|  |
| --- |
| Dalhousie University |
| CSCI 5409 | Assignment 1 |
| Cloud Computing |

|  |
| --- |
| Navneet Singh (B00810744)  5-19-2019 |

Table of Contents

[Description of local environment in terms of hardware highlighted and OS 3](#_Toc9291004)

[Setting Up Local Environment 3](#_Toc9291005)

[Frameworks used 4](#_Toc9291006)

[Sails Framework 4](#_Toc9291007)

[Setting up Sails on Windows 10 4](#_Toc9291008)

[Setting up Sails on Ubuntu 5](#_Toc9291009)

[Node JS 5](#_Toc9291010)

[Installing node js on Windows, Ubuntu or MAC OS 5](#_Toc9291011)

[Express JS 6](#_Toc9291012)

[Installing express on Windows, Ubuntu or MAC OS 6](#_Toc9291013)

[Bootstrap 6](#_Toc9291014)

[Libraries used 7](#_Toc9291015)

[jQuery 7](#_Toc9291016)

[Integrating jQuery with assignment 7](#_Toc9291017)

[Overview of frontend 8](#_Toc9291018)

[Design of frontend 8](#_Toc9291019)

[Help Section 9](#_Toc9291020)

[Validation of Input Fields 9](#_Toc9291021)

[Description of web-services, including end points and parameters 10](#_Toc9291022)

[Description of backend 11](#_Toc9291023)

[Description of deficiencies 12](#_Toc9291024)

[Testing & Submission 12](#_Toc9291025)

[Identification of source code 16](#_Toc9291026)

# Table of figures

[Figure 1: Description of local enviroment in config/local.js 3](#_Toc9291080)

[Figure 2: Current version number of sails 5](#_Toc9291081)

[Figure 3: Current version number of node 5](#_Toc9291082)

[Figure 4: Bootstrap Added to Project 6](#_Toc9291083)

[Figure 5: JQuery added to the project 7](#_Toc9291084)

[Figure 6: Frontend for the application 8](#_Toc9291085)

[Figure 7: Responsive frontend for the application 8](#_Toc9291086)

[Figure 8: Help section for the BMI Calculator 9](#_Toc9291087)

[Figure 9: Validating input fields for BMI Calculator 9](#_Toc9291088)

[Figure 10: MVC framework followed by the project. 10](#_Toc9291089)

[Figure 11: API response for the BMI Calcultor 11](#_Toc9291090)

[Figure 12: Code for BMI Response in backend 11](#_Toc9291091)

[Figure 13: Test case for testing correct values 12](#_Toc9291092)

[Figure 14: Test case for validating input height 12](#_Toc9291093)

[Figure 15: Test case for validating input weight 13](#_Toc9291094)

[Figure 16: Test case for submitting with empty fields 13](#_Toc9291095)

[Figure 17: Test case for help section of the application 14](#_Toc9291096)

[Figure 18: Test case when the backend server is down 14](#_Toc9291097)

[Figure 19: Test case when the internet connection is slow 15](#_Toc9291098)

[Figure 20: Test case when wrong input is entered in API 15](#_Toc9291099)

[Figure 21: Test case for blank height and weight 15](#_Toc9291100)

# Description of local environment in terms of hardware highlighted and OS

Operating system used for setting up the local environment is Windows 10 Professional. However, home edition would also work. Any other operating system like Ubuntu and MAC OS would also work, since, the basic hardware requirements needed to run sails is fulfilled by every operating system.

Usually the local environment settings is available at config/local.js, where other dependencies like setting up mail to send mails to people via localhost is done, storing database passwords, etc.

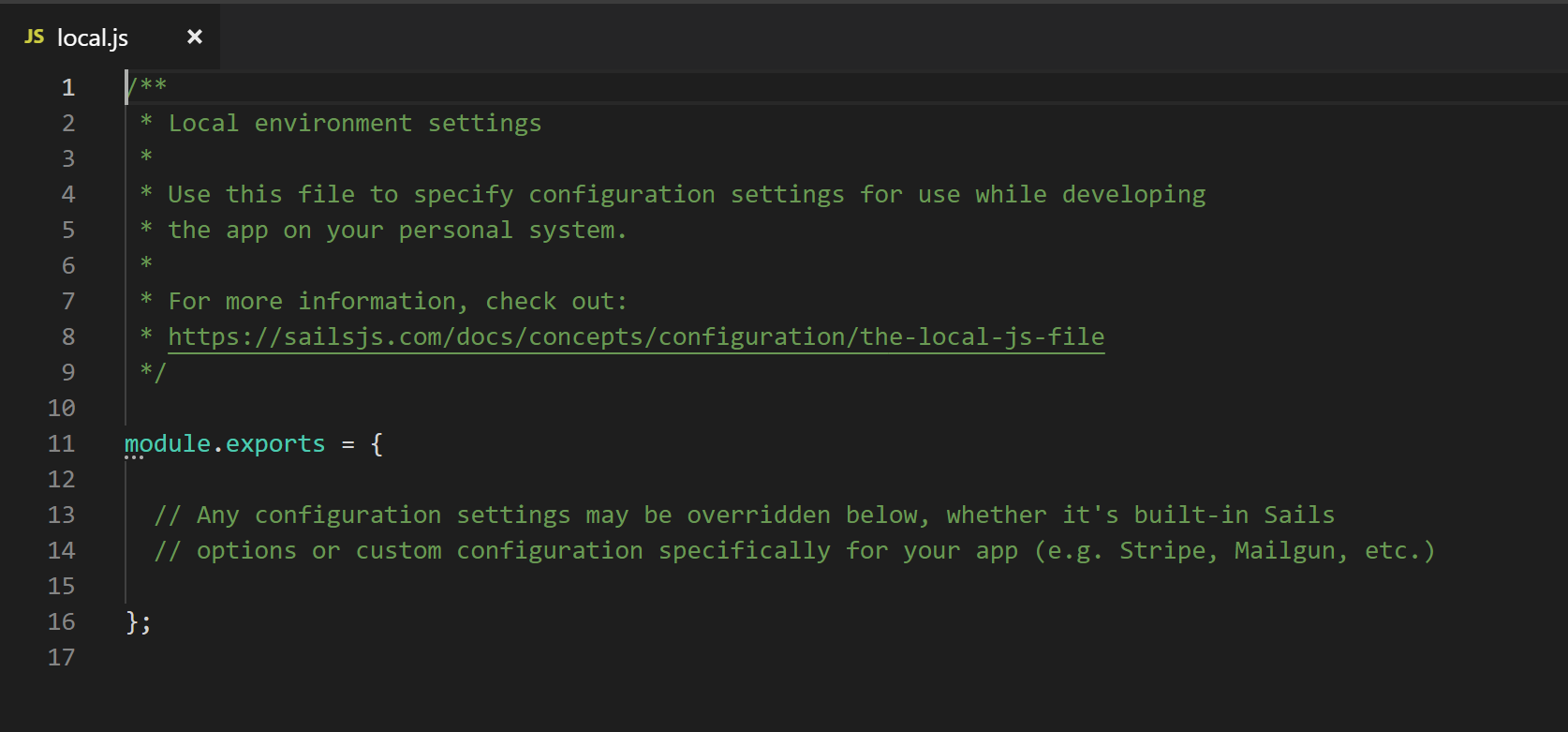


Figure 1: Description of local enviroment in config/local.js

# Setting Up Local Environment

STEP1: Start by cloning the project or downloading the project on Gitlab : <https://git.cs.dal.ca/navneet/cloud-computing-assignments>. You can clone with ssh or https using the following command

