

Efolio Task 9.1

Project Overview

Build with Gemini

Project shortcuts

Authentication

Functions

App Check

Firestore Database

Product categories

Build

Run

Analytics

All products

Related development tools

IDX

Checks

Blaze

Modify

Week7-Yuji

Functions

Dashboard Usage

Protect your Functions resources from abuse, such as billing fraud or phishing

Configure App Check

Looking for logs and health reporting? Visit the Google Cloud console for a highly customisable [logs view](#), [per-function usage details](#) and [error reporting](#)

Function	Trigger	Version	Requests (24 hours)	Min/max instances	Timeout
countBooks	Request	v2	9	0 / 100	1 m
getAllBooks	Request	v2	1	0 / 100	1 m

Items per page: 25 1-2 of 2

[Home](#) [Dashboard](#) [Program](#) [Add Book](#) [Get Book Count](#) [Weather Check](#) [Count Book API](#) [Get All Book API](#) [Login](#) [Registration](#)

Book Counter

[Get Book Count](#)  
Total count: 3

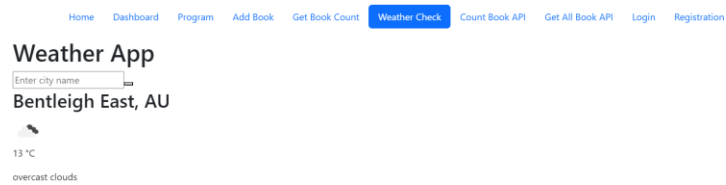
```
▼ AddBookView.vue    ▼ BookList.vue    {} package.json    ▼ WeatherView.vue    ▼ CountBookAPI.vue X    ▼ GetAllBookAPI.vue    ▼ LibraryRegistrationForm.vue    JS i

src > views > ▼ CountBookAPI.vue > {} script > [0] default

1  <template>
2    <h1>Book Counter</h1>
3    <button @click="getBookCountAPI">Get Book Count</button>
4    <p v-if="jsondata !== null">{{ jsondata }}</p>
5    <p v-if="error">{{ error }}</p>
6  </template>
7
8  <script>
9    import axios from 'axios';
10
11  export default {
12    data() {
13      return {
14        jsondata: null,
15        error: null
16      }
17    },
18    methods: {
19      async getBookCountAPI() {
20        try {
21          const response = await axios.get('https://countbooks-217homjiaq-uc.a.run.app')
22          this.jsondata = response.data
23          this.error = null
24        } catch (error) {
25          console.error('Error fetching book count:', error)
26          this.error = error
27          this.count = null
28        }
29      }
30    }
31  }
32
33  </script>
```

```
10  const {onRequest} = require("firebase-functions/v2/https");
11  // const logger = require("firebase-functions/logger");
12
13  const admin = require("firebase-admin");
14  const cors = require("cors")({origin: true});
15
16  admin.initializeApp();
17
18  exports.countBooks = onRequest((req, res) => {
19    cors(req, res, async () => {
20      try {
21        const booksCollection = admin.firestore().collection("books");
22        const snapshot = await booksCollection.get();
23        const count = snapshot.size;
24
25        res.status(200).send({count});
26      } catch (error) {
27        console.error("Error counting books: ", error);
28        res.status(500).send("Error counting books: " + error.message);
29      }
30    });
31  });
32
```

