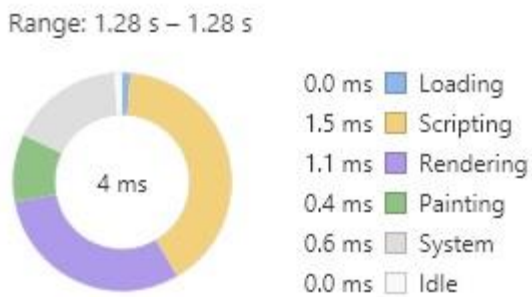
REMOVING TASK FROM LIST - **3 ms**



MARKING ONE TASK AS COMPLETED - **4 ms**

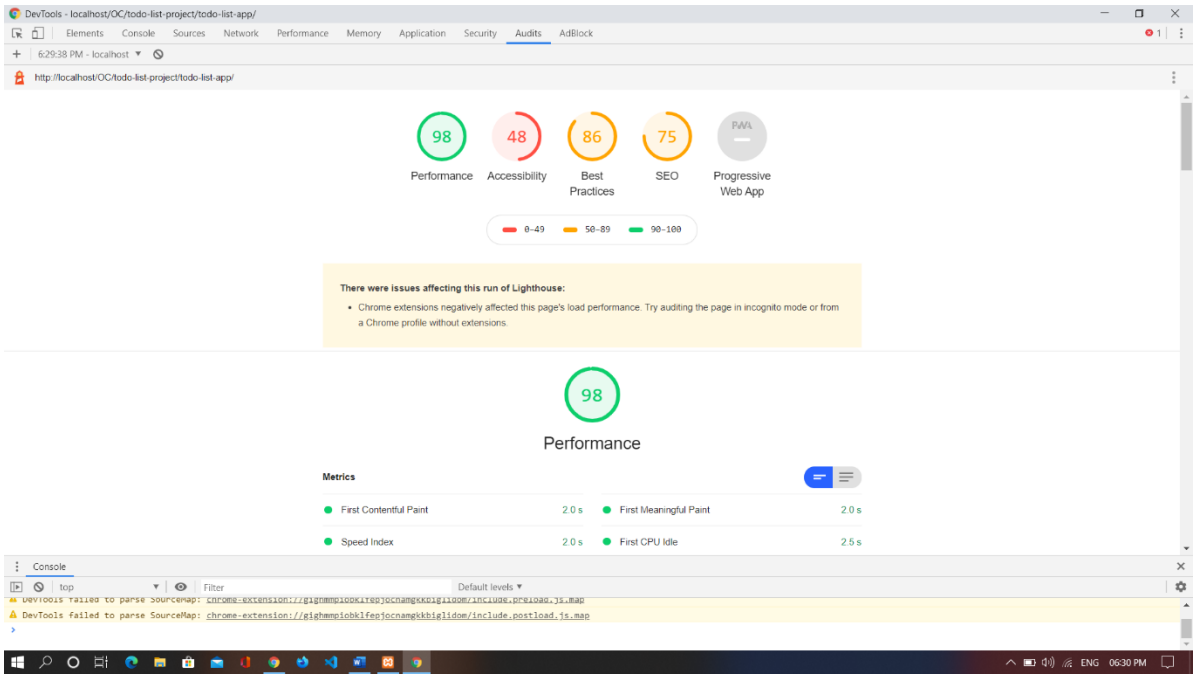


MARKING ALL TASKS AS COMPLETED - **4 ms**



Audit results

- PERFORMANCE – **9**8 of 100
- ACCESSIBILITY - **48** of 100
- BEST PRACTICES - **86** of 100



Improvements that should be made :

- Background and foreground colors do not have a sufficient contrast ratio. Low-contrast text is difficult or impossible for many users to read.
- Form elements do not have associated labels. Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Failing Elements - input.new-todo.
- Does not have a <meta name="viewport"> tag with width or initial-scale. Add a viewport meta tag to optimize your app for mobile screens.
- Document doesn't use legible font sizes. Text is illegible because there's no viewport meta tag optimized for mobile screens. Font sizes less than 12px are too small to be legible and require mobile visitors to “pinch to zoom” in order to read.
- Tap targets are not sized appropriately. Tap targets are too small because there's no viewport meta tag optimized for mobile screens. Interactive elements like buttons and links should be large enough (48x48px) and have enough space around them to be easy enough to tap without overlapping onto other elements.