SSH: git@git.cs.dal.ca:navneet/cloud-computing-assignments.git

HTTPS: <https://git.cs.dal.ca/navneet/cloud-computing-assignments.git>

STEP2: Once you have the necessary files, make sure that you have the local environment set up Follow the following steps for Windows, Ubuntu or MAC. You can download the following software here:

1. Node package manager: <https://nodejs.org/en/download/>
2. Once the node package manager is installed, open command prompt and run the following command: sudo npm -g install sails
3. Next download node js here : <https://nodejs.org/en/#download>. Select the OS and follow the setup instructions and it would install automatically.
4. Once node is installed, express is installed automatically and that would be enough to run the application.
5. Once all the necessary files have been installed navigate to the folder downloaded from gitlab in step 1.
6. Open command prompt and navigate to assignment1-Api folder (using cd Assignment1-Api) and type **“sails float”** in the command prompt.
7. Open command prompt and navigate to assignment1-Frontend (using cd (using cd Assignment1-Api) folder and type **“sails float –port 1338“** in command prompt.
8. Make sure you enter port number 1338 for the assignment1-frontend folder because the api is required to have a port number 1338. It talks to the backend api present on 1337 to make the entire system work.

All the above mentioned steps are explained in more detailed in the next section.

# Running the API

1. Once all the necessary software has been installed, and the code is running on localhost, to access the API via postman or simply hitting the URL, use the following link to test the API:

<http://localhost:1337/calculate744/?height=1.75&weight=70>

# Frameworks used

Sails framework is a REST based framework which is build on top of Node js, express and provides an all-in-one MVC framework.

## Sails Framework

Sails is an MVC style node framework which provides rapid development of server side application using JavaScript. Since it uses an MVC framework, the responsibility of the project can be divided into different modules. Applications build using sails scale up very quickly and can be used to build enterprise level application as well.

One of the mail advantages of using a framework is that it is combined by loosely coupled classes and components i.e the classes are highly customizable. You can add, remove components of framework without comprising the overall stability of the project.

Under the hood, sails is one of the well known frameworks build on top of Node.js. Express is another framework which is used. Express isn’t much concerned about the structure of the project and this is where things get a little messy. However, sails provided a structure and control of the files.

### Setting up Sails on Windows 10

In order to run sails, you need to install Node package manager. Node package manager is available on the main website of node. Available here: <https://nodejs.org/en/download/> .Once the node package manager is installed, simply run the following command to install sails:

Npm -g install sails

This would install sails globally, removing the -g would install it in your specific directory and wouldn’t be accessible if -g is not included.

### Setting up Sails on Ubuntu

In order to run sails on Ubuntu, we need the node package manager. Install the node package manager from here <https://nodejs.org/en/download/>. Once the node package manger is installed, simple run the following command to install sails:

Sudo npm -g install sails

This would download all the necessary files required for sails.

You can check the version number of sails using sails –version on windows:

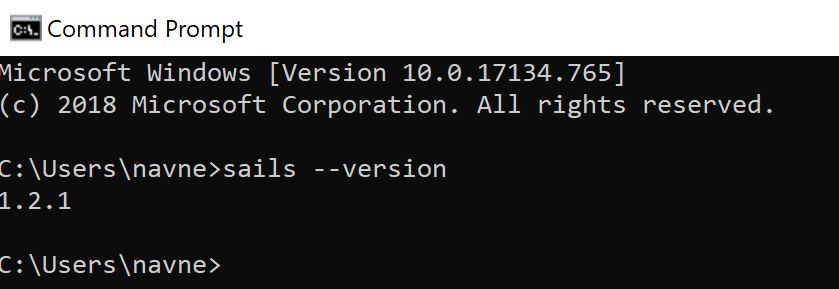


Figure 2: Current version number of sails

## Node JS

Node js is the main part of the project. It works with express to provide a seamless experience to the user to make REST request. Node is an open source run time environment that executes JavaScript outside the browser. It aims to create scalable applications.

For the BMI application, node is responsible for creating the local host server over which our sales application works. The entire environment to make a REST API work is provided by node framework.

### Installing node js on Windows, Ubuntu or MAC OS

Node js installer can be found here <https://nodejs.org/en/#download>. The installer is available for all major operating systems. Simple select the operating system and start installing.

To check the version number on a window command prompt, run the following command:

Node –version

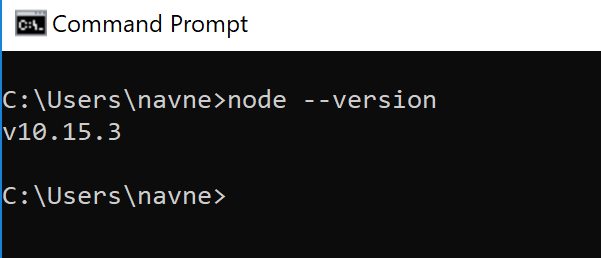


Figure 3: Current version number of node

## Express JS

Express is an open sourced web framework for node js. It was designed to provide REST framework structure for the application. Express is used by a lot of major companies like IBM, Uber, PayPal, etc.

### Installing express on Windows, Ubuntu or MAC OS

Once we have installed npm manager, we can simply use npm manager to install express for us. Simply run the following command on command prompt

Npm install express –save

This would install express for us and we can now leverage the API structure that it provides along with the framework.

## Bootstrap

Bootstrap is one of the most powerful frameworks used for designing the front end part of the application. By simply adding some predefined bootstrap classes to the project improved the overall design of the application.

To add bootstrap to the project simply add the following stylesheet link to the header of the project as shown below:

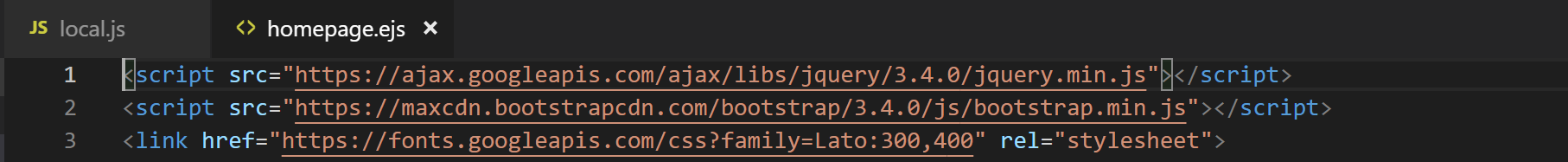


Figure 4: Bootstrap Added to Project

# Libraries used

## jQuery

To make the frontend interact with the backend API, JQuery is used. jQuery is a JavaScript library specifically designed to manipulate HTML DOM as well event handling and Ajax. For this specific assignment, jQuery’s event calls like click and keypress was used. Furthermore, ajax request was also used to make the frontend interact with the backend.

### Integrating jQuery with assignment

jQuery cloud version was used for the purpose of our assignment. We simply added the jQuery link in the header and whenever the website would load, it would connect with the cloud where jQuery was hosted and return the necessary features. The necessary was integrated with the project as shown below.

