

Task Manager by Lazar Stanisavljevic

A simple, yet effective, task management solution

<b>Summary</b>	<b>2</b>
<b>Development</b>	<b>2</b>
<b>Form</b>	<b>7</b>
Html	7
Javascript	7
CSS	11
<b>Function</b>	<b>12</b>

## Summary

The website is designed to allow a user to organise and sort out their upcoming tasks for a shorter period of time. While there is no set timespan available, the current format would more likely be applicable for a user's day to day scheduling and organisation. Provided alongside a simple weather app that displays the current and upcoming weather in a country, one can use this information to plan out activities that might just rely on weather, while prioritising the ones that don't, all by placing out the tasks in the provided planner!

## Development

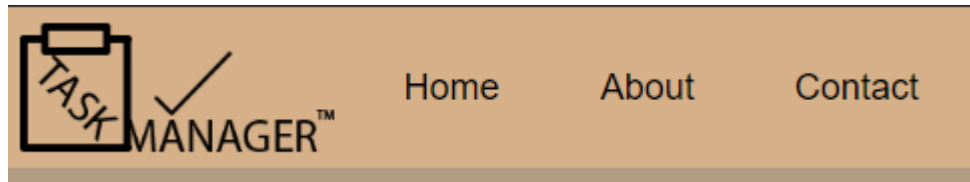
To initialize the planning stages of the project, I began to write down as many of the HTML tags, Javascript functions and CSS formats I could remember, while using my past assignments as assistance. Looking back through my tasks, I managed to collect a very comprehensive list of what I could use in my project and then proceeded to brainstorm ideas for what I could build using all this utility.

Getting stuck at this stage would have been an understatement, however, I took a step back and went back to the core of HTML to design a responsive layout, much like we did for the initial tasks in chapter 1.



*Initial layout for landing page*

After playing around with the layout, I ended up deciding I would like to keep the header and footer in their respective places, putting availability for the nav bar(s) to be placed in both of these, but initially beginning by placing the nav bar in the header alongside a logo I created for an initial idea I had for a type of daily task-manager website.



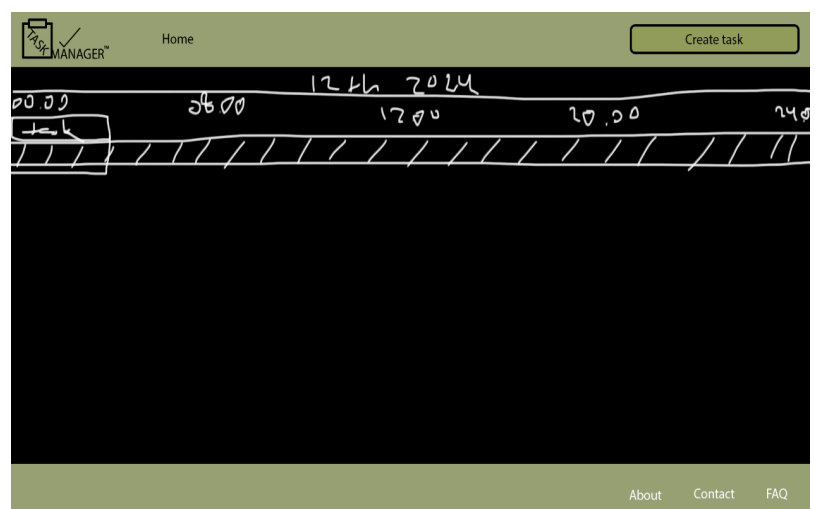
*Header design with navbar using ul with display:flex;*

Besides the header and footer, I figured a 75/25 split for the main content on the page would suffice for my task manager concept; 75% of the page dedicated to the planner and 25% of the page dedicated to some other content I had yet to decide (was thinking about a blog-post timeline). At the top, right below the header, I decided to include an interval for the user to be shown their date and time. I always enjoy having these QoL functions on websites so I decided to give some space for this on my page 😊

The landing page now had its initial design, at least for the Desktop sized view. (I had already begun planning the smaller screen layouts, hence why I decided to opt for a grid layout for the main content as I find it simple to reorganize later).

