Politecnico di Torino

Master's degree program in computer engineering

Master Thesis

Modern technological and analytical approaches in the monitoring of errors generated by the frontend layers of web applications

Relatori:

MAURIZIO MORISIO

Candidati:

MOHADESEH ALIPOUR

Academic year 2020 –2021

Abstract

Over the last few years, Web applications are continually evolving, and following these changes, the number of errors generated is expanding in frontend layers as well. As a result, appropriate web application implementation is one of the most critical considerations for developers and administrators. Indeed, it has had a detrimental impact on the company's reputation and resulted dissatisfaction among customers and potentially a loss of them.

Therefore, the purpose of the study was to investigate some analytical approaches to track user activity and monitor the errors occurring in frontend layers of web applications. Actually, it aimed to examine these techniques in order to determine the health of the website and improve its efficiency.

In this paper, this topic is addressed by developing a website named “worldbusiness”. Thus, four approaches are used that Google tag Manager and Google Analytics are dedicated to tracking user behaviour, while Sentry and Rollbar are consecrated to monitor the errors; finally, to show the result, installed mentioned tools by inserting a snippet of JavaScript into web page.

The research method used in the study was comparative research and is assumed that based on previous studies, using analytical approaches we can monitor the errors generated by the frontend layers of web applications.

Due to the fact that this study concentrated on monitoring tools, there were several limits to the study. Specifically, because this research was conducted on a single website created by the author, it was unable to reach a wide range of users. As a consequence, the advantages and disadvantages of these techniques were investigated using a small statistical sample and previous research, which among of them, by GTM, all data was collected and user transaction, the number of times the pages has been visited, duration of time being on the website was recorded and displayed on GTM as a platform. Besides, by the sentry, the errors encountered by each user were recorded, and also the user’s URL and type of error.

Keywords: Web applications, Web monitoring, tracking error, analytical approaches, Web analytics

Abbreviations

JS	JavaScript
GA	Google Analytics
GTM	Google Tag Manager
URL	Uniform Resource Locator
DSN	Data Source Name
SDK	Software development kit
FCP	First Contentful Paint
LCP	Largest Contentful Paint
FID	First Input Delay
CLS	Cumulative Layout Shift

Acknowledgment

I would like to express my special thanks to my professor,
MAURIZIO MORISIO who provided me with wonderful
advice through difficult periods, as well as
OVERACE GROUP company for giving me an opportunity
to work on this project.
In addition, I would like to thank my family and friends
for their support that without their help,
I could not achieve this success.

Table of Contents

Table of Figures

1. Introduction

Over the last few years, web applications are growing very fast and according to the estimates (Croll, Power, 2009) There has been 1.6 billion websites. Following this rapid expansion, administrators have encountered a huge number of errors in the frontend layers and as a result, it has caused dissatisfaction among customers and reduced the company's reputation (Filipe, Araujo, 2016).

Therefore, in order to ensure the proper performance of the websites, we can take the advantage of frontend web monitoring to reduce the harmful repercussions of programming errors.

Indeed, frontend web monitoring is the act of evaluating and tracking a website's uptime status and performance, tracking the errors, resolving them, and maintaining the health of online applications to guarantee that it is operating at peak efficiency (Croll, Power, 2009). It's done to improve the user experience by eliminating any errors that might affect the program.

According to the study (Filipe, Araujo, 2016), 16% of the top 1000 websites have errors, and as well as according to (Pertet, & Narasimhan, 2005), web application errors can have irreparable consequences for companies, with the following examples highlighting some of the most severe. First, it can cause losing or dissatisfying the users. However, user feedback is one of the most important key factors to determine the level of the website. Furthermore, it can harm the reputation of the company and even potentially have an effect on its stock price.

Considering these consequences and in order to avoid them, we decided to utilize certain methods in this paper to monitor user behavior on our produced website called, “WorldBusiness”, as well as discover errors in the frontend layers.

Based on the studies that have been done (Hootsuite, 2019; Croll, & Power, 2009), using tools we can detect errors in advance and in fact improve the performance of the website and the level of customer satisfaction.

We collected metrics and frontend layer’s data, such as user interactions, button clicks, purchases and as well as JavaScript problems and in this way, we could notify system administrators.