## Efolio Task 10.1



```
1  <template>
2    <div class="container">
3      <div class="header">
4        <h1>Weather App</h1>
5        <div class="search-bar">
6          <input type="text" v-model="city" placeholder="Enter city name" class="search-input">
7          <button @click="searchByCity"></button>
8        </div>
9        <!--The <main> tag in HTML is used to specify the main content of a document
10        More info about main, check https://www.w3schools.com/tags/tag\_main.asp-->
11        <main>
12          <!--If there are no data returned, then skip rendering the information-->
13          <div v-if="weatherData">
14            <!--Display the weather data attribute returned from API
15            Example of API data: https://openweathermap.org/current-->
16            <h2>
17              {{ weatherData.name }}, {{ weatherData.sys.country }}
18            </h2>
19            <div>
20              <!--The image source of of the weather icon will be coming from a function called iconUrl
21              which will be shared in script later-->
22              
23              <p>{{ temperature }} °C</p>
24            </div>
25            <!-- weather[0] means the current weather, the way we need to obtain data depends on how
26            the API JSON data looks-->
27            <span>{{ weatherData.weather[0].description }}</span>
28          </div>
29        </main>
30      </div>
31    </div>
32  </template>
33
```

```

35 <script>
36 // The info section in 10.1.1 provided detailed information about this package
37 import axios from "axios";
38
39 const apikey = "5c350c3e1bee12f10bf972c0676e503f";
40
41 export default {
42   name: "App",
43   data() {
44     return {
45       city: "",
46       weatherData: null,
47       hourlyForecast: [],
48       dailyForecast: [],
49     };
50   },
51   computed: {
52     temperature() {
53       return this.weatherData
54         ? Math.floor(this.weatherData.main.temp - 273)
55         : null;
56     },
57     iconUrl() {
58       return this.weatherData
59         ? `http://api.openweathermap.org/img/w/${this.weatherData.weather[0].icon}.png`
60         : null;
61     },
62   },
63   mounted() {
64     this.fetchCurrentLocationWeather();
65   },
66   methods: {
67     async fetchCurrentLocationWeather() {
68       if (navigator.geolocation) {
69         navigator.geolocation.getCurrentPosition(async (position) => {
70           const { latitude, longitude } = position.coords;
71           const url = `http://api.openweathermap.org/data/2.5/weather?lat=${latitude}&lon=${longitude}&appid=${apikey}`;
72           await this.fetchWeatherData(url);
73         });
74       }
75     },
76     async searchByCity() {
77       if (this.city) {
78         const url = `http://api.openweathermap.org/data/2.5/weather?q=${this.city}&appid=${apikey}`;
79         await this.fetchWeatherData(url);
80       } else {
81         console.error("Please enter a city name");
82       }
83     },
84     async fetchWeatherData(url) {
85       try {
86         const response = await axios.get(url);
87         this.weatherData = response.data;
88       } catch (error) {
89         console.error("Error fetching weather data:", error);
90       }
91     }
92   }
93 }
94 </script>

```

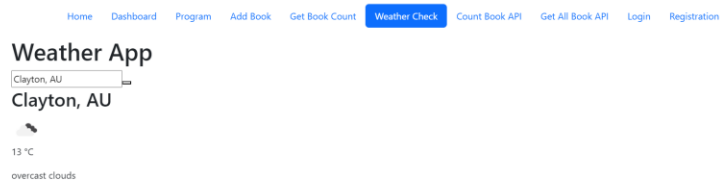
## Book Counter

[Get Book Count](#)  
 ( "count": 3 )

```

1  <template>
2      <h1>Book Counter</h1>
3      <button @click="getBookCountAPI">Get Book Count</button>
4      <p v-if="jsondata !== null">{{ jsondata }}</p>
5      <p v-if="error">{{ error }}</p>
6  </template>
7
8  <script>
9      import axios from 'axios';
10
11     export default{
12         data(){
13             return{
14                 jsondata: null,
15                 error: null
16             }
17         },
18         methods: {
19             async getBookCountAPI(){
20                 try {
21                     const response = await axios.get('https://countbooks-217homjiaq-uc.a.run.app')
22                     this.jsondata = response.data
23                     this.error = null
24                 }
25                 catch (error){
26                     console.error('Error fetching book count:', error)
27                     this.error = error
28                     this.count = null
29                 }
30             }
31         }
32     }
33 </script>
    
```