Furthermore, the design for how the task manager should work started coming along. At first the plan was to build a 24 hour task planner, where the user was able to specify a time slot in their daily schedule for when the task was supposed to be performed, almost like a calendar, however, this idea

started having cracks at the early stages in development, so I settled with a simple, yet sleek design to allow the user to just rearrange the tasks in an order or priority. No time schedule, just priority.

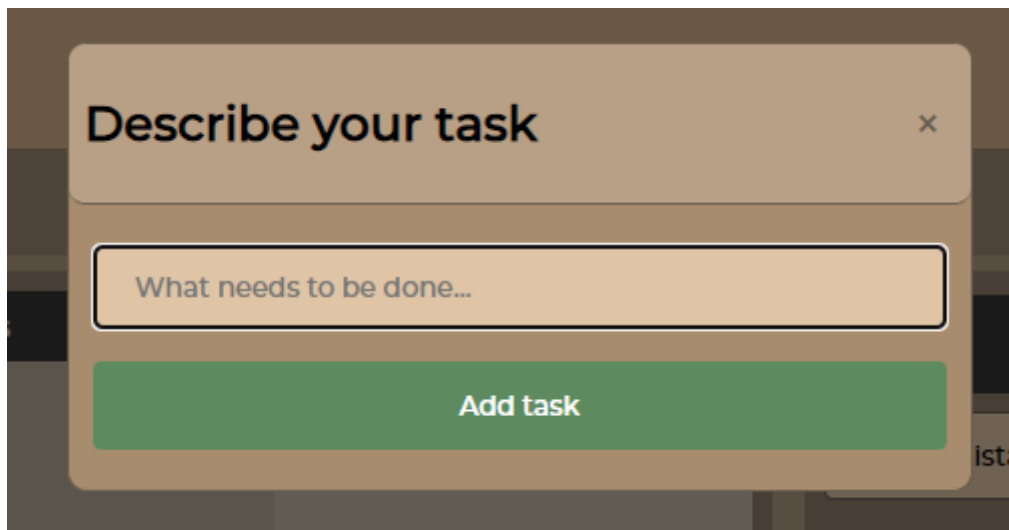


*Sketch in photoshop for the first iteration of design for my Task Manager*



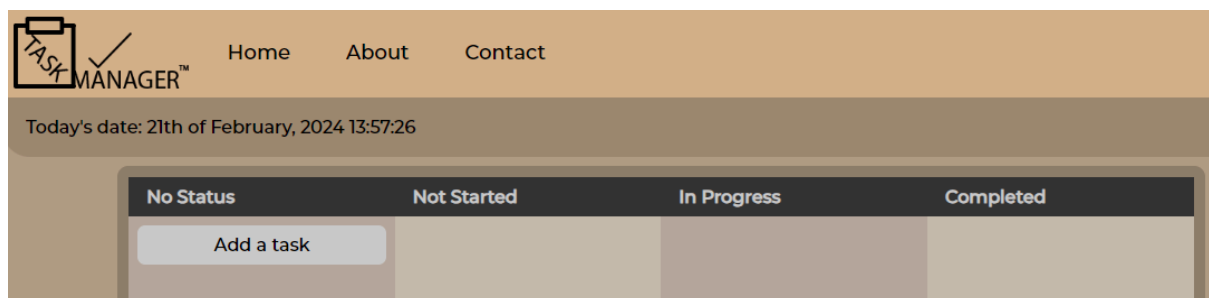
*Secondary design for my Task Manager. A design with focus on simplicity and accessibility*

I decided to learn how to code a modal to make the appearance look nice for when you press the “Add a task” button. Doing so I would be able to have an overlay on the background while the user is requested to fill out task details, and it would feel more like a proper input request. Sort of like a prompt, but not as rudimentary.