In order to address these problems and demonstrate these analytical approaches, a website has been developed in React and JavaScript and it aimed to highlight a few useful features of GTM Analytics and monitoring approaches to show how to track the events and monitor the errors then compare and consider the advantage and limitation of these features as mentioned above.

Based on literature and previous works, there are a variety of techniques to monitoring errors. Four strategies for tracking user behaviour on websites and monitoring errors to assess the health of the created website were investigated in this research.

The research method used in this paper is comparative research. Accordingly, to compare and analyse these techniques, developed a website using JavaScript and React (is a free and open-source front-end JavaScript library), then installed these tools by inserting a snippet of JavaScript into web page that is covered in detail in the following chapters.

The remainder of the paper is structured as follows. Chapter 2 provides an overview of previous research findings. Chapter 3 describes the research method used in this paper and discuss about these tools. Chapter 4 describes the settings and examines possible client-side monitoring solutions. In Chapter 5 show the conclusion of the analysis, the strengths, and limitations of these approaches.

2. Literature Review

This section covers a summary of research conducted within previous years on search terms web applications, google analytics, tracking and monitoring errors in frontend.
The aim is to analyze all these tools and detect the errors of the website and to learn more about how tracking and error monitoring impacts the website's efficiency.
Therefore, this chapter is divided into two parts. Section 2.1 describes literature review about the Importance of web analytics; Section 2.2 cover the Importance of web monitoring.

2.1 Importance of web analytics

The process of tracking activity and behavior on a website, such as how many people visit, how long they stay, how many pages they view, and which pages they visit, is known as web analytics; in which, essentially provide insights and data that may be utilized to improve the user experience for website visitors (Farney , 2016).

(Song, Ward, Choi, Nikoo, Frank, Shams, & Krausz, 2018) Conducted research on WalkAlong.ca, a youth-oriented mental health web-portal, including 3076 users and from Nov 13, 2013-Nov 13, 2014, to assess Web-based mental health treatments by using google analytics as a tool, in which among the Google Analytic factors, entire website engagement, such as pages viewed each session, use rate of certain services, and user access method (desktop, mobile phone or tablet) and location were examined which 67% of users were from Canada.

According to the literature review by (Yeager , 2017) on both print and electronic resources at Elon University's Belk Library, assessed statistics of electronic usage, by using EZproxy and google analytics to analyze the data on the website and to compare vendor information to check how the library's electronic resources are being utilized.

In the study by (Jansen, Jung, & Salminen, 2020) where, conducted on the data of 86 websites in 26 countries with different industries, they compared data of these websites in one year by two tools, google analytics and SimilarWeb. There was significant difference between two approaches for total visits, unique visitors, and bounce rate.

(O'Brien, Young, Arlitsch, & Benedict, 2018) investigated the privacy of the academic library of 279 websites by using google analytics and google tag manager and security protection tools.

(Conrad, 2015) surveyed tracking DSpace metadata using google analytics by importing data manually and collected data by google tag manager automatically.

(Azim, & Hasan, 2018) performed a study on usage of web analytics features among Indian Libraries. They used the online social networking platform 'LIS Links' in their research to conduct an online survey using Google forms, which the final analysis included with 100 legitimate replies. They also discussed about both paid and free web analytics tools between Indian Libraries.

(Kirk, Morgan, Tonkin, McDonald, & Skirton, 2012) conducted research on education website of genetics generated for nurses to show and assess the feedback of all the visitors in the world. They utilized google analytics in their investigation; moreover, they have collected the data of 123 countries with 33,536 visitors, during years of 2009-2011, which among countries, the majority of the users have belonged to the United Kingdom, United States, Canada, and the Netherlands.

(Bai, Law, & Wen, 2008) empirically investigated a theoretical model of the influence of website quality on consumer satisfaction and purchase intent in Chinese online visitors' statistics and sample was taken in three hotels.

(Jeong, Oh, & Gregoire, 2003) surveyed research on 16 hotel Web sites to the understanding of website quality and its implications in the hospitality sector, and data gathered through an electronic survey.

(Macbeth, 2016) conducted a survey on tracking services and also the security and privacy implications of this tracking, and online German banks; by examining where third parties appear on internet banking pages, what is loaded, and who these third parties are.

2.2 Importance of web monitoring

Web monitoring tools are used to discover problems with the site before people report them, that obviously increases total uptime while decreasing user disappointment (King, 2008).

