Case Study: Tracking and Optimizing Business Growth with Firebase, Google Tag Manager (GTM), and Google Analytics 4 (GA4)

1. Introduction

This case study demonstrates how Firebase Hosting, Google Tag Manager (GTM), and Google Analytics 4 (GA4) were implemented on a website for real-time tracking, analytics, and business insights. This project outlines the end-to-end setup, covering everything from deploying the website for free using Firebase Hosting to implementing custom events in GTM and leveraging the power of GA4 for business growth.

2. Firebase Hosting Overview

Firebase provides a free and reliable platform to host web applications, which is essential for small-scale and medium-sized businesses looking to create an online presence without incurring heavy costs. For this project, Firebase Hosting was used to deploy the website.

Key Features of Firebase Hosting:

- **Free tier**: Firebase offers a free-tier for hosting static websites, which includes HTTPS, a custom domain, and basic CDN features.
- **Real-time database integration**: It seamlessly connects with Firebase's real-time database.
- **Scalability**: Firebase can handle scale effortlessly, offering enterprise-level performance for all hosting needs.

3. Google Tag Manager (GTM) Implementation

Google Tag Manager was implemented to manage the tracking and analytics tags for this project. GTM enables the deployment of custom tags, events, and variables without the need to modify the website's code repeatedly.

GTM Implementation Steps:

- **Step 1**: Install the Google Tag Manager snippet within the `<head>` and `<body>` sections of the webpage.
- **Step 2**: Set up custom events and variables to track user interactions, such as form submissions.

- **Step 3**: Use custom tags to fire events like `form_submit_click` and pass hashed data into the dataLayer, which is later sent to GA4 for reporting.

Custom Events Setup:

- The `form_submit_click` event was implemented to track form submissions, passing data like the user's hashed name and email into the dataLayer.
- GTM listens for these events and triggers corresponding GA4 tags, enabling seamless integration of user data into GA4.

4. GA4 Integration and Custom Events

Google Analytics 4 (GA4) is a robust platform that supports the collection and analysis of real-time data from websites and applications. GA4 offers a significant upgrade over Universal Analytics, with its focus on user-centric data models, cross-platform tracking, and machine learning capabilities.

Key GA4 Features:

- **Event-based data model**: Unlike traditional analytics, GA4 is event-driven, meaning every interaction (like a button click, form submission, or scroll) is captured as an event.
- **Cross-platform tracking**: It can track users across multiple platforms (web and app), giving businesses a unified view of the customer journey.
- **Advanced reporting and insights**: GA4 leverages machine learning to provide predictive insights, funnel tracking, and path analysis.

Custom Event Implementation in GA4:

In this project, custom events like `form_submit_click` were created in GA4 via GTM to track form submissions. The hashed data from the form (user's name and email) is passed into the dataLayer and pushed to GA4 to ensure compliance with privacy regulations.

1. **Custom Events Example**:

```
window.dataLayer.push({
    'event': 'form_submit_click',
    'name_hashed': hashedName, // Hashed name
    'email_hashed': hashedEmail, // Hashed email
    'message': message // Plain message
});
```

Similarly the navigation top menu is tracked as well as all the CTAs present on the webpage using the GTM (Google Tag Manager)

GA4 Tag Setup: In GTM, create a tag for GA4 (Event type), where you pass the hashed user details and other information through the dataLayer into GA4's reporting interface. This allows you to track custom events that are essential for business insights and optimization.

5. Google Apps Script: Using Google Sheets as a Database

Google Sheets was used as a lightweight database for form submissions, with Google Apps Script handling the interaction between the form and the Google Sheets backend.

