# Task Management Application design document

## Design Specification

**Version 1.0**  
**26 July, 2020**

Table of Contents

[Task Management Application design document 1](#_Toc46762276)

[Design Specification 1](#_Toc46762277)

[Overview 2](#_Toc46762278)

[Business Rules 2](#_Toc46762279)

[Enhancements incorporated 2](#_Toc46762280)

[Save or restore 2](#_Toc46762281)

[Service worker for PWA 2](#_Toc46762282)

[Validation for title length 2](#_Toc46762283)

[Validation for no of lists that can be added 3](#_Toc46762284)

[Libraries Used 3](#_Toc46762285)

[Services and Data Factory 3](#_Toc46762286)

[Caching 3](#_Toc46762287)

[Components 4](#_Toc46762288)

[Component Design and Interface 4](#_Toc46762289)

[Utilities 4](#_Toc46762290)

[Flow Diagram 5](#_Toc46762291)

[1. Overview 5](#_Toc46762292)

[2. Change theme 6](#_Toc46762293)

[3. Add task list 7](#_Toc46762294)

[4. Edit task list 8](#_Toc46762295)

[5. Delete task list 9](#_Toc46762296)

[6. Add task 10](#_Toc46762297)

[7. Edit task 11](#_Toc46762298)

[8. Delete task 12](#_Toc46762299)

[9. Save to storage 13](#_Toc46762300)

[10. Reorder the task list 14](#_Toc46762301)

[11. Reorder the tasks in the lists 14](#_Toc46762302)

[Unit Test Coverage 15](#_Toc46762303)

[Theming 15](#_Toc46762304)

[Steps to Run the application 17](#_Toc46762305)

# Overview

This document will summarize the high-level design for the solution to task management application requirements shared by Tavisca.

# Business Rules



## Enhancements incorporated

### Save or restore

* 1. The created lists can be synced to the session storage using the ‘Store the lists’ button.
  2. The logic for this has been incorporated in DataService. This can be extended to interact with server using REST etc leaving the interface same.

### Service worker for PWA

1. Service workers for angular have been added to the application for enabling the offline mode in application. The last state cached during the network availability will be retained even when user is offline.
2. When the network in available again the new data will be cached and the application state will be refreshed.

### Validation for title length

1. Validation has been added to limit the length of the title for tasks and lists
2. On exceeding the limit an informational message is shown to the user

### Validation for no of lists that can be added

1. Validation has been added to limit the no of lists that can be added
2. On exceeding the limit an informational message is shown to the user

# Libraries Used

|  |  |
| --- | --- |
| **Name** | **Version** |
| **@angular/material** | 8.2.3 |
| **@angular/pwa** | 0.1000.4 |
| **@angular/service-worker** | 8.2.14 |
| **Angular** | 8.2.14 |
| **Typescript** | 3.5.3 |

# Services and Data Factory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Service Name | Method Name | Input Parameters | Output Parameters | Description |
| DataService | getTaskList | - | Array<TaskListItem> | Fetches the task list from storage and returns as an array of items |
| DataService | setTaskList | taskList: Array<TaskListItem> | - | Saves the task list into the storage |
| AppService | initializeTaskList | - | - | Initializes the task list and stores in the property for communication between components |
| AppService | saveLists | - | - | Saves the task list current state to storage |

# Caching

For storage purpose currently using the session storage of window. The interface for this is in DataService which can be modified accordingly

# Components

## Component Design and Interface

Communication between the components is done via the Input and Output interfaces. Following table lists the design and respective interfaces for the components used-

|  |  |  |
| --- | --- | --- |
| No. | Component | Description |
| 1 | **AppComponent** **Selector**: app-root **Input:** NA **Output**: NA | The bootstrap component of the application |
| 2 | **TaskListItemComponent** **Selector**: app-task-list-item **Input:** taskListItem: TaskListItem, taskList: Array<TaskListItem>, literals  **Output**: NA | The individual task list item component. It handles the edit and delete of the task list item as well as the manipulation of the tasks in the task list item |
| 3 | **ConfirmationDialogComponent** **Selector**: app-confirmation-dialog **Input:** NA **Output**: NA | The dialog component that can be used for confirmation purpose. Currently used for delete confirmation in the application |
| 4 | **EditItemDialogComponent** **Selector**: app-edit-item-dialog **Input:** NA **Output**: NA | The dialog component used for editing of the task list as well task titles |

# Utilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Utility Name | Method Name | Input Parameters | Output Parameters | Description |
| AppUtil | validateTaskList | title, context | Validation status | Validates the task list item in the specified context and returns the validation status |
|  | validateTask | title, context | Validation status | Validates the task in the specified context and returns the validation status |
|  | validateDuplicateTaskList | title, context | Validation status | Validates the task list item in the specified context for duplicity |
|  | validateTitleMaxLength | title | Validation status | Validates the specified title for the character limit |
|  | reorderArrayItem | arr, index, insertAt | - | Inserts the item at the index position in the array to the insertAt position and reorders the list accordingly |
|  | validateDuplicateTask | title, context | Validation status | Validates the task in the specified context for duplicity |

# Flow Diagram

## Overview

A close up of a logo

Description automatically generated

## Change theme

A screenshot of a cell phone

Description automatically generated

## Add task list

A screenshot of a cell phone

Description automatically generated

## Edit task list

A screenshot of a cell phone

Description automatically generated

## Delete task list

A screenshot of a cell phone

Description automatically generated

## Add task

A screenshot of a cell phone

Description automatically generated

## Edit task

A screenshot of a cell phone

Description automatically generated

## Delete task

A screenshot of a cell phone

Description automatically generated

## Save to storage

A screenshot of a cell phone

Description automatically generated

## Reorder the task list

A screenshot of a cell phone

Description automatically generated

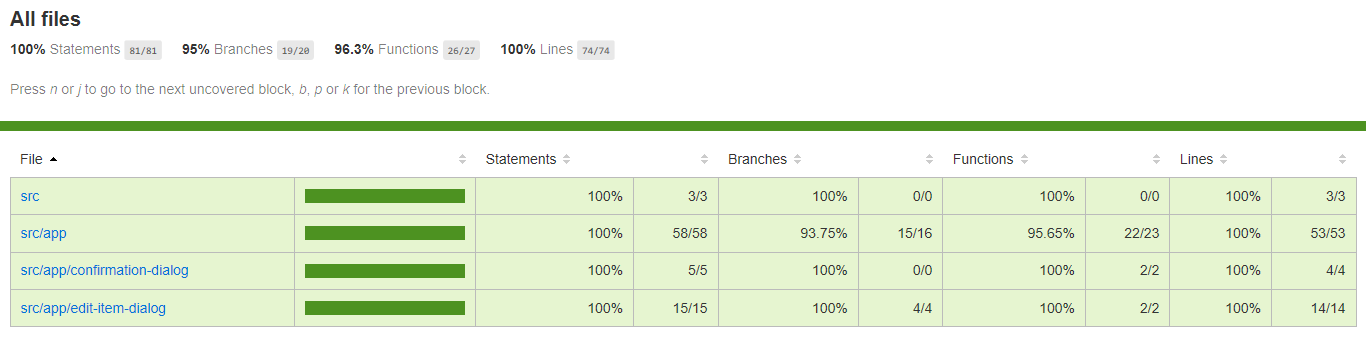
## Reorder the tasks in the lists

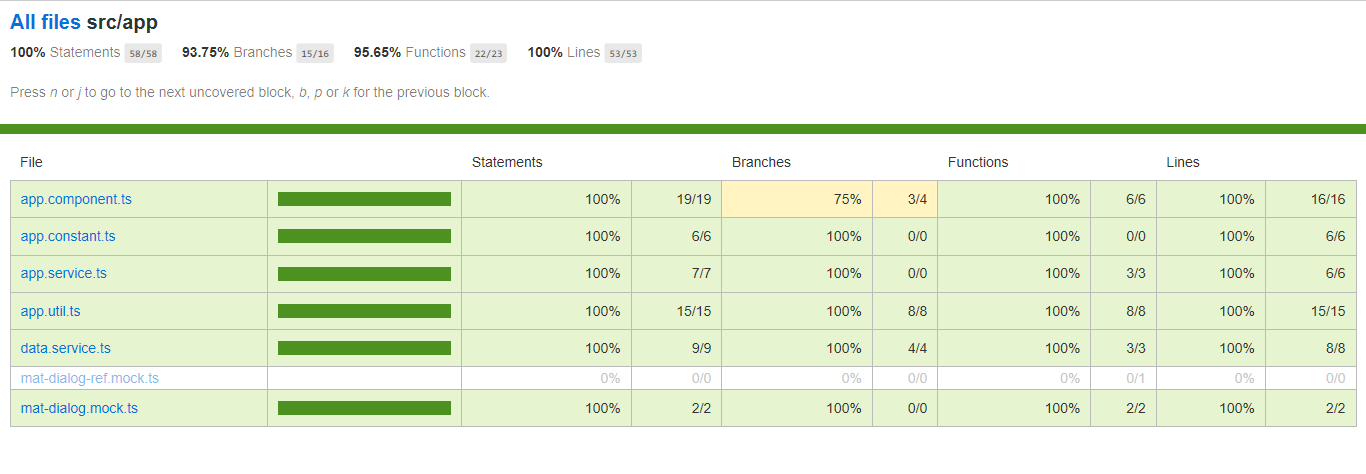
A screenshot of a cell phone

Description automatically generated

## Unit Test Coverage

Jasmine test cases have been covered for the components and services in the application. For details go to the coverage directory in codespace and open the index.html file.