According to reports (Business Wire Retrieved, 2020, Jul 14), annually, organizations invest roughly $4.6 million on problem management in which the reason could be because of the number of occurrences, inefficient procedures and processes, and a complete absence of usable data to help determine the root cause. In the following, 91% of companies, due to performance and availability problems lost revenue and the average monthly loss is anticipated to be $634,000.

And according to other reports conducted by (TeaLeaf Technology Inc. by Harris Interactive, 2005), 89% of all customers during online transaction have encountered difficulties.
Another report shows during shopping online in 2002, 72.5% of top forty websites revealed errors, including, Blank pages, erroneous data displayed on Web sites, wrong goods displayed in response to a user request, and difficulty to make a transaction, within the first fifteen minutes of testing (A TeaLeaf Technology Inc. white paper, 2003).

Therefore, (Pertet, & Narasimhan, 2005) on their research, by working on data gathered in websites posted on technology websites such as CNET.com and eweek.com. examined Causes of Failure in Web Applications. According to this research, 80% of failures in websites is composed of software failures and human error.

Based on the research conducted by (Filipe, Araujo, 2016), on Client-Side Monitoring Techniques for Web Sites, they tested with and counted the web page errors of 3000 websites by three client-side monitoring approaches including: stand-alone applications, browser extensions and JavaScript snippets with analytic tools to show their limitations.
Significantly, they found that 16% of the top 1,000 sites have errors, so meaningfully, less prominent sites contain more errors. On another study in 2019 they investigated Client-Side Monitoring of HTTP Clusters Using Machine Learning Techniques that process data collected and uploaded by web clients.

(Trinh, Vu, & Le, 2019) described a crowdsourcing-based solution to website monitoring that use browser extensions as checkpoints.

(Kinnunen, 2020) In his study compared free-to-use tools and evaluated the performance (loading and rendering speed) of an existing WordPress site using these tools, GTmetrix, PageSpeed Insights and WebPageTest.

(Li, & Gorton, 2010) on their paper, by examining the web logs and using the tool named REBA, detected the user-visible errors.

(Ocariza , Pattabiraman, & Zorn, 2011) conducted empirical research on fifty web applications from the Alexa Top 100 to determine the core causes of error messages displayed by JavaScript code in web applications.

(Fenstermacher, & Ginsburg , 2003) offered an efficient client-side monitoring system that allows for adaptive data collecting.

(Kiciman, & Livshits, 2007) analyzed the behavior of over 90 Web 2.0 applications using AjaxScope, a dynamic instrumentation platform that enables cross-user monitoring and just-in-time control of web application behavior on end-user desktops, and remotely monitoring and debugging the client-side behavior of Web 2.0 applications.

According to the literature review by (Mehta, & Bharadwaj, 2015) the role of sentry and guard processes were assessed to enhance and integrate the project outputs of software teams.

The results also revealed that the conduct of sentry actions will improve a team's knowledge integration significantly.

(Chyrun, Burov, Rusyn, Pohreliuk, Oleshek, Gozhyj, & Bobyk, 2019) analyzed three different software products (Subvertion, Content Downloader, and SiteLock) with the goal of developing a smart automated monitoring system that compares its contents with previously saved editions and alerts the user if there are any discrepancies. As a result, the user will no longer have to manually monitor the defined Internet resources.

The essential ideas of Web Content Monitoring were outlined in (Chyrun, Gozhyj, Yevseyeva, Dosyn, Tyhonov, & Zakharchuk, 2019). Specialized programs or parsers, news subscriptions, search engine use, and human data collecting were among the methods used. They discovered that using specialized programs to monitor material is the most effective technique (parsers).

On the other hand, (Filipe, 2020) conducted comprehensive research on Client-Side Monitoring of Distributed Systems, which was divided into two main branches, white-box, and black-box, by employing three approaches, a stand-alone approach, a browser extension, and a JavaScript snippet, and using Google Analytics to display the errors.

3. Research Method

This chapter presents research methodology employed in monitoring of errors generated by the frontend layers of web applications. Therefore, is separated into three sub-sections. The first portion concentrated on the developed website, while the second evaluated the analytical techniques, then the third one examined monitoring tools.