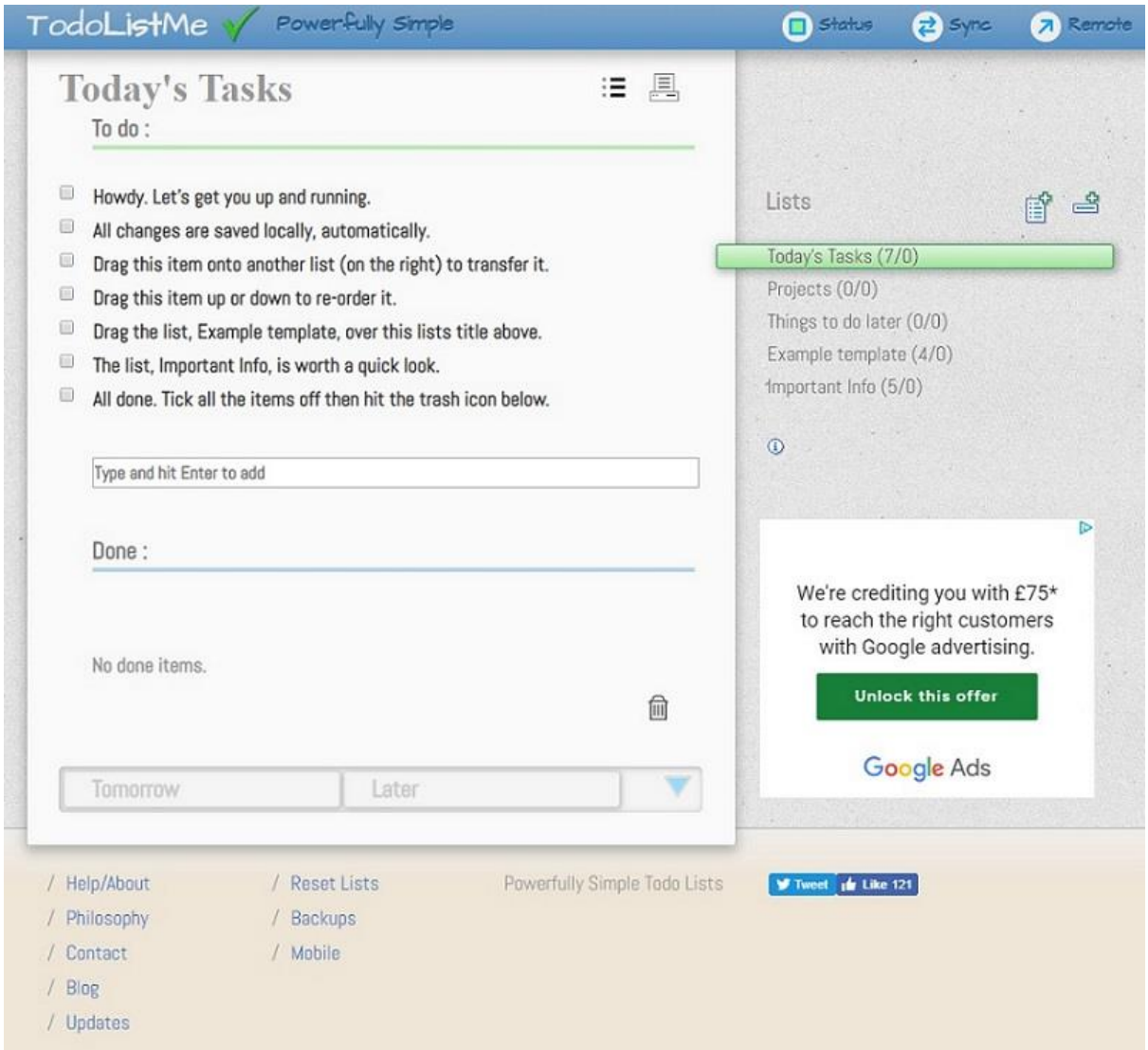
7. Competitors Todo List Me App performance audit

Audit was performed by using Chrome browser

Version 80.0.3987.163 (Official Build) (64-bit)
It has been done by using Developer Tools.
System - Windows 10