How Google Apps Script Was Used:

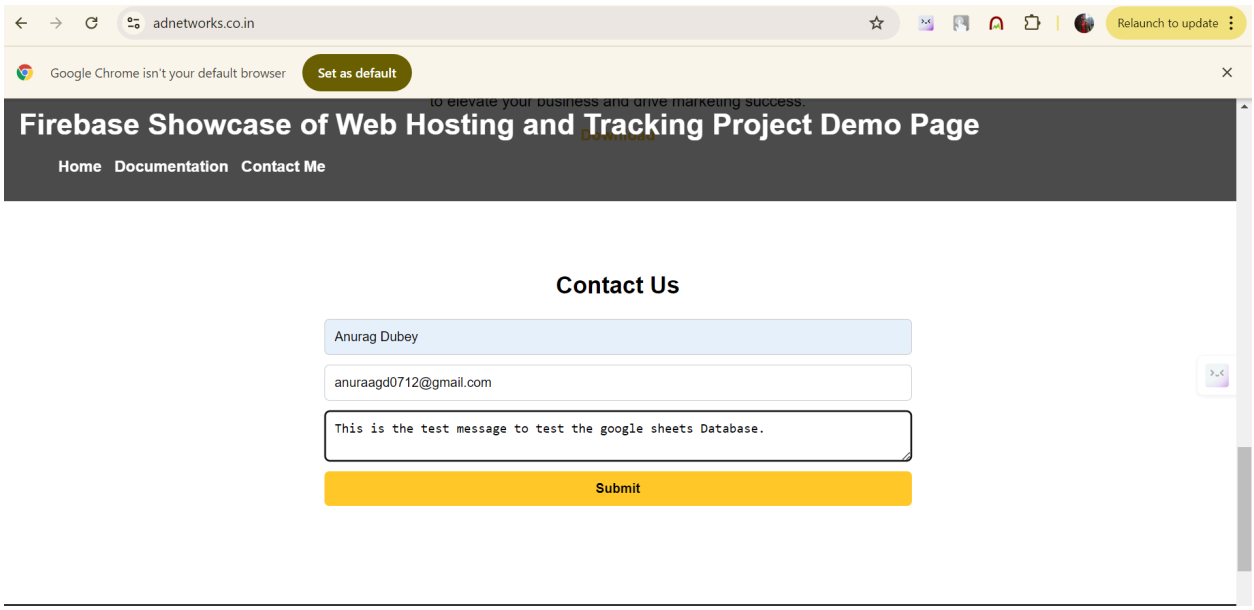
- **Form Data Submission**: When a user submits the form, the details are sent to a Google Apps Script function that writes the data into a Google Sheet.
- **Real-time Updates**: Google Sheets stores this data in real time, making it an efficient and cost-free database for small-scale projects.
- **Reporting & Integration**: The data stored in the Google Sheet can be analyzed independently or used in conjunction with GA4 for more comprehensive reporting.

Code Example:

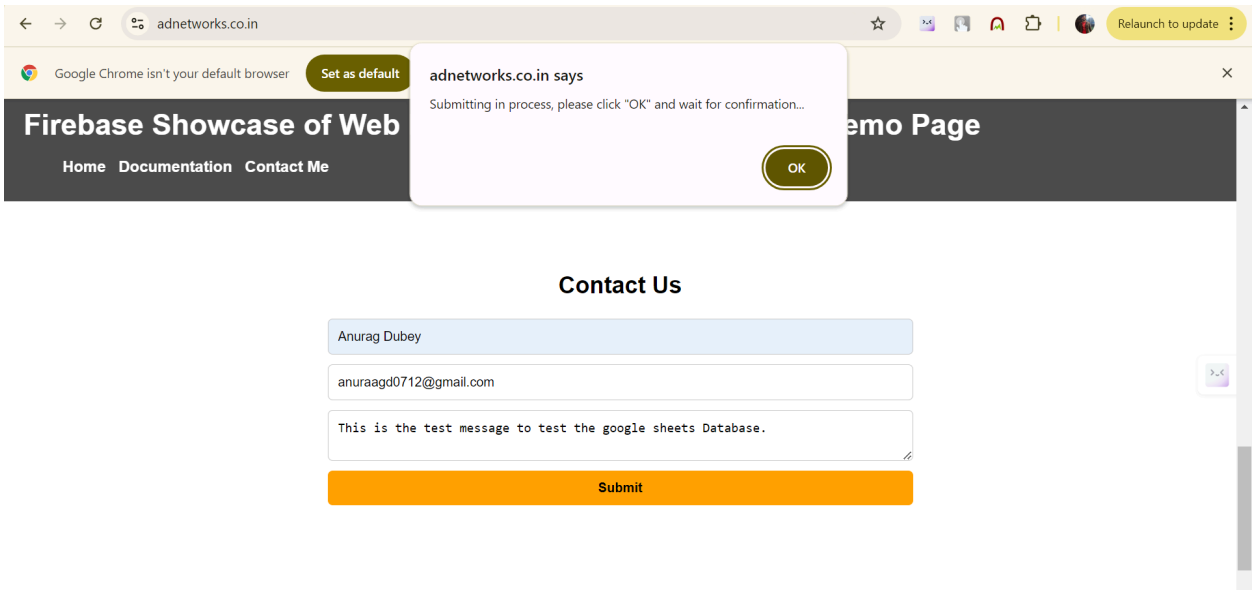
The following Apps Script is used to handle form data submission:

```
function doPost(e) {
  var sheet =
SpreadsheetApp.openById('YOUR_SPREADSHEET_ID').getSheetByName('Sheet1');
  var rowData = [
    e.parameter.name,
    e.parameter.email,
    e.parameter.message,
    new Date()
  ];
  sheet.appendRow(rowData);
  return ContentService.createTextOutput("Success");
```