3.1 Description of the developed website

To investigate the analytical techniques on the frontend layers of the application, a website is developed in React and JavaScript. As shown in Figure3.1, Figure3.2 and Figure3.3, the website includes a home page, services, marketing and contact pages respectively. On the home page, users may explore the entire website by scrolling, and watch a business video, as well as subscribe to the website by entering their email address, and then click on buttons to access to the other pages. In addition, users may shop on the services and marketing pages. By choosing a good, the desired product is placed in the product basket, where it may check the list of selected goods before ordering or canceling the final transaction. (Figure3.4) Finally, contact us page where new visitors can ask their questions and send messages to an individual at organization.

The research method utilized in this work was a comparative method in which the user actions were examined, and probable client-side errors were revealed by adding JavaScript code snippets related with each analytical tool on the website, which was addressed in depth in Chapter 4.

Finally, all these tools were compared in depth to determine their benefits and drawbacks.

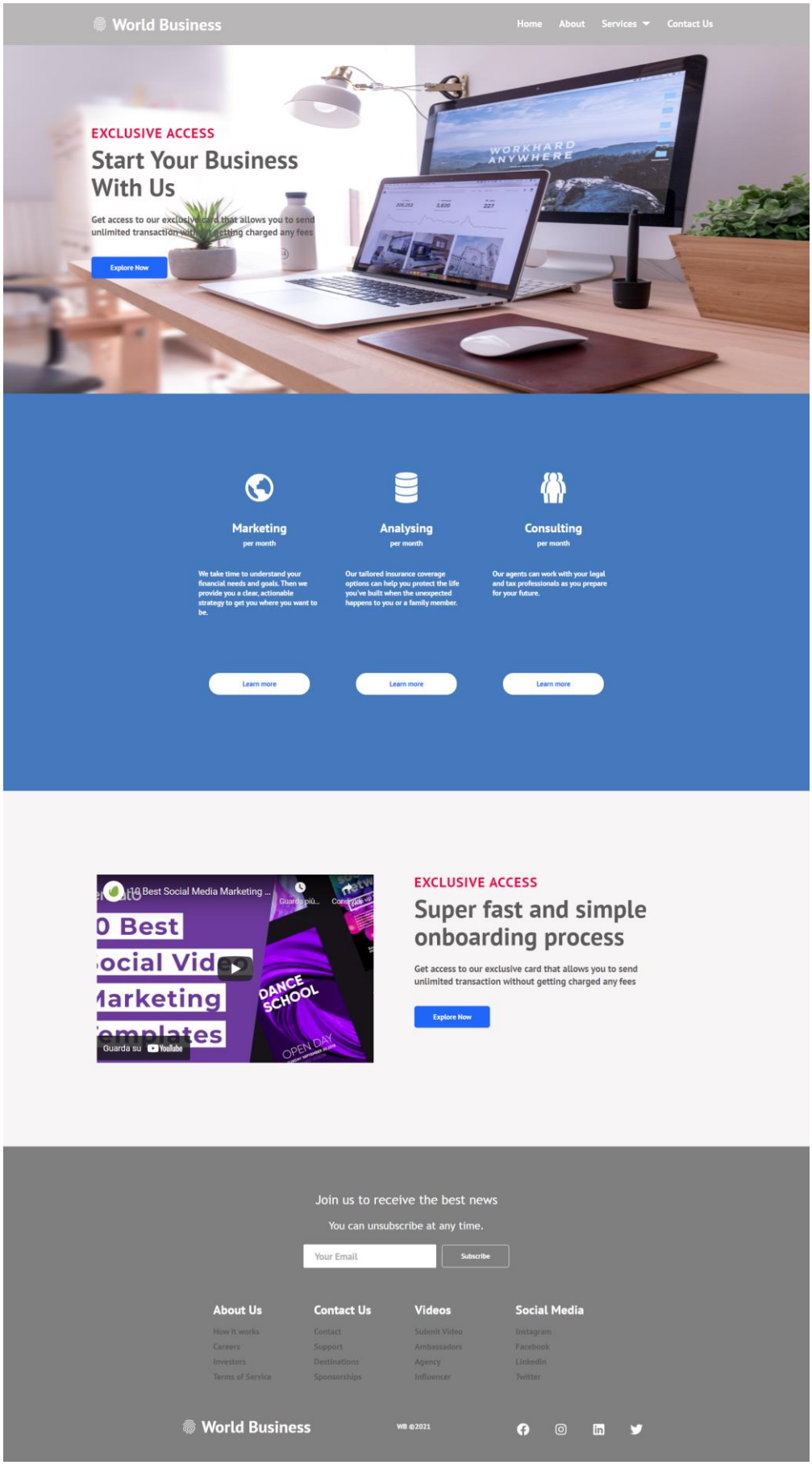


Figure3.1. home page

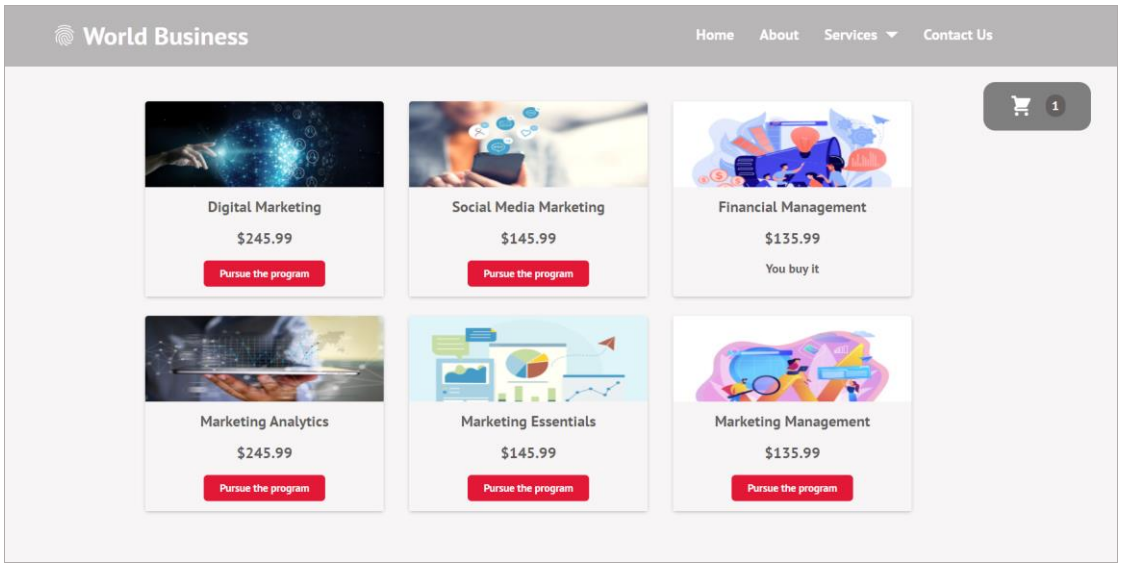