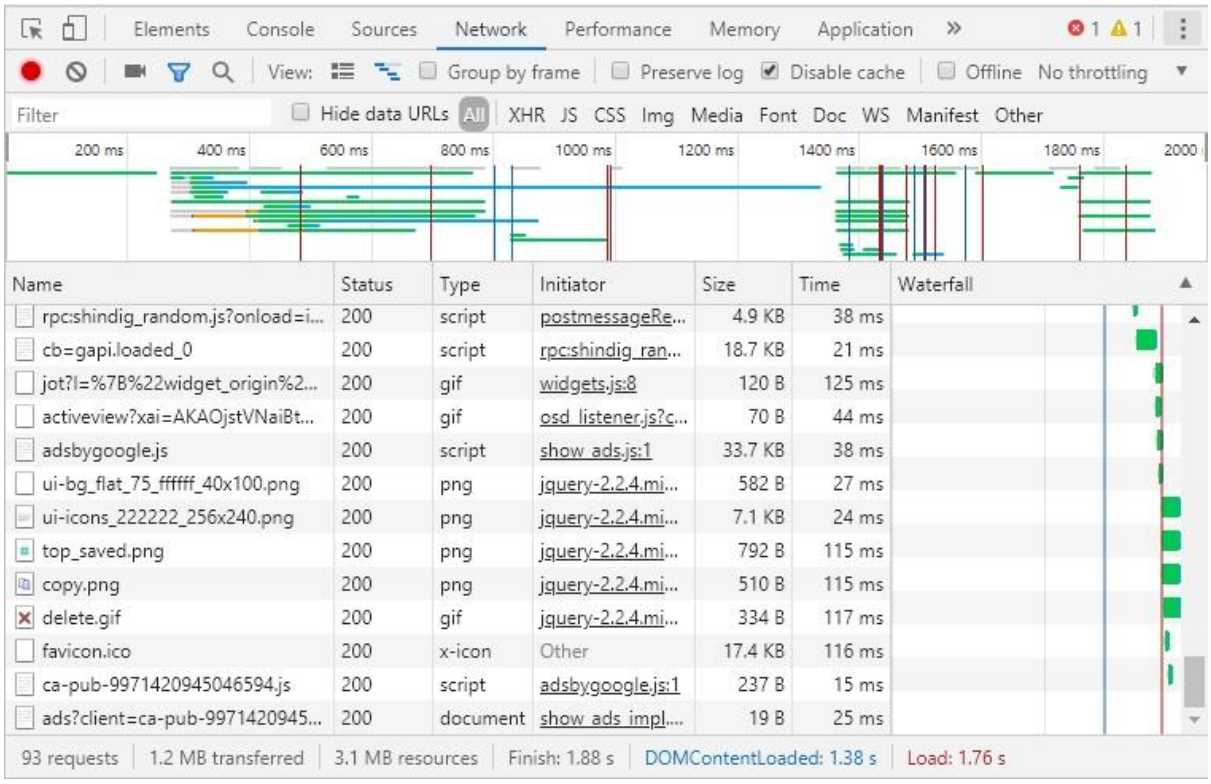
Competitors application is hosted under this link :

http://todolistme.net/



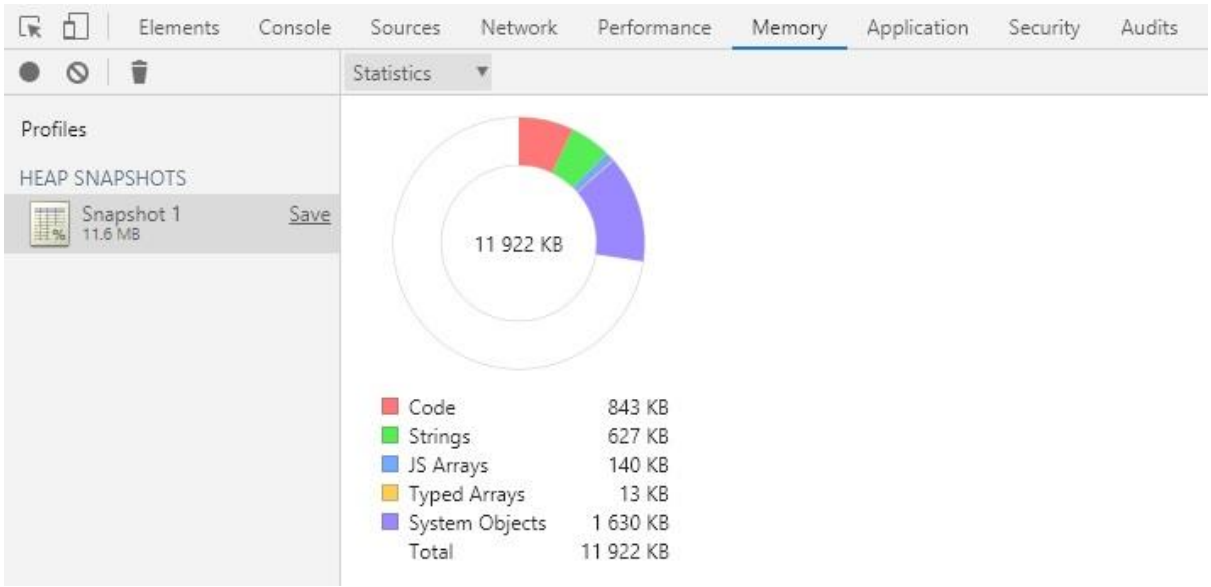
Loading

- PAGE LOADING TIME - **1.76 s**
- DOM CONTENT LOADED TIME - **1.38 s**
- NUMBER OF REQUESTS SENT - **93**
- AMOUNT OF DATA TRANSFERRED - **1.2 MB**