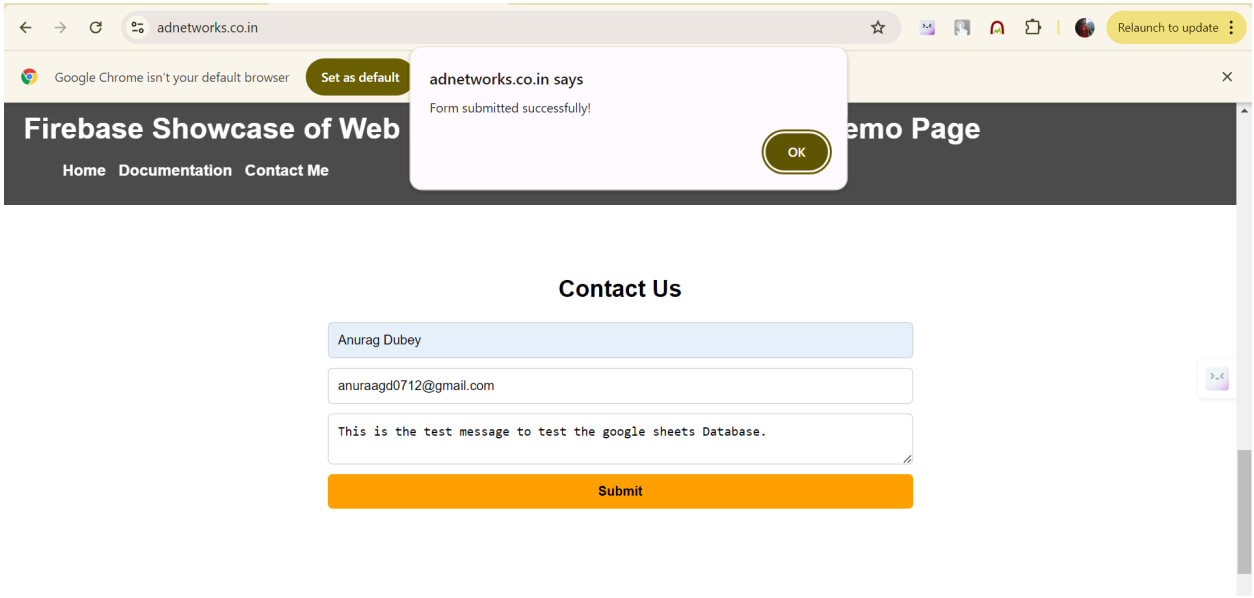
}



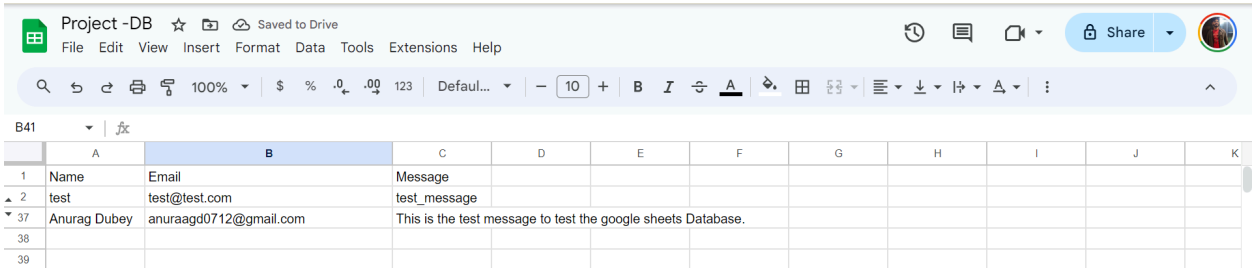
Fill the form and Hit submit.



The alert for submit click pops up follow the instructions.