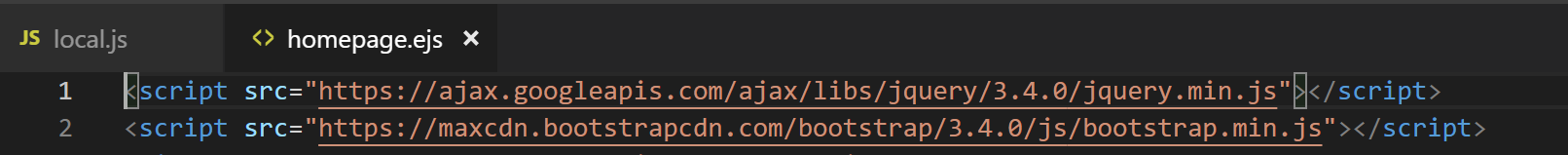


Figure 5: JQuery added to the project

# Overview of frontend

## Design of frontend

Frontend of the website was designed in such a way to make it easy for the user to understand what the purpose of the field or button is. For example consider the following screenshot of the frontend:

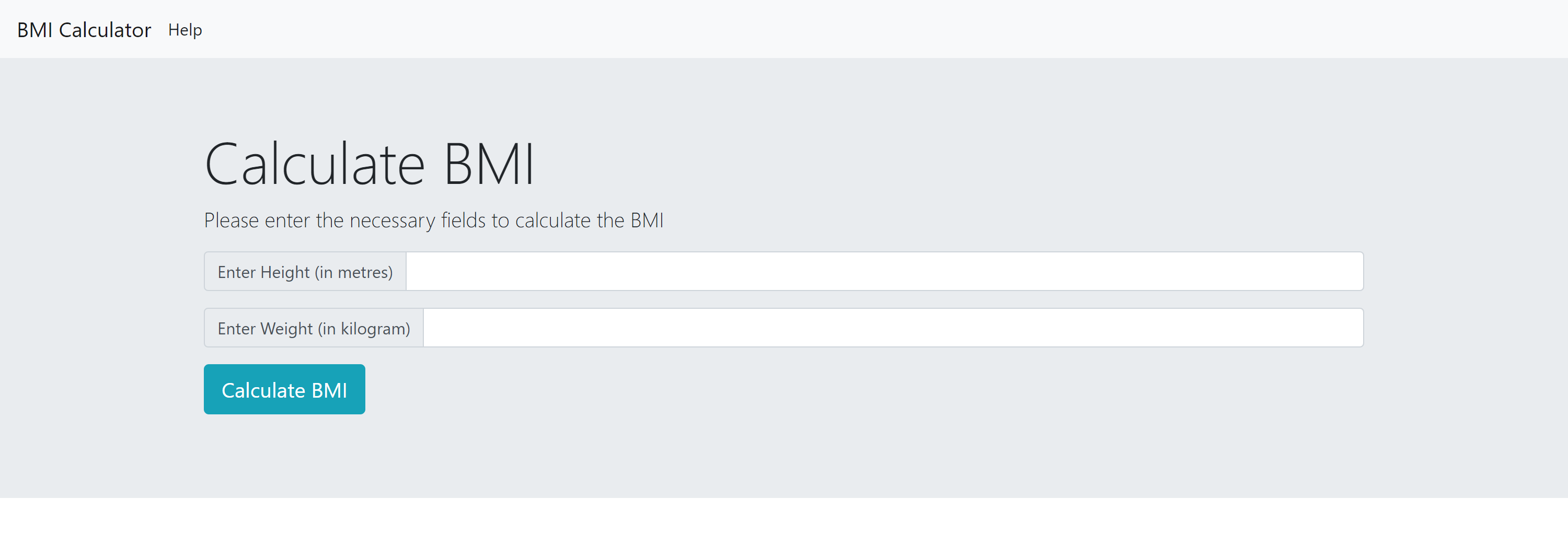


Figure 6: Frontend for the application

We can see that the message has been shown to the user as to what to enter in the input field. It has also been mentioned that the user has to enter their heights in meters and weight in kilogram. The button is self descriptive that it calculated the BMI of the values entered.

The entire frontend is designed in such a way to be accustomed to any screen size as shown below:

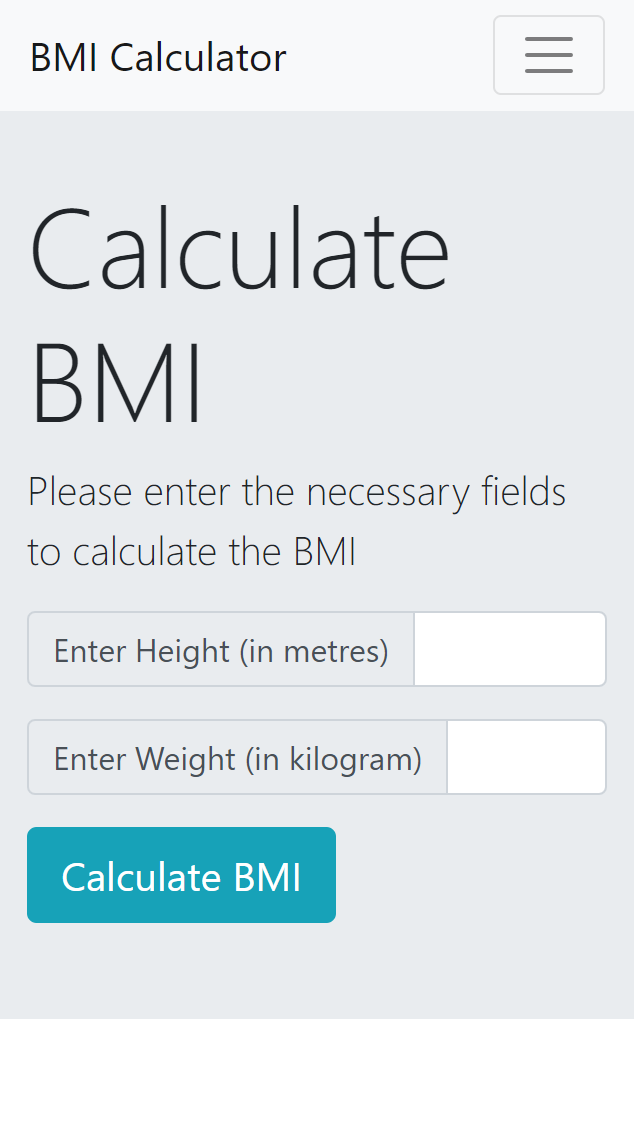


Figure 7: Responsive frontend for the application

## Help Section

Help section has also been added, so that the user can know more about the application as shown in figure