## Efolio Task 10.2



```
1  <template>
2    <div class="container">
3      <div class="header">
4        <h1>Weather App</h1>
5        <div class="search-bar">
6          <input type="text" v-model="city" placeholder="Enter city name" class="search-input">
7          <button @click="searchByCity"></button>
8        </div>
9        <!--The <main> tag in HTML is used to specify the main content of a document
10        More info about main, check https://www.w3schools.com/tags/tag\_main.asp-->
11        <main>
12          <!--If there are no data returned, then skip rendering the information-->
13          <div v-if="weatherData">
14            <!--Display the weather data attribute returned from API
15            Example of API data: https://openweathermap.org/current-->
16            <h2>
17              {{ weatherData.name }}, {{ weatherData.sys.country }}
18            </h2>
19            <div>
20              <!--The image source of of the weather icon will be coming from a function called iconUrl
21              which will be shared in script later-->
22              
23              <p>{{ temperature }} °C</p>
24            </div>
25            <!-- weather[0] means the current weather, the way we need to obtain data depends on how
26            the API JSON data looks-->
27            <span>{{ weatherData.weather[0].description }}</span>
28          </div>
29        </main>
30      </div>
31    </div>
32  </template>
33
```

```

35 <script>
36 // The info section in 10.1.1 provided detailed information about this package
37 import axios from "axios";
38
39 const apikey = "5c350c3e1bee12f10bf972c0676e503f";
40
41 export default {
42   name: "App",
43   data() {
44     return {
45       city: "",
46       weatherData: null,
47       hourlyForecast: [],
48       dailyForecast: [],
49     };
50   },
51   computed: {
52     temperature() {
53       return this.weatherData
54         ? Math.floor(this.weatherData.main.temp - 273)
55         : null;
56     },
57     iconUrl() {
58       return this.weatherData
59         ? `http://api.openweathermap.org/img/w/${this.weatherData.weather[0].icon}.png`
60         : null;
61     },
62   },
63   mounted() {
64     this.fetchCurrentLocationWeather();
65   },
66   methods: {
67     async fetchCurrentLocationWeather() {
68       if (navigator.geolocation) {
69         navigator.geolocation.getCurrentPosition(async (position) => {
70           const { latitude, longitude } = position.coords;
71           const url = `http://api.openweathermap.org/data/2.5/weather?lat=${latitude}&lon=${longitude}&appid=${apikey}`;
72           await this.fetchWeatherData(url);
73         });
74       }
75     },
76     async searchByCity() {
77       if (this.city) {
78         const url = `http://api.openweathermap.org/data/2.5/weather?q=${this.city}&appid=${apikey}`;
79         await this.fetchWeatherData(url);
80       } else {
81         console.error("Please enter a city name");
82       }
83     },
84     async fetchWeatherData(url) {
85       try {
86         const response = await axios.get(url);
87         this.weatherData = response.data;
88       } catch (error) {
89         console.error("Error fetching weather data:", error);
90       }
91     }
92   }
93 }
94 </script>

```

## All Books

Get All Books

```
{ "books": [ { "id": "GwNqTnWljdvOAfD2m2G", "name": "Yujie's Book", "isbn": 1234 }, { "id": "JImQnuB2PDobcyUS1nF", "name": "Harry Potter", "isbn": 2000 }, { "id": "p52o8mHGruYqNPlsWXWJ", "isbn": 999, "name": "Book 2" } ] }
```

```
1  <template>
2    <h1>All Books</h1>
3    <button @click="getAllBookAPI">Get All Books</button>
4    <p v-if="jsondata !== null">{{ jsondata }}</p>
5    <p v-if="error">{{ error }}</p>
6  </template>
7
8  <script>
9    import axios from 'axios';
10
11  export default{
12    data(){
13      return{
14        jsondata: null,
15        error: null
16      }
17    },
18    methods: {
19      async getAllBookAPI(){
20        try {
21          const response = await axios.get('https://getallbooks-217homjiaq-uc.a.run.app')
22          this.jsondata = response.data
23          this.error = null
24        }
25        catch (error){
26          console.error('Error fetching book count:', error)
27          this.error = error
28          this.count = null
29        }
30      }
31    }
32  }
33  </script>
```



```
const {onRequest} = require("firebase-functions/v2/https");
// const logger = require("firebase-functions/logger");

const admin = require("firebase-admin");
const cors = require("cors")({origin: true});

admin.initializeApp();

exports.countBooks = onRequest((req, res) => {
  cors(req, res, async () => {
    try {
      const booksCollection = admin.firestore().collection("books");
      const snapshot = await booksCollection.get();
      const count = snapshot.size;

      res.status(200).send({count});
    } catch (error) {
      console.error("Error counting books: ", error);
      res.status(500).send("Error counting books: " + error.message);
    }
  });
});

exports.getAllBooks = onRequest((req, res) => {
  cors(req, res, async () => {
    try {
      const booksCollection = admin.firestore().collection("books");
      const snapshot = await booksCollection.get();
      const books = [];
      snapshot.forEach((doc) => {
        books.push({id: doc.id, ...doc.data()});
      });

      // Send the array of books as a response
      res.status(200).send({books});
    } catch (error) {
      console.error("Error fetching books: ", error);
      res.status(500).send("Error fetching books: " + error.message);
    }
  });
});
```