*Modal design*

Alongside the design process of the modal, I subsequently added a grid-based display to my body with all the elements, (header, footer, main, section and aside), placed out appropriately in the desired sections. Having decided I wanted my header and footer elements to stretch out and cover 100% of the width of the available screen on the device, but for my section, body and aside to be positioned within a grid-based layout, I opted to place the header and footer independently outside the grid at their appropriate locations, and eventually decided this would work the best for the section as well, as I wanted to format the main separate from the rest.

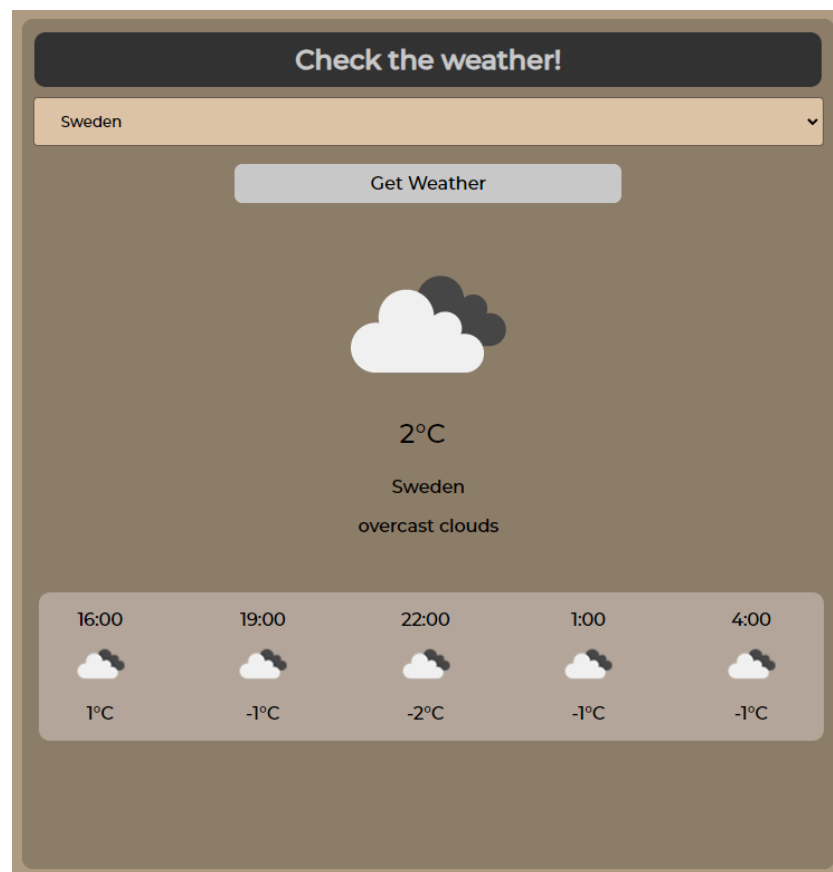


*Header, section and body design after some reorganising with layouts*

As you can tell at this stage I decided to format my text with a font, using one of my favoured fonts, “monsterrat”, as well as adding some border rounding styling to a lot of the elements to remove the harsh nature of 90° angles. These two changes really brought out the simplicity and readability I was hoping for this page.