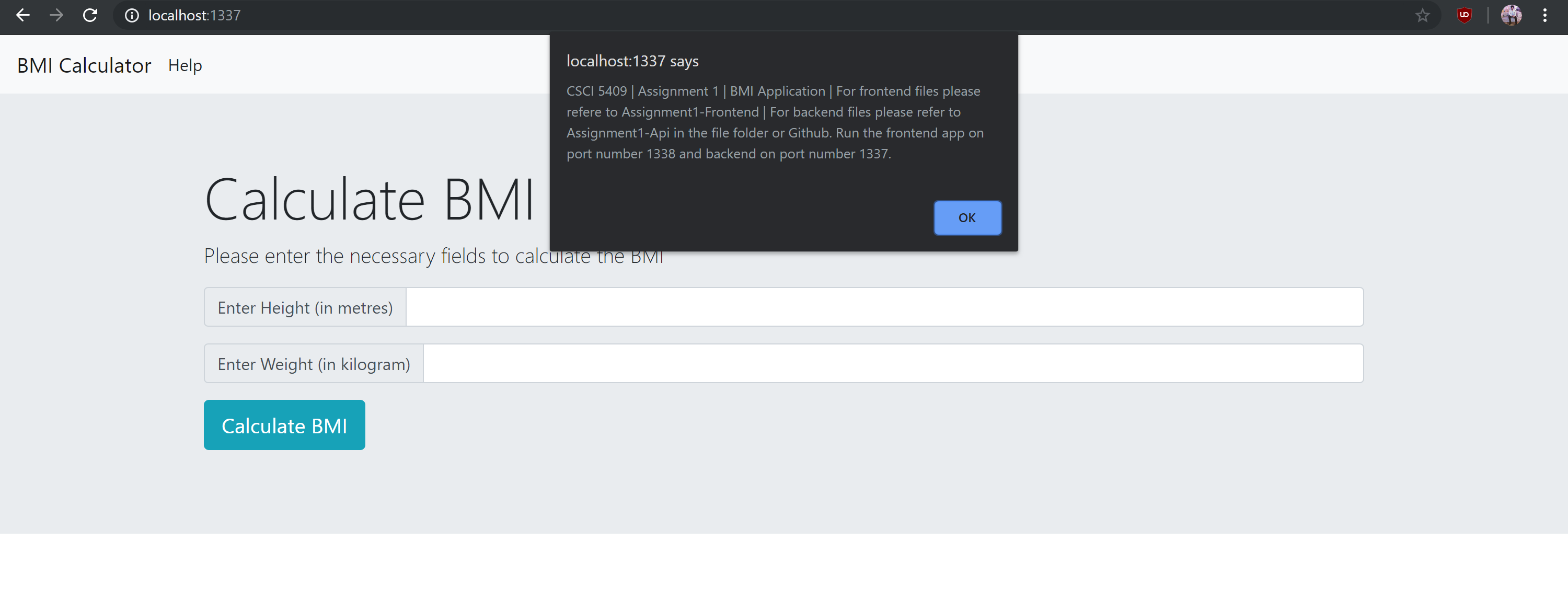


Figure 8: Help section for the BMI Calculator

Whenever the user clicks on help button on the navbar, a popup appears which shows the necessary messages to help user know more about the application.

## Validation of Input Fields

On the spot validation is provided for the user using jQuery. Whenever the user starts typing the height or the weight, on the spot validation is provided and highlighted in red incase the height or weight is invalid or blank. Using this technique helps user provide on the spot validation and improve the overall quality of the application.

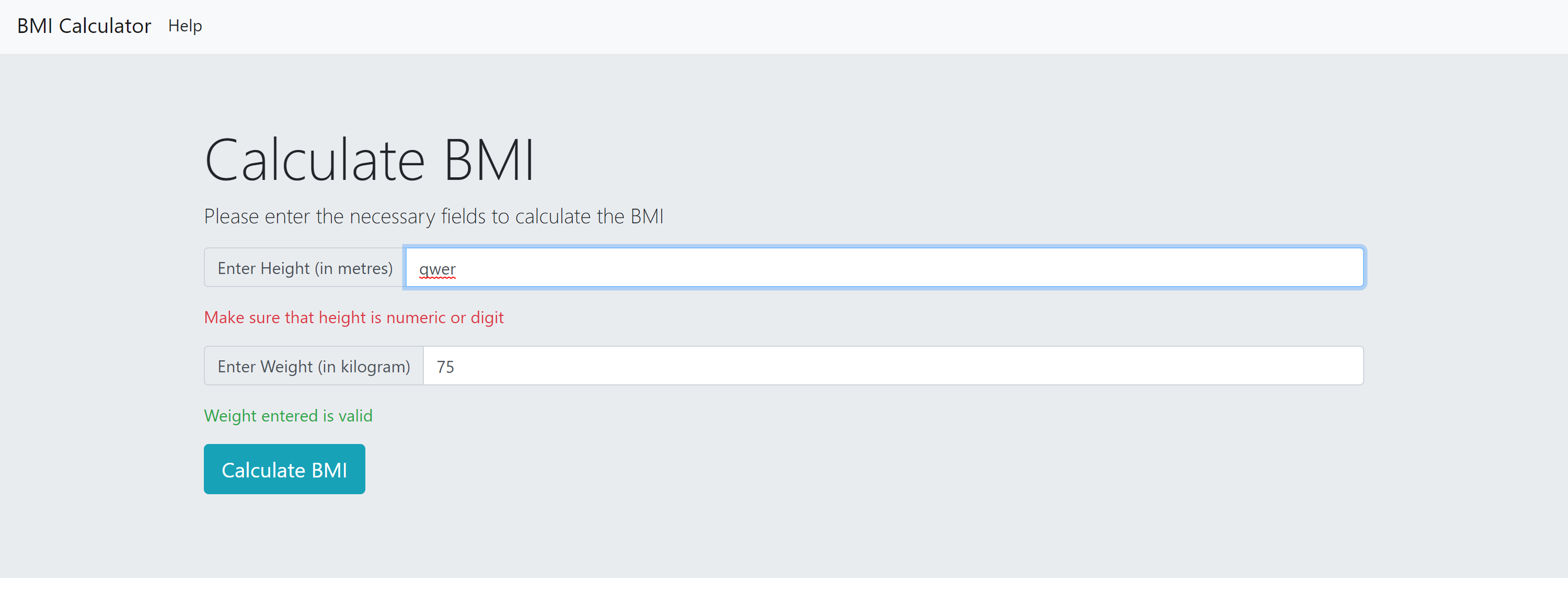


Figure 9: Validating input fields for BMI Calculator

# Description of web-services, including end points and parameters

In order to understand the purpose of web-services in our application we need to understand the MVC model associated with it. The view always interacts with the controller and controller further talks with the model. Model further returns the response back to the controller and the controller further interacts back with the view as shown in figure<<enter fig name>>.

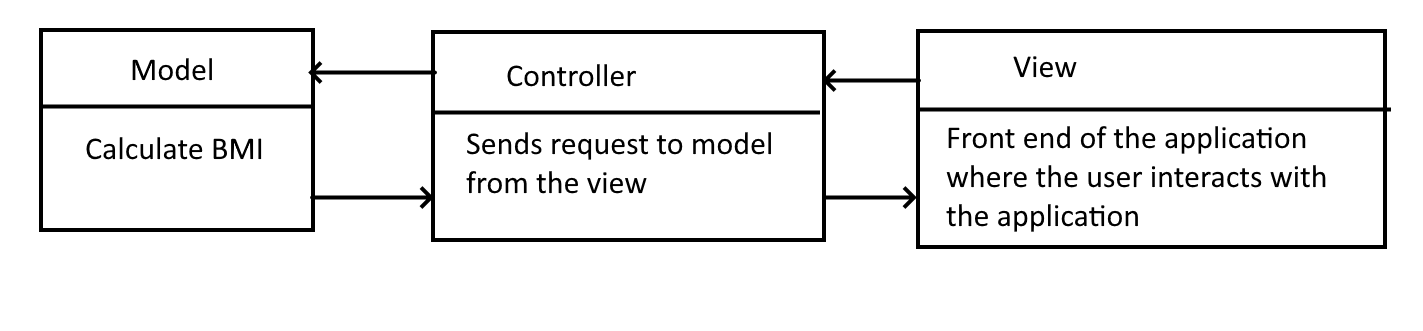


Figure 10: MVC framework followed by the project.

The part which includes the model and the controller together combines the web-services of the application. Here we define the routes which are accessed by the view and model to send and retrieve information to the user.

In our application, whenever the user enters the height and weight to calculate the BMI in the view, it hits a particular URL on the controller. The controller further interacts with the model where the calculation takes place and returns back the calculated message to the controller and controller further returns back the message to the user.