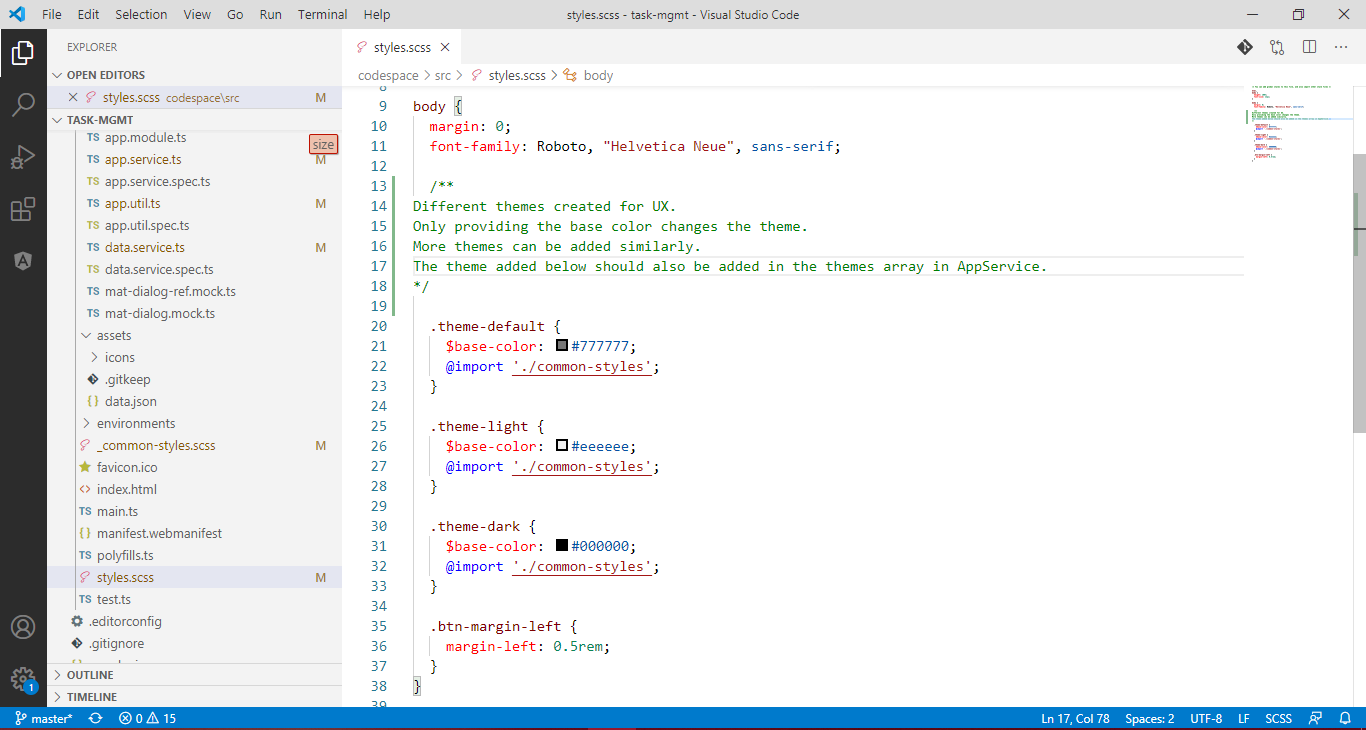




## Theming

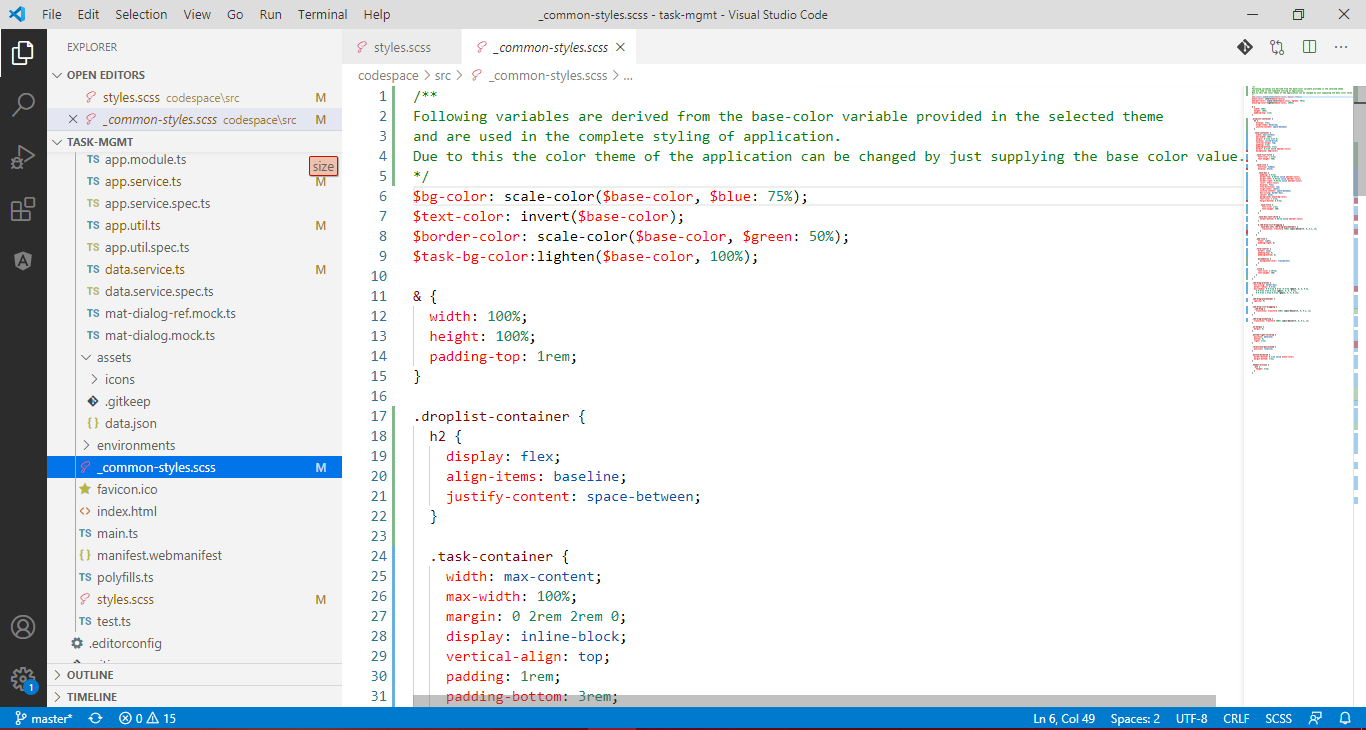
The theme of the application can be toggle via the provided dropdown for the same.