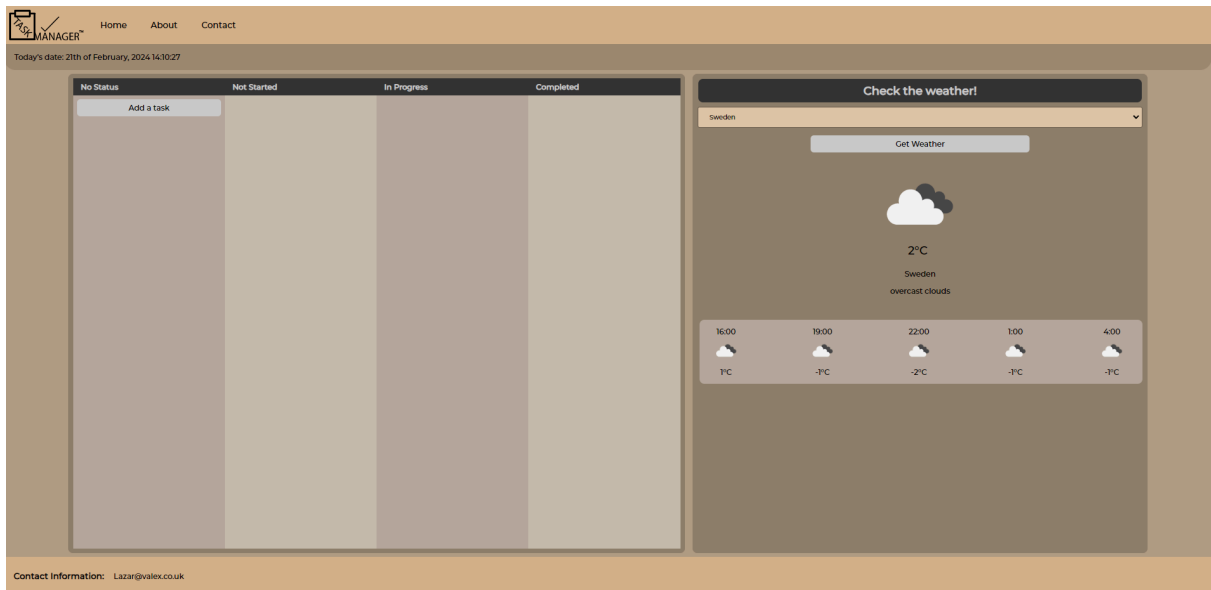
Having finished this, I finally decided what I wanted to have in the remaining 25% of the, now formatted, main element. I chose to develop a simple weather application that would sit alongside the task planner to allow the user to utilize the current and upcoming weather of different countries to plan their tasks.

I did this by utilizing an api from Openweathermap that would allow me to fetch the current weather and a weather forecast, (including all the descriptions, temperatures and icons/images), to display on my website.



*Design of the weather-by-country app*

Now that all the elements were in place, I started formatting the layout for a phone/tablet experience alongside a desktop experience, that once finalized I was pretty happy with the results.



Desktop layout



Phone layout (iPhone SE)

# Form

## Html

The HTML structure of the website is very simple. There are two pages, index.html and about.html. The index.html acts as the landing page and also the home page that hosts the main usecase of the website, namely the task manager and the weather app. The about.html page is just a separate page to host some text explaining what the webpage is about and why it was made.

## Javascript

I decided to structure the Javascript into two main section. A section for functions and variables to be initiated upon the requirement location pathname = /index.html, and a section for functions and variables to be initiated regardless of the location pathname. In this situation, the location pathname refers the name of the html document that is currently active / open. My reasoning behind this decision was to prevent any errors from occurring when opening the about.html page as when the javascript executed on this page, it came with errors that were set up for the index.html page. These errors occurred as catch events were triggered on load as they could not reference the information required from index.html. These errors originated from the following fetch web API events:

```
fetch('https://restcountries.com/v3.1/all')
  .then(response => response.json())
  .then(data => {
    const countrySelect = document.getElementById("country");
    const countryNames = data.map(country => country.name.common).sort();
    // Extracts common names and sort alphabetically

    countryNames.forEach(commonName => {
      const option = document.createElement("option");
      option.value = commonName;
      option.textContent = commonName;
      countrySelect.appendChild(option);
    });
    // Adds options to the select dropdown element
  })
  .catch(error => {
    console.error('Error fetching countries data: ', error);
    alert('Error fetching countries data. Please try again.');
```

```

fetch(currentWeatherUrl)
  .then(response => response.json())
  .then(data => {
    displayWeather(data);
    // Calls displayWeather function with data from URL query
  })
  .catch(error => {
    console.error("Error fetching current weather data: ", error);
    alert("Could not fetch weather data for this country. Sorry!");
  });
// Fetch event that gets current weather from a json and calls displayWeather()

fetch(forecastUrl)
  .then(response => response.json())
  .then(data => {
    displayHourlyForecast(data.list);
    // Calls displayHourlyForecast function with data list from URL query
  })
  .catch(error => {
    console.error("Error fetching hourly forecast data: ", error);
  });
// Fetch event that gets upcoming forecast from a json and calls displayHourlyForecast()