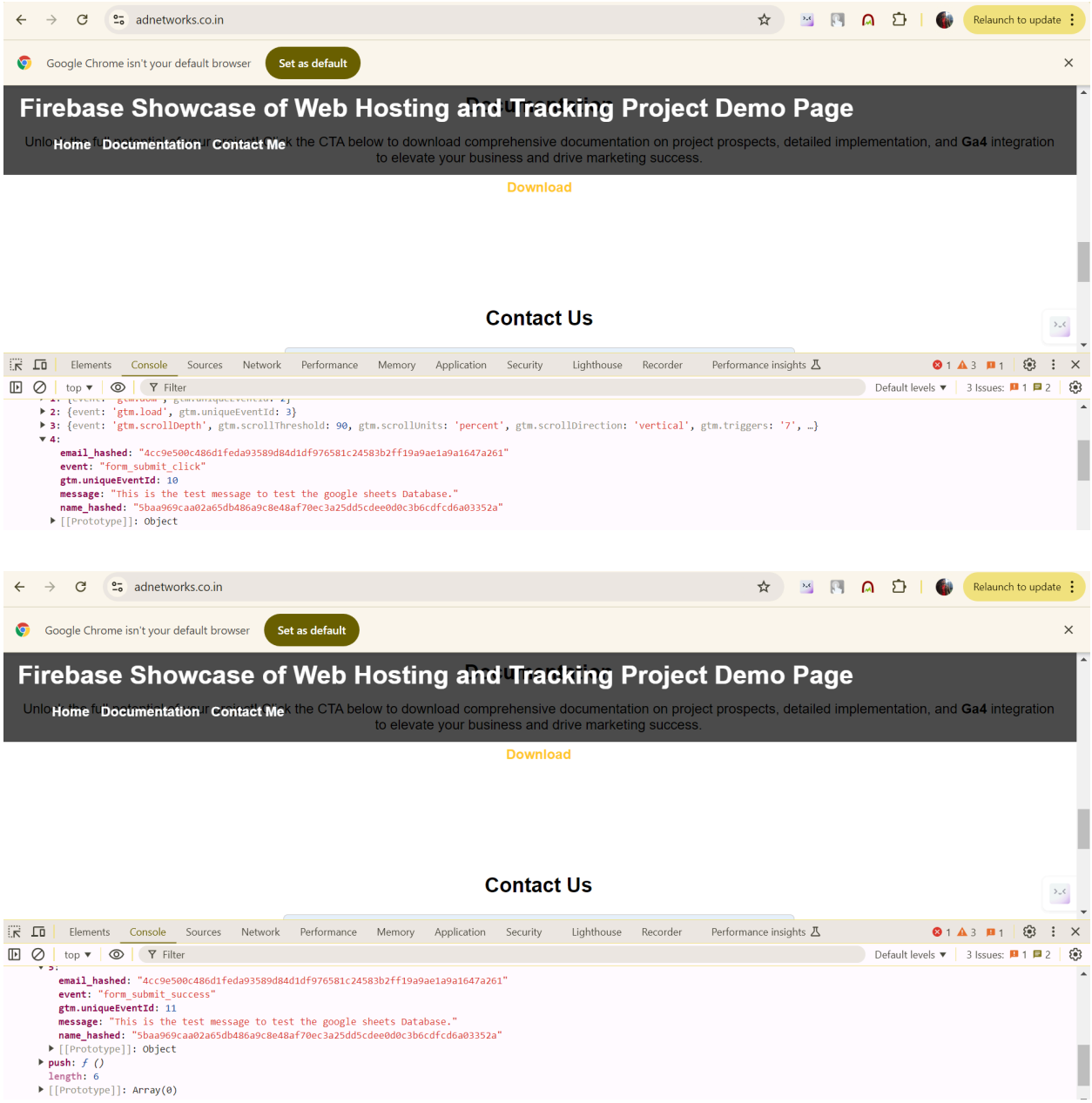


Form Submitted successfully messages pops up.



The google app script fetching the data into the Document Successfully.

Data layer form_submit_click with the hashed form content.



Datalayer form_submit_success fires after the validation from google sheets api comes and the hashed content is pushed into the data layer

Now the hashed content can be sent to ga4 servers to process the report further. After which we can use Vlookup on the local database to join in the full data and add additional fields in the form as per the business case.

6. Data Privacy: Hashing the Name and Email

To ensure data privacy and security, sensitive user information such as the name and email address were hashed using SHA-256 before being sent to the dataLayer and GA4.

Hashing Implementation:

- **Why Hashing?**: By hashing sensitive data like email addresses, we ensure compliance with data protection regulations such as GDPR. The hashed data is irreversible, which protects users' personal information.
- **SHA-256 Hashing**: The JavaScript `crypto.subtle.digest` API was used to hash the data. This ensures that even if the hashed data is passed through the dataLayer to GA4, the original information cannot be reconstructed.

Example Code:

```
async function hashData(data) {

    const encoder = new TextEncoder();

    const dataBuffer = encoder.encode(data);

    const hashBuffer = await crypto.subtle.digest('SHA-256', dataBuffer);

    const hashArray = Array.from(new Uint8Array(hashBuffer));

    return hashArray.map(b => b.toString(16).padStart(2, '0')).join('');

}
```

This hash is then pushed into the dataLayer and GA4 via custom events.

7. Combining GA4 and Google Sheets for Advanced Reporting

The combination of GA4 and Google Sheets provides a powerful reporting mechanism that can be used to track, analyze, and optimize business performance.

- **GA4 Reporting**: Custom reports based on user interaction events (like form submissions) can be built in GA4 to track key metrics like user engagement, conversion rates, and user journeys.
- **Google Sheets Analysis**: Data collected in Google Sheets can be cross-referenced with GA4 data to create a more comprehensive view of user behavior and business performance. This data can be used to monitor trends and make data-driven decisions to optimize marketing efforts.

8. Conclusion: Streamlining Business Growth

This project highlights the immense potential of integrating Firebase Hosting, Google Tag Manager, and Google Analytics 4 to streamline business growth. With real-time data tracking, privacy-centric implementations (hashed data), and comprehensive reporting tools, businesses can make informed decisions, improve marketing strategies, and optimize the user experience.

By leveraging Google Sheets as a lightweight backend and combining it with GA4 reports, this project provides a cost-effective, scalable solution for small and medium-sized businesses looking to improve their digital strategy and increase overall growth.