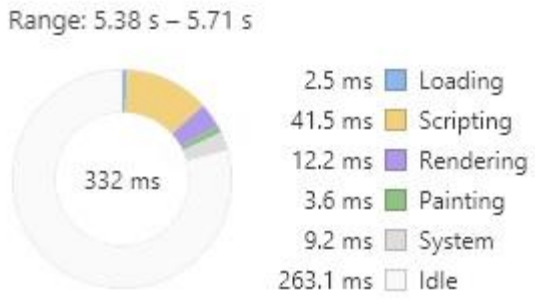
Memory

- TOTAL AMOUNT OF MEMORY USED - **11.6 MB**

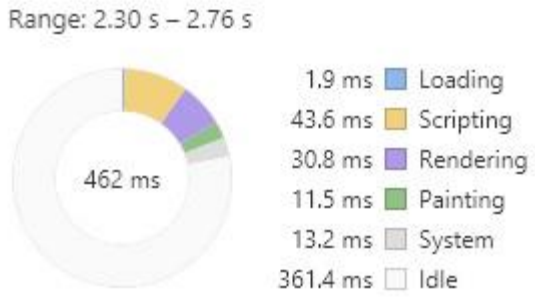


User actions

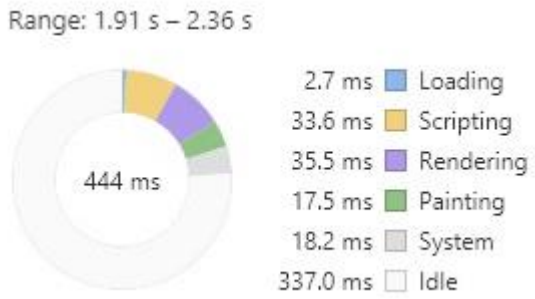
ADDING NEW TASK TO LIST - **332 ms**



REMOVING TASK FROM LIST - **462 ms**



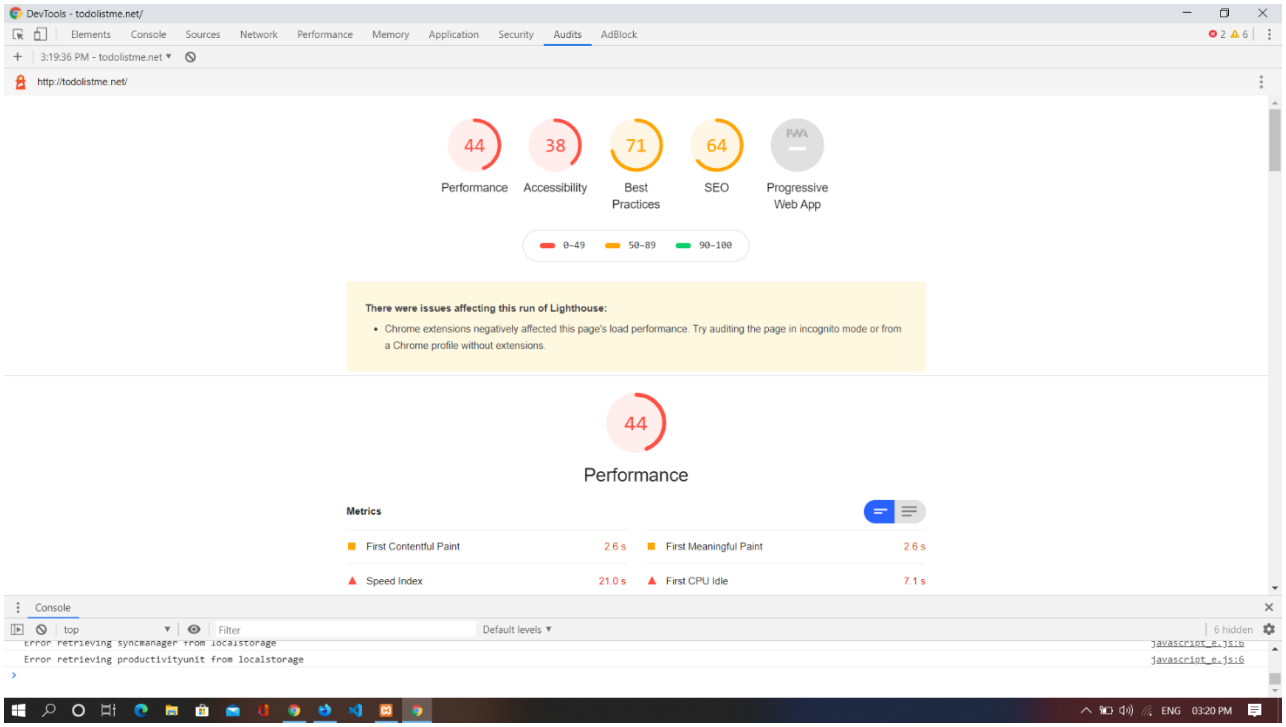
MARKING ONE TASK AS COMPLETED - **444 ms**