```

In pseudocode syntax, both of these code snippets would be simplified to:

Fetch data from 'website\_with\_data' using API:

If <successful> then

Parse the response as JSON

If <JSON parsing is successful> then

Retrieve relevant element from HTML document and append data or call function using data

If <JSON parsing fails> then

Display an error message indicating failure to parse JSON data

If <error occurs during fetching> then

Log the error to the console

Display an alert message indicating failure to fetch data

This is the essence of these code snippets where the red colour coded if-then statement would be instantly called upon the loading of about.html. This is why I wrapped all these functions only relevant to index.html in the following if-then statement:

```

// vv INDEX.HTML vv //

if (location.pathname === "/index.html") {
  // Makes sure that the current page open is the index
  // Used to load all the below code that is relevant to said page

```

If <current\_page is index.html> then ...



The date function that is used to display the date right under the header is at the top of the javascript document, as this is relevant to both the index.html and the about.html pages. In pseudocode the format would look like this:

Initialize variables and Date object

Format and Append text

Setup Interval <updateTime>

Initialize functions <updateTime, getEnding, checkMinutes, checkSeconds>

### Functions

#### updateTime

Calls for new Date object, updates variables with new data, formats and appends text

#### getEnding

Uses Date object to update variable that gets added after date text (examples are “th” in 25th or “rd” in 23rd)

#### checkMinutes

Uses Date object to update text used to display minutes. Adds a 0 in front of text if it's less than 10. (examples are the 0 added if time displayed is 23:04, otherwise it would display as 23:4)

#### checkSeconds

Identical to **checkMinutes** but for seconds instead.

The remainder of the javascript code is in regards to the task manager and weather application. The task manager code is structured as the following:

### Main structure

Initialize variables by referencing HTML elements

Add EventListeners to allow dragging of elements

Add functions for (dragStart, dragEnd, dragOver, dragEnter, dragLeave and dragDrop events)

### Modal structure

Initialize variables by referencing HTML elements

Add EventListeners to open and close Modal

### Task Creation structure

Initialize variables by referencing HTML elements

Add EventListeners to allow submission of task

Get input from user

If <input is empty> then

    Alert <request input>

Create div element for task

Create and Append text to div element

Make div element *draggable*

Create span

Create and Append text to span

Add EventListener to span to Remove div element

Append div element to main structure

Add EventListeners to allow *dragging* of tasks

Clear input window and Close Modal

The weather app code is structured as the following:

Fetch data of all countries from 'https://restcountries.com/v3.1/all':

If <successful> then

    Extract common country names and sort alphabetically

    Populate select dropdown with country names

If <error occurs> then

    Log error and alert user to retry fetching data

Function getWeather

*Uses apiKey and country input from user to construct URLs for weather*

    If <country is empty> then

        Alert user to enter a country name

        Exit function

    Fetch currentWeatherUrl

        If <successful> then

            Display current weather data

        If <error occurs> then

            Log error and alert user

Fetch forecastUrl

If <successful> then

Display hourly forecast data

If <error occurs> then

Log error

Function displayWeather

*Uses data from getWeather to populate HTML page with content*

Clear existing content in weather-related divs

If <data is "404"> then

Display error message

Else

Extract country name, temperature, description, and icon from data

Display collected data

Function displayHourlyForecast

Clear existing content in hourly forecast div

Extract data for next 5 hours

For <each hour's data> then

Extract hour, temperature, and icon information

Display hour, icon, and temperature

Function showImage

Show weather icon image

## CSS

The CSS for the website consists of one file, style.css. It is constructed from the top-down in the format:

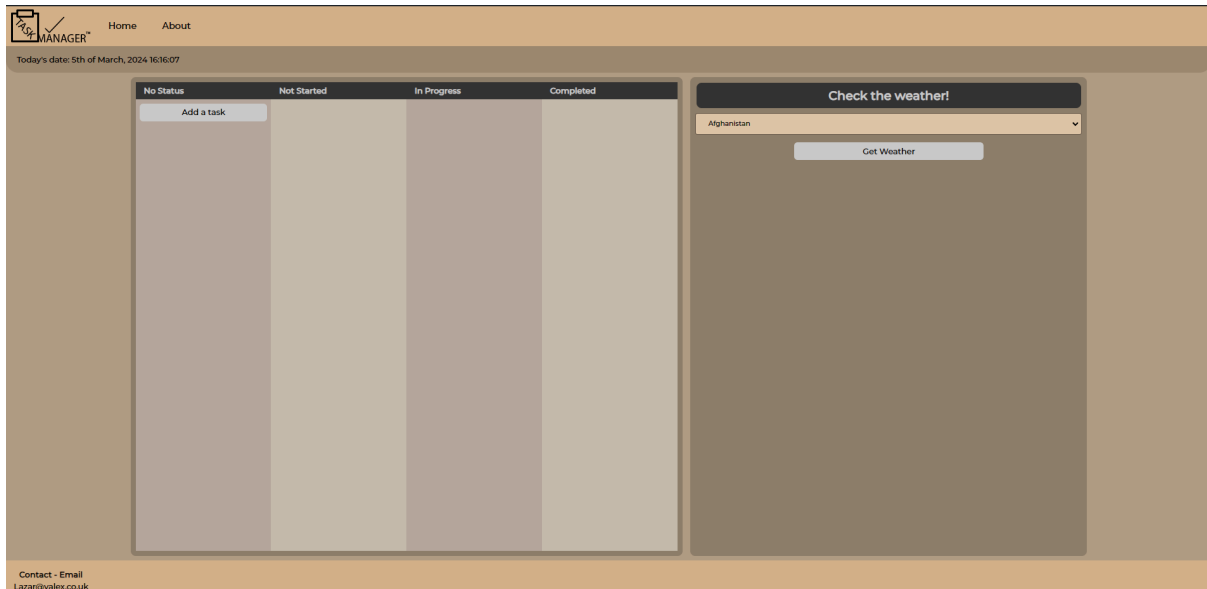
Medium/Smaller screens -> Larger screens

Other than this, there is nothing in particular that should be noted about this file. Anything that isn't overwritten by the larger screens section in the latter part of the document, will inherit its format and style from the medium/smaller screens section. This would include things such as font family, and the ability to select and highlight text.

The reason as to why I decided to remove the ability for the user to highlight and select text is because there is no element of my website that would require this, apart from potentially needing to copy my email from the footer. As it caused issues with the task manager element that accepts droppable elements, this way it nullifies any such problems.

## Function

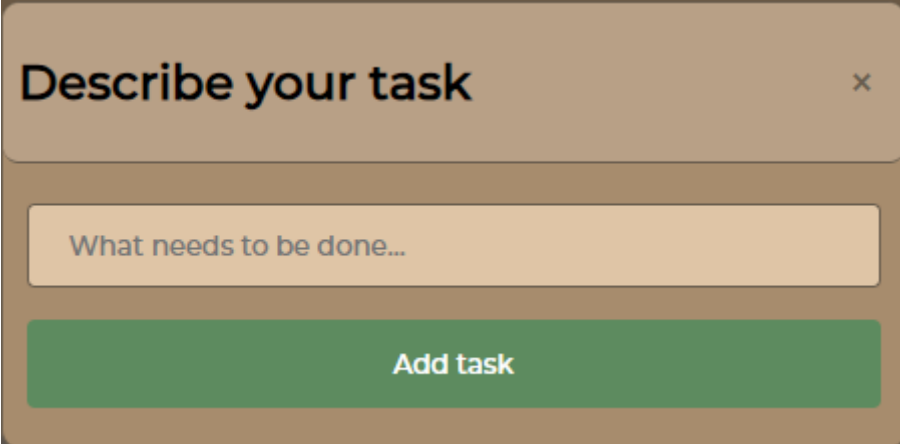
When arriving to the landing page, index.html, you are greeted with my custom made logo, a simple but functional header, today's date and time, precise to the second, the two main elements, (task manager and weather app), and finally a footer at the bottom of the page with my contact information listed.