The themes are differentiated based on the root selector for the theme in the styles.scss which is set as the class for the application container.

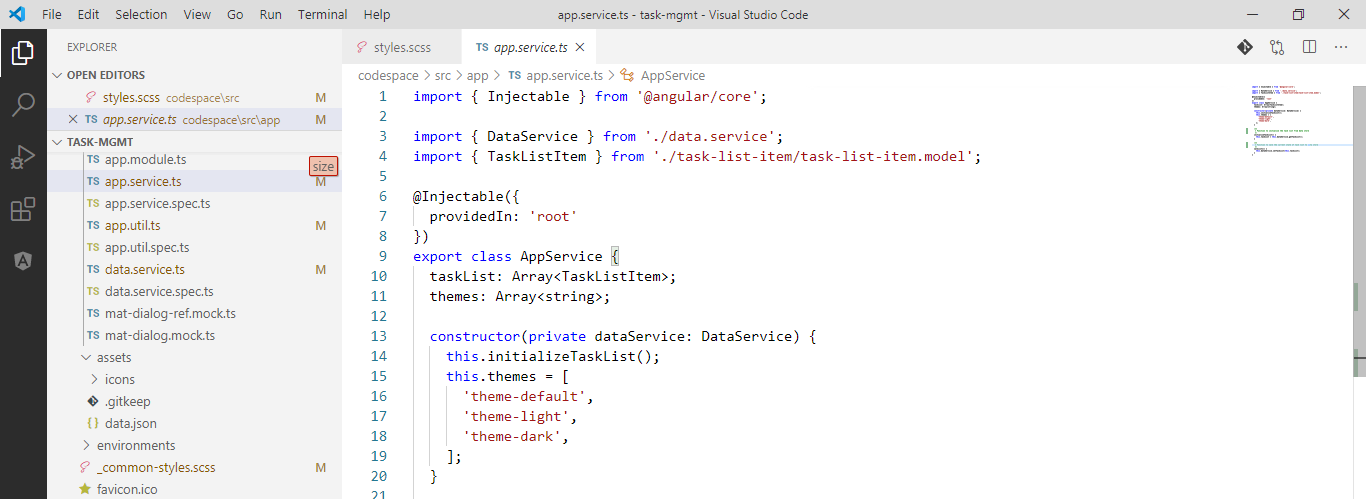


For adding a new theme follow the steps below-

1. Add a class to the styles.scss for e.g. theme-dark
2. Declare a base-color for the theme
3. Import the common-styles partial

The common styles consist of the variables derived from the base-color variable and used in the application for styling it.

1. Add the created theme class into themes array in AppService



## Steps to Run the application

Follow the steps listed below to run the application:

1. Download the code
2. Move to the following directory-

task-mgmt\codespace

1. Open a command prompt in the path mentioned in step#2 and run the following commands
   1. npm install // This will install the node-modules
   2. ng build --prod // This will compile the application
   3. ng serve -o // This will run the application and open up a browser tab loading the application
   4. ng test // This will run the test cases and refresh the coverage directory