Endpoints and the parameter is shown in the following table:

Table 1: API end-points for BMI Calculator.

|  |  |
| --- | --- |
| End points | Purpose |
| http://localhost:1337/calculate744/?height=180&weight=75 | This is the API which the frontend hits when a user enters the height and weight.  Calculate744 is the name of the route which is hit where 744 is the last three digits of my B00 number.  Height and weight are the optional parameters passed to the end point. |
| /calculate744 | GET request is passed which hits the function of the controller in the routes. |

# Description of backend

As shown in the MVC diagram in figure <<insert figure here>>, the web-service generated here interacts with the backend and is responsible for carrying out the main calculation of the BMI. Here, a key-value pair has also been created to show the message related to the BMI. Let us consider the following API:

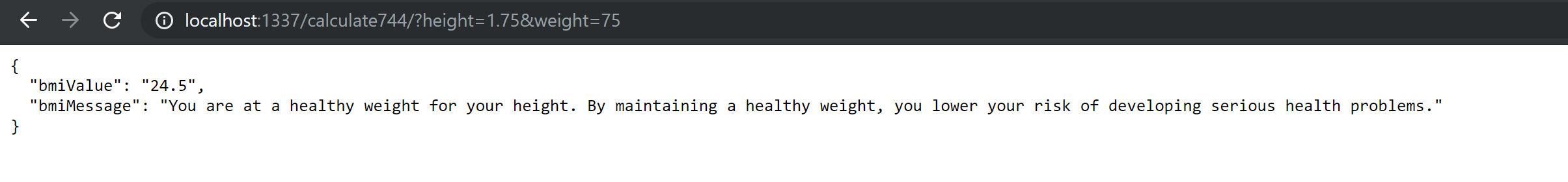


Figure 11: API response for the BMI Calcultor

Within figure <<figure name>> we observer the optional parameters that are passed to the API i.e height and weight and the calculation is done and bmi value is returned along with the message from the hash table in the backend.

To have a better understanding of the backend consider the following screenshot of the code:

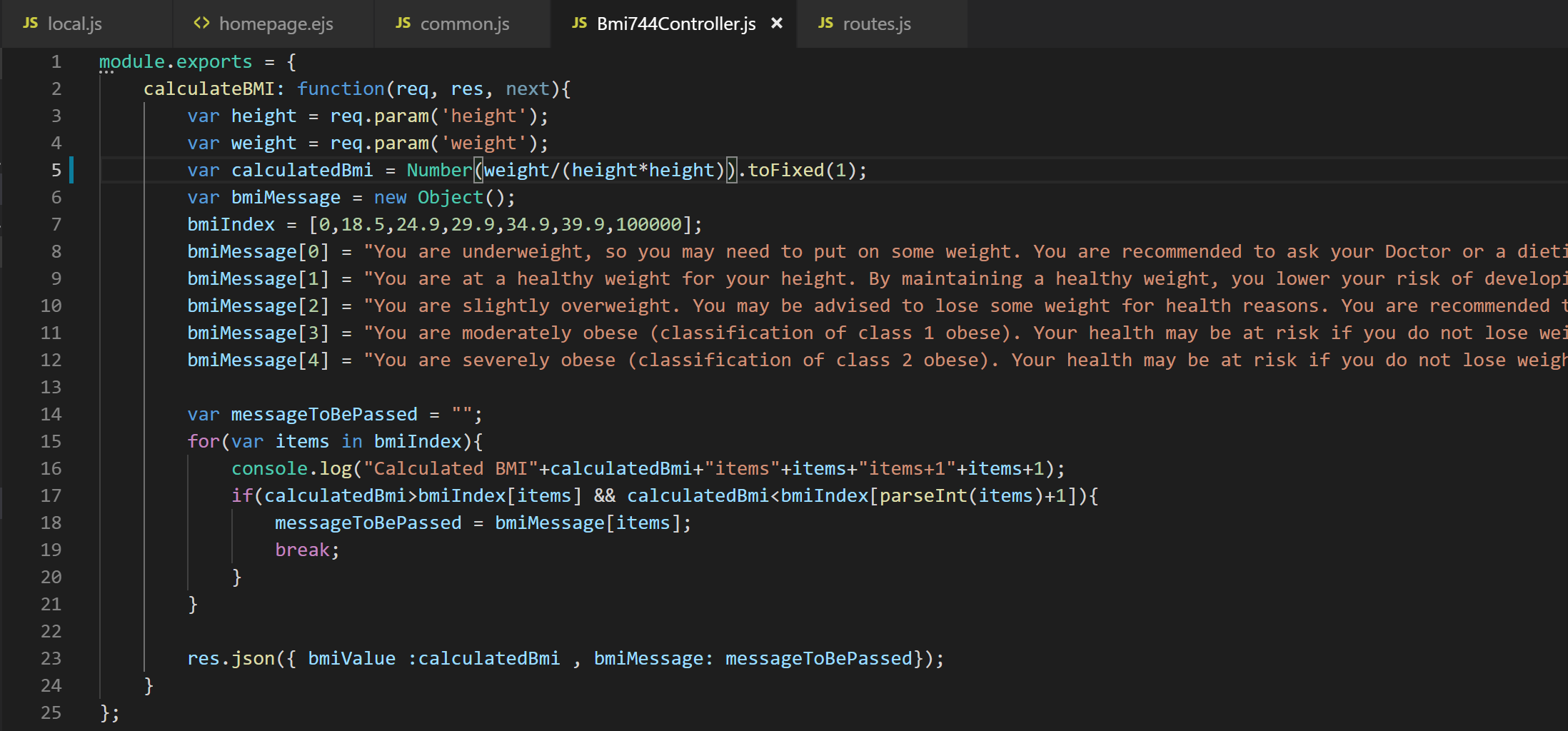


Figure 12: Code for BMI Response in backend

Within the lines 1 – 5, the node js code taking the input from the query string as height and weight and are calculating the Bmi using the BMi formulae. Further the BMI is rounded off to the nearest 1 digit integer and returned as a response.

In order to handle the messages received along with the BMI values, BMI index is created and messages are passed at these index, such that whenever the index are parsed it checks within what range does it fall and sets the appropriate message as shown in lines 14-22.

Finally in line number 23, the json response is returned along with the BMI value and message passed.

# Description of deficiencies

As far as the assignment is concerned, all the requirements were fulfilled and no deficiencies were found.

# Testing & Submission

To test it thoroughly, the following test cases were covered.

Test Case 1: Testing it for correct values.