*Index.html landing page*

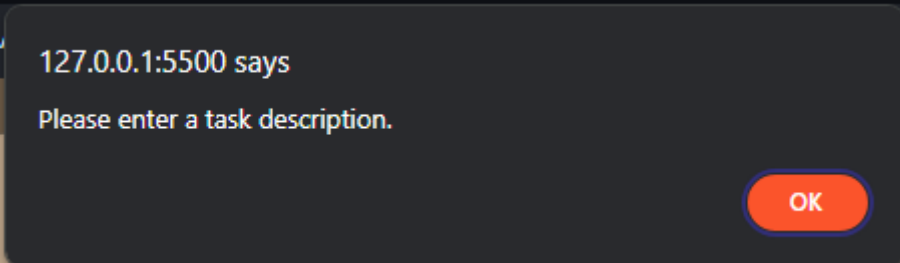
Should the user wish to add a task, they can simply press the Add a task button located under the “No Status” container of the task manager. Pressing this will in turn prompt the user to “Describe your task” and hints with a placeholder text within the input box “What needs to be done...”

The user is now directed towards the task input box and can now fill in the task they wish to log in their task manager. Once completed, they can press the “Add task” button to add the task to the “No Status” section of the task manager, right under the “Add a task” button. If the user wishes not to add a task at this time, they can exit this modal interface by two means; either click the cross button to the top-right of the interface, or simply click anywhere outside the modal. Both of these options will exit out the modal interface and clear any input the user initially had put in the box.



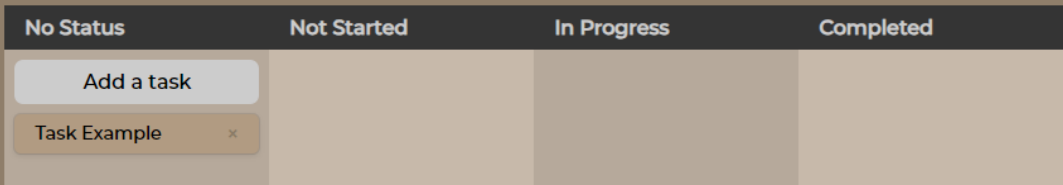
*Task modal requesting task description*

If the “Add task” button is pressed *without* filling out any text in the input box, the system will prompt the user to “Please enter a task description”.



*Empty task descriptions warning*

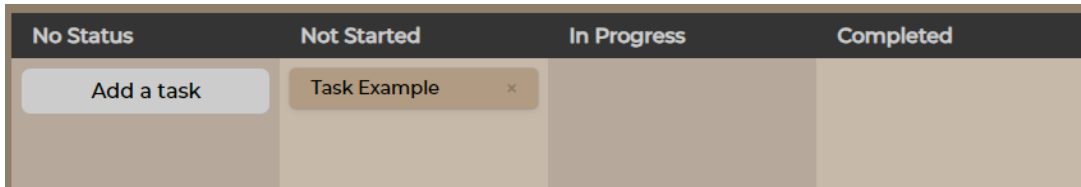
Once a task is added, it can be moved around between the four different status sections of the task manager.