Personalized User Greeting Script for Returning Visitors

This JavaScript code refer below for sample code. provides personalized greeting messages to users based on the number of times they have visited the website. By utilizing the **Google Analytics (_ga) cookie** and **localStorage**, the script tracks user sessions and dynamically generates personalized messages to enhance user engagement. Below is a step-by-step breakdown of the functionality:

1. **Retrieving Google Analytics Cookie (`_ga`)**:
    - The script searches for the `_ga` cookie in the browser's cookie storage. The `_ga` cookie is often used by Google Analytics to track individual users.
    - If the cookie is found, its value can be displayed in the greeting message (though this is optional).
2. **Session Count with `localStorage`**:

    - The script checks the `localStorage` for a stored session count under the key `visitCount`.
    - Every time a user visits the site, the session count is incremented and stored back into `localStorage`. This tracks how many times the user has visited the site across different sessions.
3. **Unique Greeting Messages**:
    - The script contains a predefined set of greeting messages. These messages are designed to engage returning users, expressing appreciation for their loyalty.
    - The greeting is personalized based on the number of visits. It cycles through different messages depending on the session count.
4. **Displaying the Greeting**:
    - On each page load, the script checks if it is the user's first visit or a returning visit.
        - **First-time visitors** receive a special welcome message encouraging them to connect on LinkedIn.
        - **Returning visitors** receive one of the predefined messages, along with the number of times they have visited the site. The message changes dynamically with each visit.
5. **User-Friendly Experience**:
    - The personalized greeting is displayed via a pop-up alert when the page loads. This enhances the user experience by acknowledging their return and offering a sense of engagement.

Use Cases for Business Growth:

- **Engagement**: The script is designed to engage users and foster loyalty by acknowledging returning visitors, which can help improve user retention.
- **Tracking & Analytics**: By optionally leveraging the `_ga` cookie, businesses can combine personalized user interactions with Google Analytics data for a deeper understanding of user behavior.
- **User Interaction**: Encouraging users to return through personalized messages can lead to improved engagement metrics and business growth through sustained user interaction.

Dummy code for personalized greeting when user visits website

```
<script>
  // Step 1: Function to retrieve a dummy cookie value (no real data is being used)
  function getCookieValue() {
    var cookieString = document.cookie;
    var cookies = cookieString.split('; ');
    for (var i = 0; i < cookies.length; i++) {
      var cookie = cookies[i].split('=');
      var cookieName = cookie[0];
```

```
    var cookieValue = cookie[1];

    // Looking for a dummy cookie (starting with "_dummy_cookie")
    if (cookieName.indexOf('_dummy_cookie') === 0) {
      return cookieValue; // Return the dummy cookie value
    }
  }
  return null; // Return null if no cookie is found
}

// Step 2: Function to track user visits without storing personal data
function trackVisitCount() {
  // Get the current visit count from localStorage (this is anonymous)
  var visitCount = parseInt(localStorage.getItem('visitCount')) || 0;

  // Increment the visit count on each visit
  visitCount += 1;

  // Save the new visit count to localStorage (no personal data is used)
  localStorage.setItem('visitCount', visitCount);

  return visitCount;
}

// Step 3: Function to generate anonymous greeting messages
function generateGreeting(visitCount) {
  var greetings = [
    'Welcome to our demo site!',
    'Thanks for visiting again!',
    'We appreciate your return!',
    'You are a valued visitor!',
    'Good to see you again!',
    'Thanks for your loyalty!',
    'Another visit? That's awesome!',
    'Your presence is appreciated!',
    'We love repeat visitors!',
    'Glad to see you back!'
  ];

  // Pick a greeting based on visit count (cycling through messages)
  var messageIndex = visitCount % greetings.length;
  return greetings[messageIndex] + ' This is your ' + visitCount + 'th visit.';
}
```

```
  // Step 4: Function to display the generated greeting message
  function showGreeting() {
    var cookieValue = getCookieValue(); // Get the dummy cookie value
    var visitCount = trackVisitCount(); // Track how many times the user has visited

    // First visit: Show a custom message
    if (visitCount === 1) {
      alert("Welcome to our demo site for the first time!");
    } else {
      // For subsequent visits, show a dynamically generated greeting
      var greetingMessage = generateGreeting(visitCount);

      // Optionally, display the dummy cookie value
      // if (cookieValue) {
      //   greetingMessage += ' Your cookie ID is: ' + cookieValue + '.';
      // }

      // Display the greeting message as an alert
      alert(greetingMessage);
    }
  }

  // Step 5: Trigger the greeting when the page loads
  window.onload = function() {
    showGreeting();
  };
</script>
```