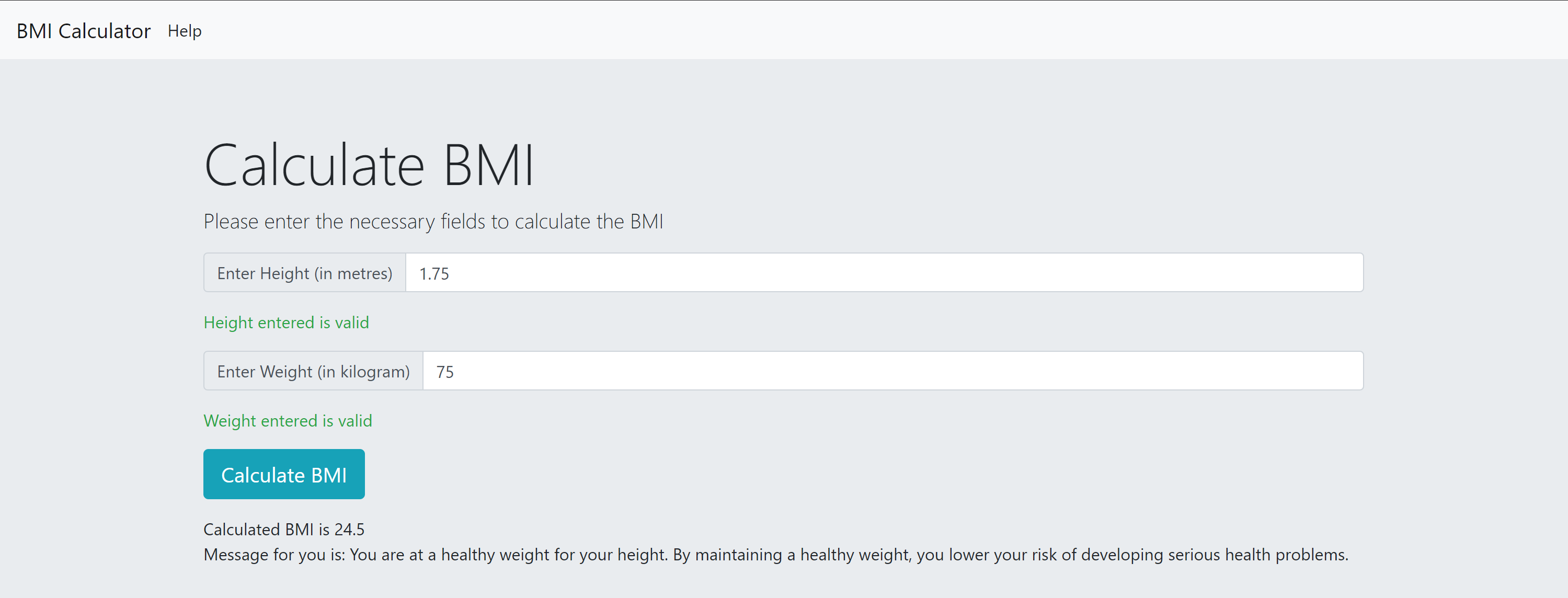


Figure 13: Test case for testing correct values

As we can see that both height and weight is valid input and the response is received along with the message, hence the api is working fine for correct values.

Test Case 2: Validating the input height.

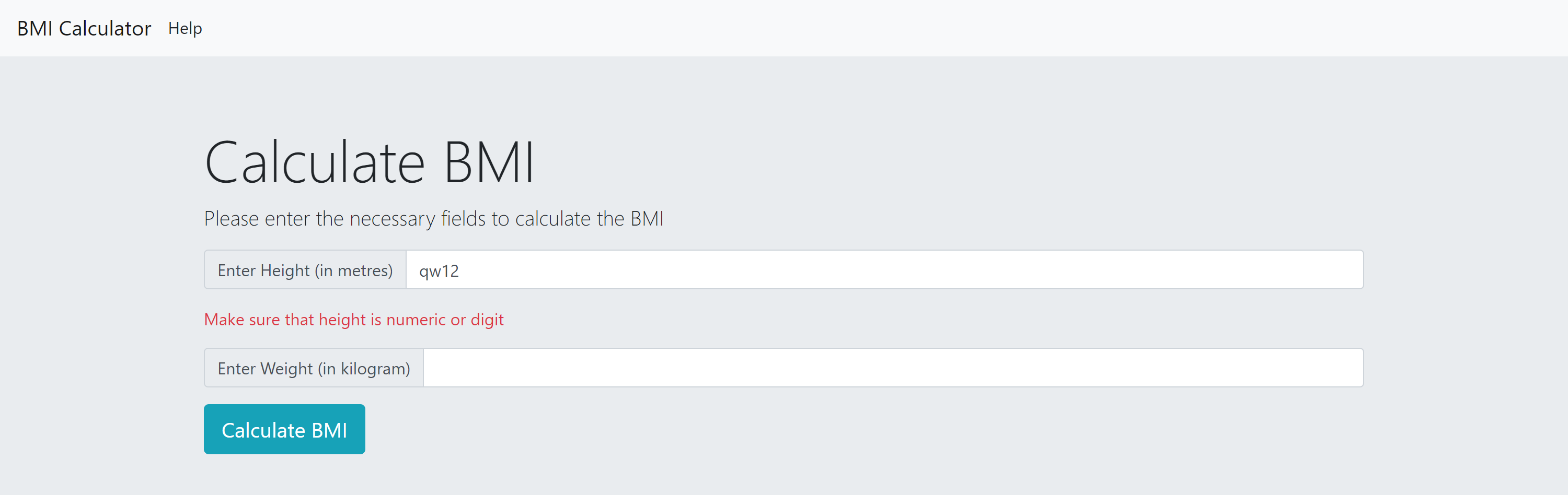


Figure 14: Test case for validating input height

In case the height entered is non-numeric then real time validation is provided to the user that the height entered should be numeric.

Test Case 3: Validating the input weight

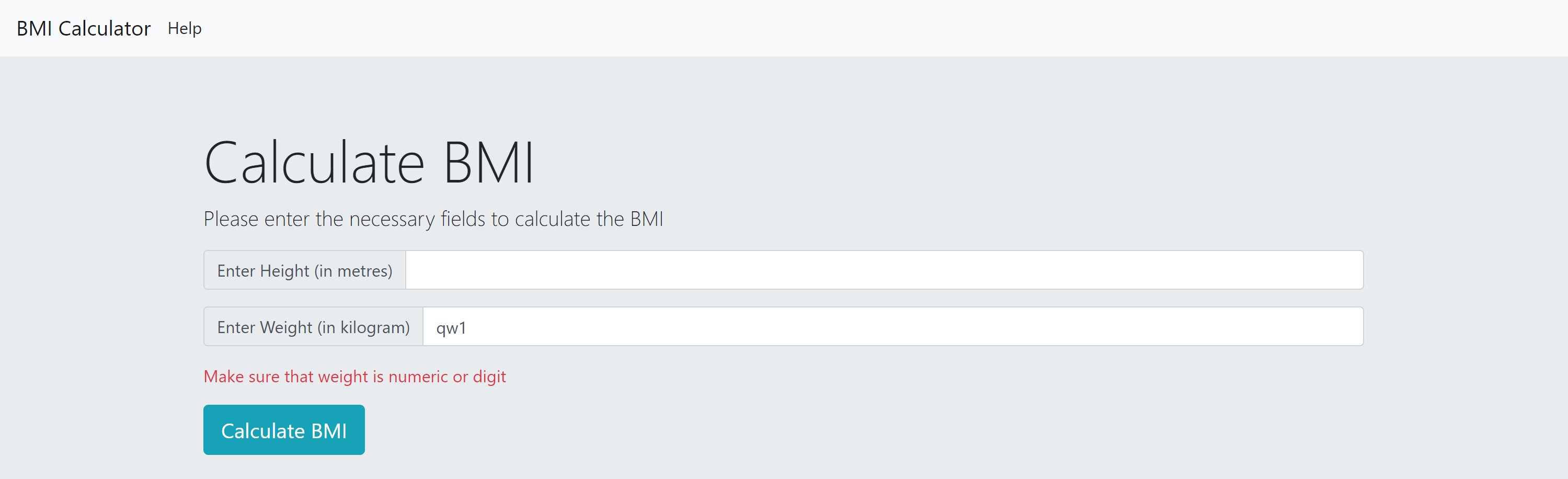


Figure 15: Test case for validating input weight

In case the user enters non-numeric input for weight, then real time validation is provide to the user to make sure that they enter should be numeric.