*Task added to the “No Status” category*



*Task being dragged and hovering over the “Not Started” category*



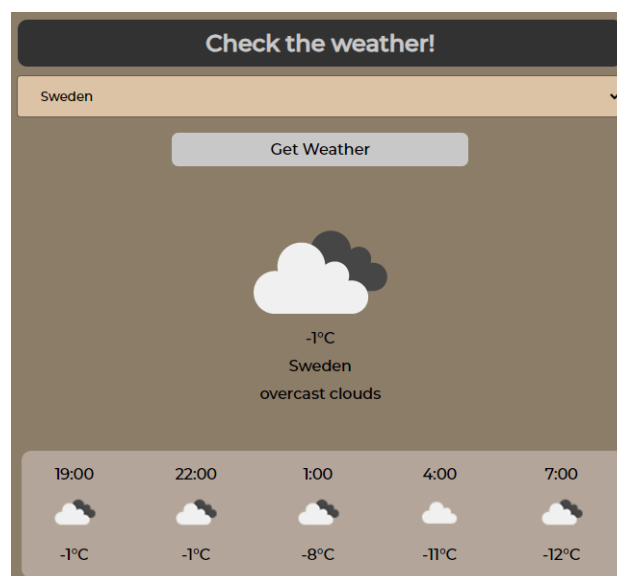
*Task dropped in “Not Started” category*

If a task has to be removed from the task manager, this can be done by pressing the cross to the right-most part of the task.

The weather app is now conveniently placed to the right, (on larger screens), or below, (for smaller screens), of the task manager. Here the user can select a country from the dropdown list and press the button “Get Weather”. This will display to the user;

- An icon, representing the current weather conditions
- The temperature in degrees celsius
- The country name
- A text description of the weather conditions
- A forecast over the next 12 hours using timestamps, icons and temperatures in celsius

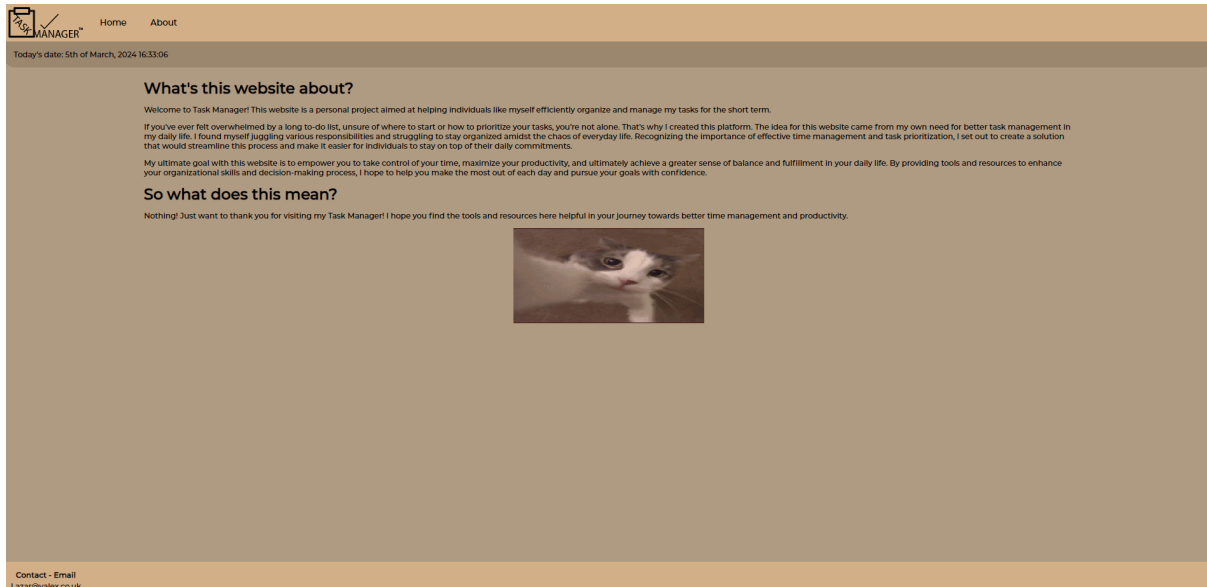
The user can use this information to better plan their tasks and rearrange them depending on what they believe is appropriate in regards to the weather (if so is applicable)



*Weather application showing current and upcoming weather in Sweden*

Moving on to about.html, this is a simple section retaining some information from index.html such as the header, footer and date elements.

This section hosts a “What’s this website about?” description where I go into some detail regarding why this website was created, why I felt inspired to make it and what I hope to achieve with it. All this alongside a thank-you message for visiting! 😊



*About.html landing page*