MARKING ALL TASKS AS COMPLETED - **FUNCTION NOT AVAILABLE**

Audit results

- PERFORMANCE - **44** of 100
- ACCESSIBILITY - **38** of 100
- BEST PRACTICES - **71** of 100



Improvements that should be made :

- Background and foreground colors do not have a sufficient contrast ratio. Low-contrast text is difficult or impossible for many users to read.
- [id] attributes on the page are not unique. Failing Elements - span#later_number.
- <frame> or <iframe> elements do not have a title. Screen reader users rely on frame titles to describe the contents of frames.
- Image elements do not have [alt] attributes. Informative elements should aim for short, descriptive alternate text.
- Form elements do not have associated labels. Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Failing Elements - input#newtodo.newtodonormal and other 7 inputs.
- <html> element does not have a [lang] attribute. If a page doesn't specify a lang attribute, a screen reader assumes that the page is in the default language that the user chose when setting up the screen reader. If the page isn't actually in the default language, then the screen reader might not announce the page's text correctly.
- Does not use HTTPS. All sites should be protected with HTTPS, even ones that don't handle sensitive data. HTTPS prevents intruders from tampering with or passively listening in on the communications between your app and your users.
- Uses document.write(). For users on slow connections, external scripts dynamically injected via document.write() can delay page load by tens of seconds.
- Includes front-end JavaScript libraries with known security vulnerabilities. Some third-party scripts may contain known security vulnerabilities that are easily identified and exploited by attackers. Library Version - jQuery@2.2.4 / Vulnerability Count - 2.
- Does not have a <meta name="viewport"> tag with width or initial-scale. Add a viewport meta tag to optimize your app for mobile screens.
- Document doesn't use legible font sizes. Font sizes less than 12px are too small to be legible and require mobile visitors to “pinch to zoom” in order to read.
- Tap targets are not sized appropriately. Tap targets are too small because there's no viewport meta tag optimized for mobile screens. Interactive elements like buttons and links should be large enough (48x48px) and have enough space around them to be easy enough to tap without overlapping onto other elements.

- Image texture.png used for background is definitely too big (size). Time to load it takes 562 ms (over half second). It needs to be changed for smaller image and set in css as background-image with repeat to fill up entire page.
- Some of advertisements content is loaded as images. It takes a lot of time.
- Too many inline styles has been found in the code. They shoud be moved into an external css file.

8. Comparison of audits

		OUR APP	COMPETITORS APP
LOADING	PAGE LOADING TIME	58 ms	1.76 s
	DOM CONTENT LOADED TIME	56 ms	1.38 s
	NUMBER OF REQUESTS SENT	12	93
	AMOUNT OF DATA TRANSFERRED	41.2 KB	1.2 MB
	TOTAL AMOUNT OF MEMORY USED	3.7 MB	11.6 MB
USER ACTIONS	ADDING NEW TASK TO LIST	3 ms	332 ms
	REMOVING TASK FROM LIST	3 ms	462 ms
	MARKING ONE TASK AS COMPLETED	4 ms	444 ms
	MARKING ALL TASKS AS COMPLETED	4 ms	FUNCTION NOT AVAILABLE

Our Todo List App

Advantages (+)

- Short loading time
- Low memory usage
- Short time of performing functionalities
- Clean and readable code
- Perfect commented code
- Simple design

Disadvantages (-)

- Limited functionality
- Only local storage is used to save data (Data is lost after cookies are being cleaned).

Competitors Todo List Me App

Advantages (+)

- Ability to sort tasks (Normal, Alphabetical, Random and Top 3)
- Ability to print lists
- Ability to drag and drop tasks
- User can create multiple lists
- User can create categories
- Data can be saved on a remote

Disadvantages (-)

- Long loading time
- High memory usage

- Long time of performing functionalities
- Google ads significantly extend loading time
- Code is not well organized