Test Case 4: Submitting without entering height and weight.

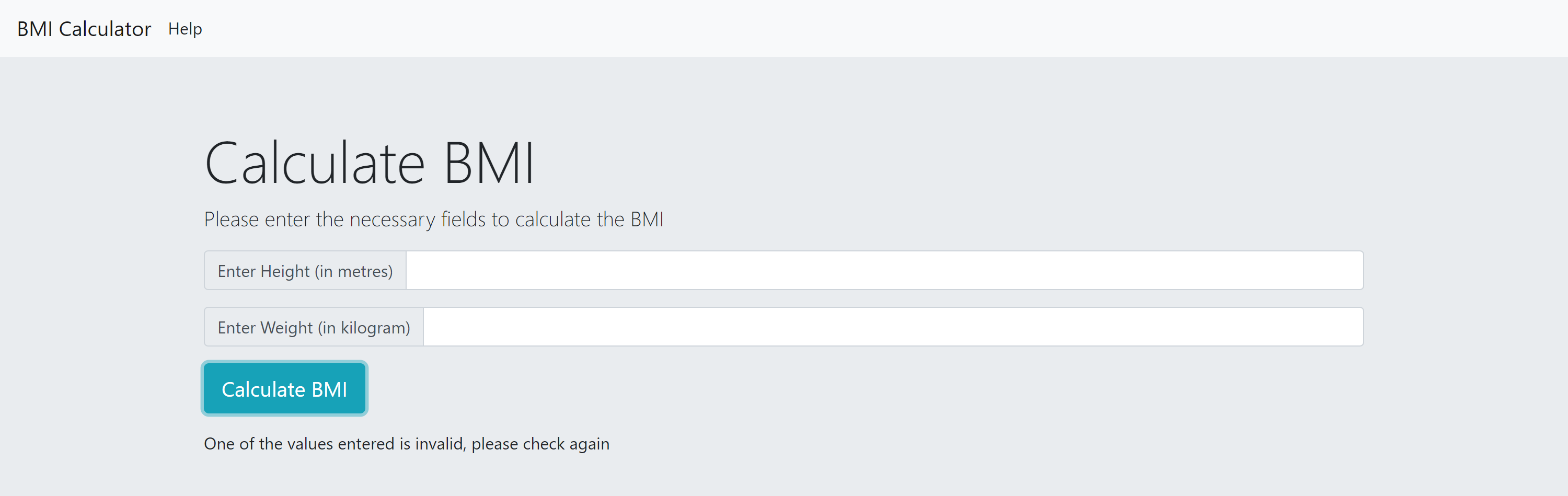


Figure 16: Test case for submitting with empty fields

Whenever the user clicks on calculate BMI without entering the values, a message is shown that values entered is invalid. Please check the values again.

Test Case 5: Providing help section to the user

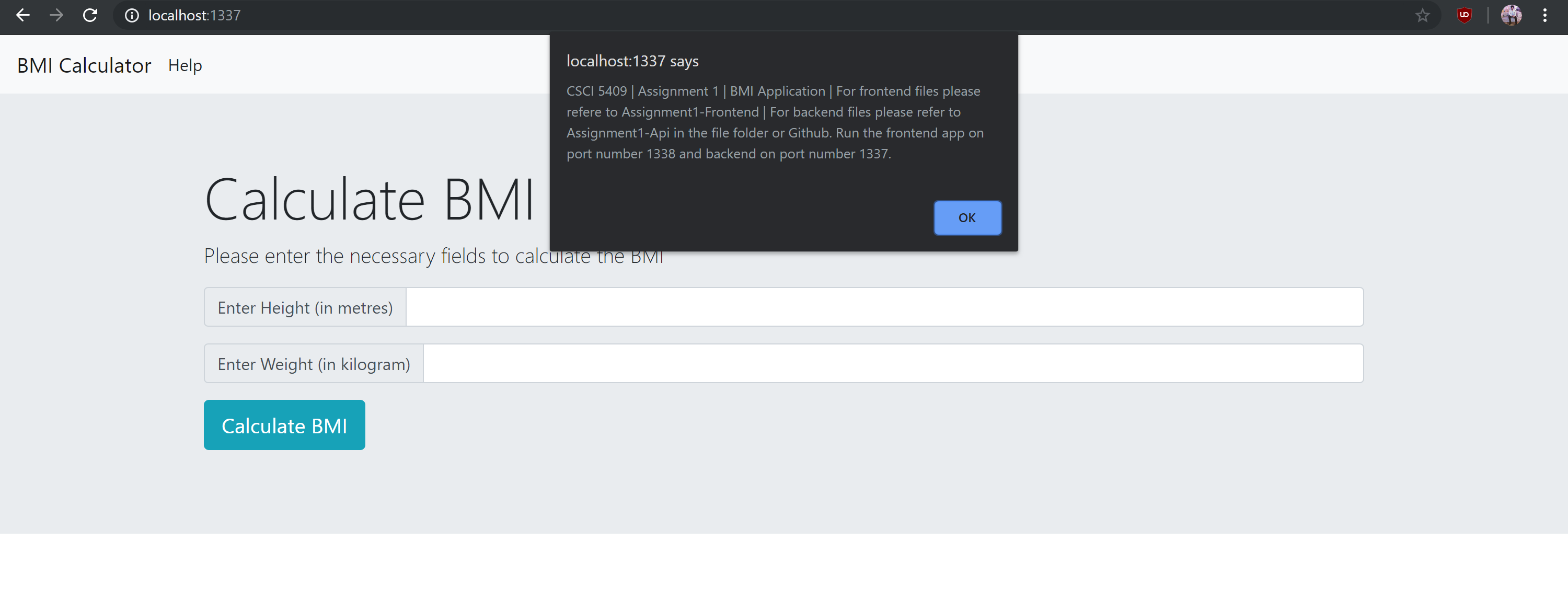


Figure 17: Test case for help section of the application

Help section is provided to the user to make sure they know what the application does.

Test Case 6: When the backend-server is down.

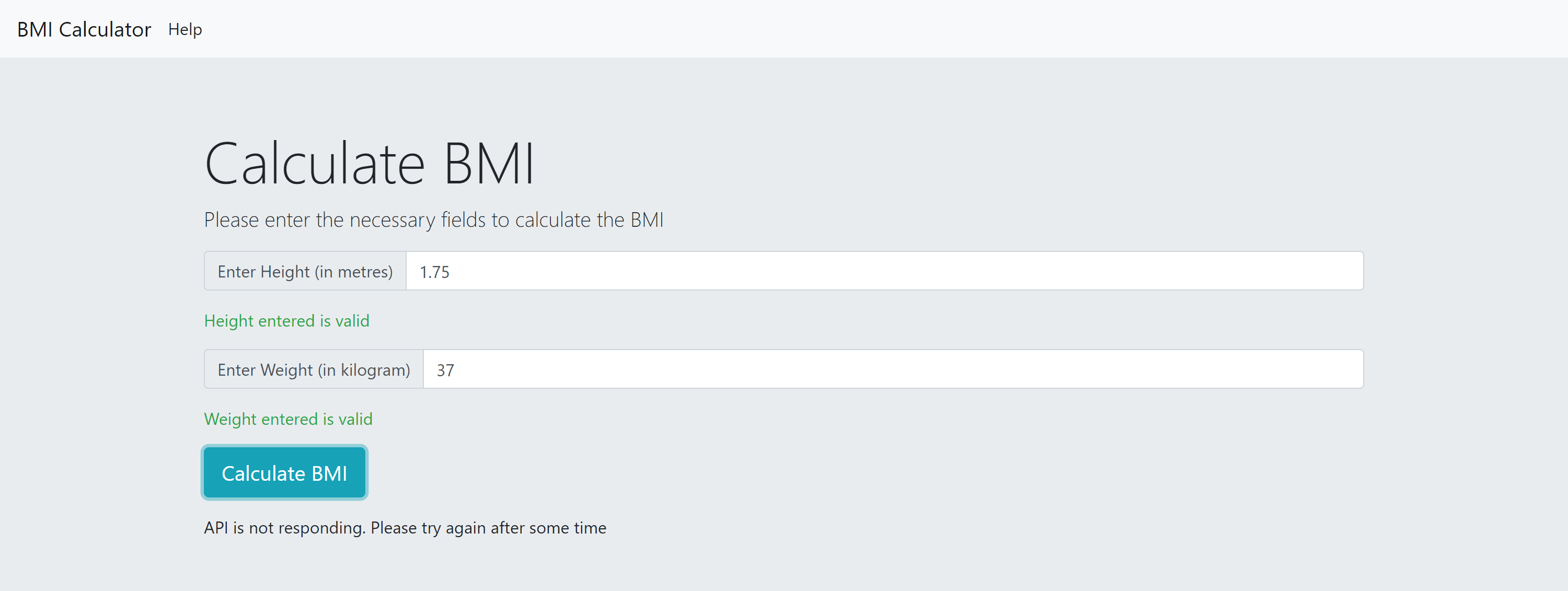


Figure 18: Test case when the backend server is down

Within the image shown in figure <<figure name>>, a request was sent to the server but the API was down, so error message related to the server is shown in the figure.

Test Case 7: When the internet connection is slow

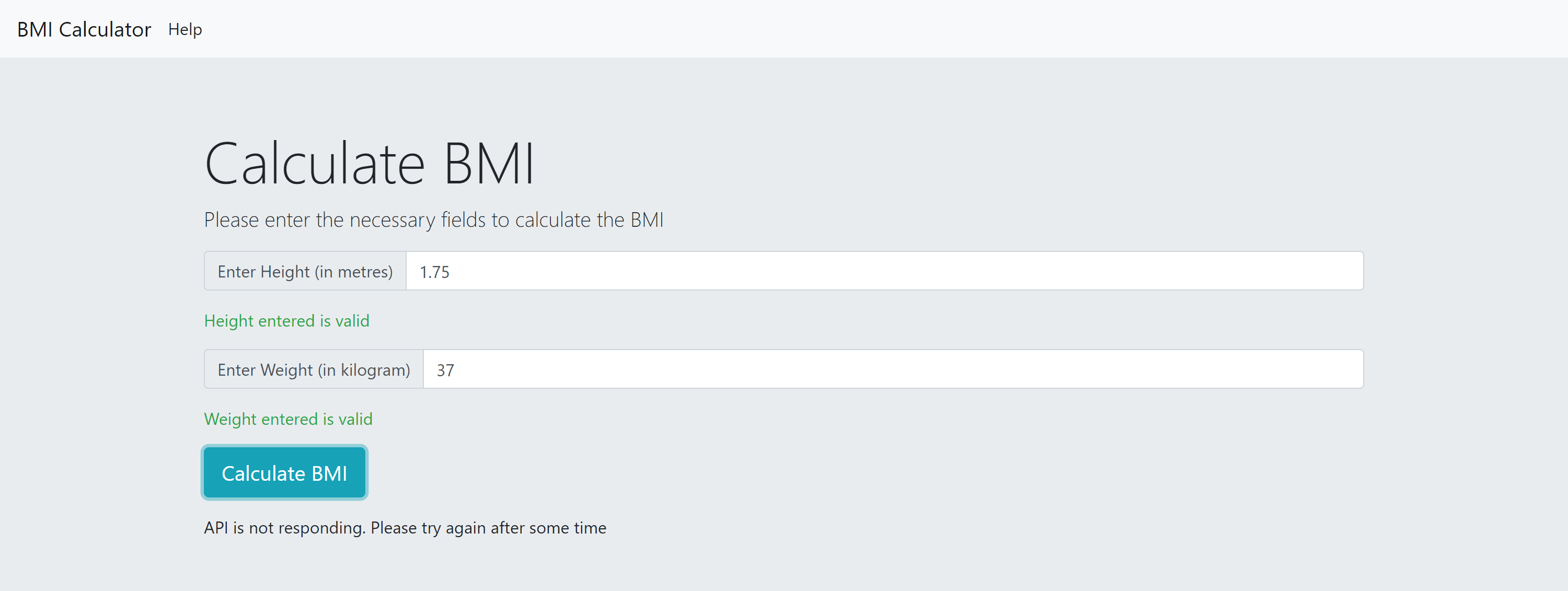


Figure 19: Test case when the internet connection is slow

In case the application is not connected to the internet, a timeout is set to 4 ms. Once the response is not received the frontend would show the message that API is not responding, please try again after ome time.

Test Case 8:Running the api with invalid inputs

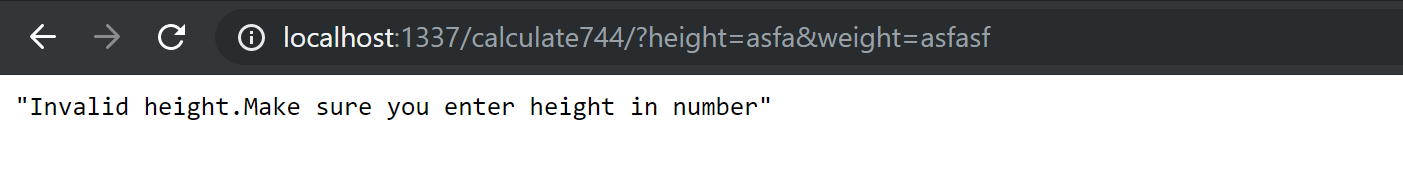


Figure 20: Test case when wrong input is entered in API

In case someone explicitly enters wrong height and weight (in non-digit format), then error would be thrown to the user that the height is invalid, enter the height in number.

Test Case 9: Testing for blank height and weight in API

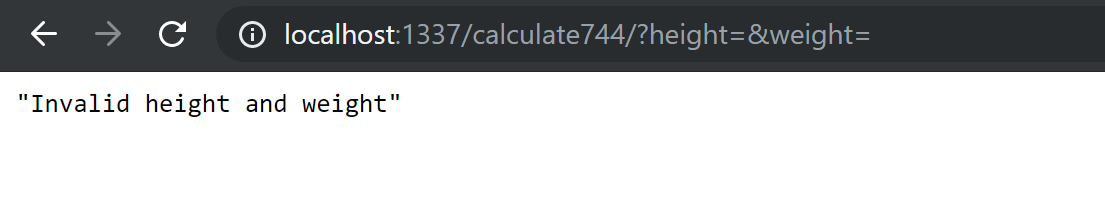


Figure 21: Test case for blank height and weight

In case height and weight entered is blank then the appropriate error message would be shown to the user that height and weight are invalid.

# Identification of source code

Bootstrap documentation : <https://getbootstrap.com/>

Jquery : <https://en.wikipedia.org/wiki/JQuery>

Sails documentation: <https://sailsjs.com/documentation/reference>

Code taken from the documentation have been added as a comment for their respective files.