Figure3.2. Marketing page

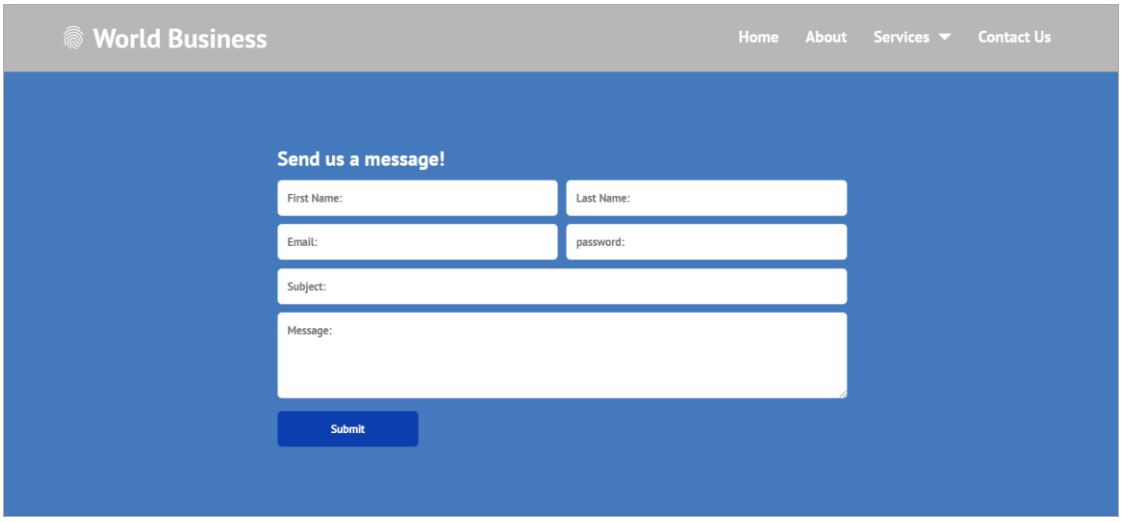


Figure3.3. Contact Us page

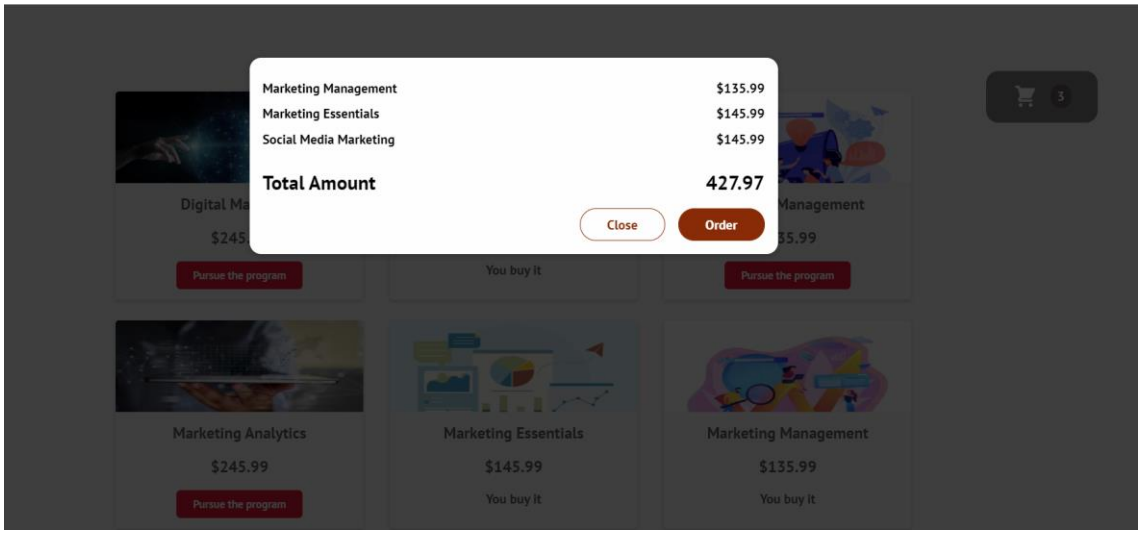


Figure3.4. order shopping

3.2 Tracking event tools

Web analytics enables companies to monitor the performance and usability of their website, as well as discover more about its visitors, such as who they are, where they came from or accessed the site, and how they use it (Fagan, 2014; Azim, & Hasan, 2018).

So, for the purpose of doing an online analysis of the mentioned above web site, in this research employed GA, GTM to investigate user behaviors.

Therefore, after developing the website and inserting the JavaScript code snippet related to each analytical tools, various configurations are made then data associated with user activities are gathered on the platforms. For instance, page view, button click, video watching, navigate in pages, shopping and transaction associated with each user were defined that there was some difference between GA and GTM.

3.2.1 Comparison of GA and GTM:

As Google Analytics becomes progressively approved, most of the developers, use Google Analytics, as a free available tool to exposure and capture the data. However, Google Tag Manager let us manage analytics tags in one place. It acts as a layer between our website and the different platforms we are using.

GTM enables developers to add numerous tags automatically, without having to alter the code on every page. A tag, for example, can manage numerous event tracking implementations in Google Tag Manager. The tag fires on a page in accordance with the rules defined for that tag. It also allows us to simplify the management of the different tags we are using.

In GA, the JavaScript code snippet should be inserted on every page of the website, while GTM play as a data layer, and after inserting code snippet on index.html, a container is created automatically. Then GA tag is constructed and fired on each page. All additional tags may be created in GTM directly, like, user transactions, button click, etc. In addition, GTM work based on tags, variable and triggers that discussed on chapter 4 and for each event, developers only need to define a tag, so hardcoding is not necessary in GTM.

Therefore, As shown in Figure3.5, Figure3.6, utilizing GTM and GA user's route in website, including pages visited per session, utilization rate of specific features, and user access mode and location, number of online users, type of transaction and event, was exposed. As well as

Total Visits (the sum of single visits to a website including one or more pageviews.), Unique Visitors and Bounced Rate (single-page sessions divided by total number of single-page sessions) were displayed.

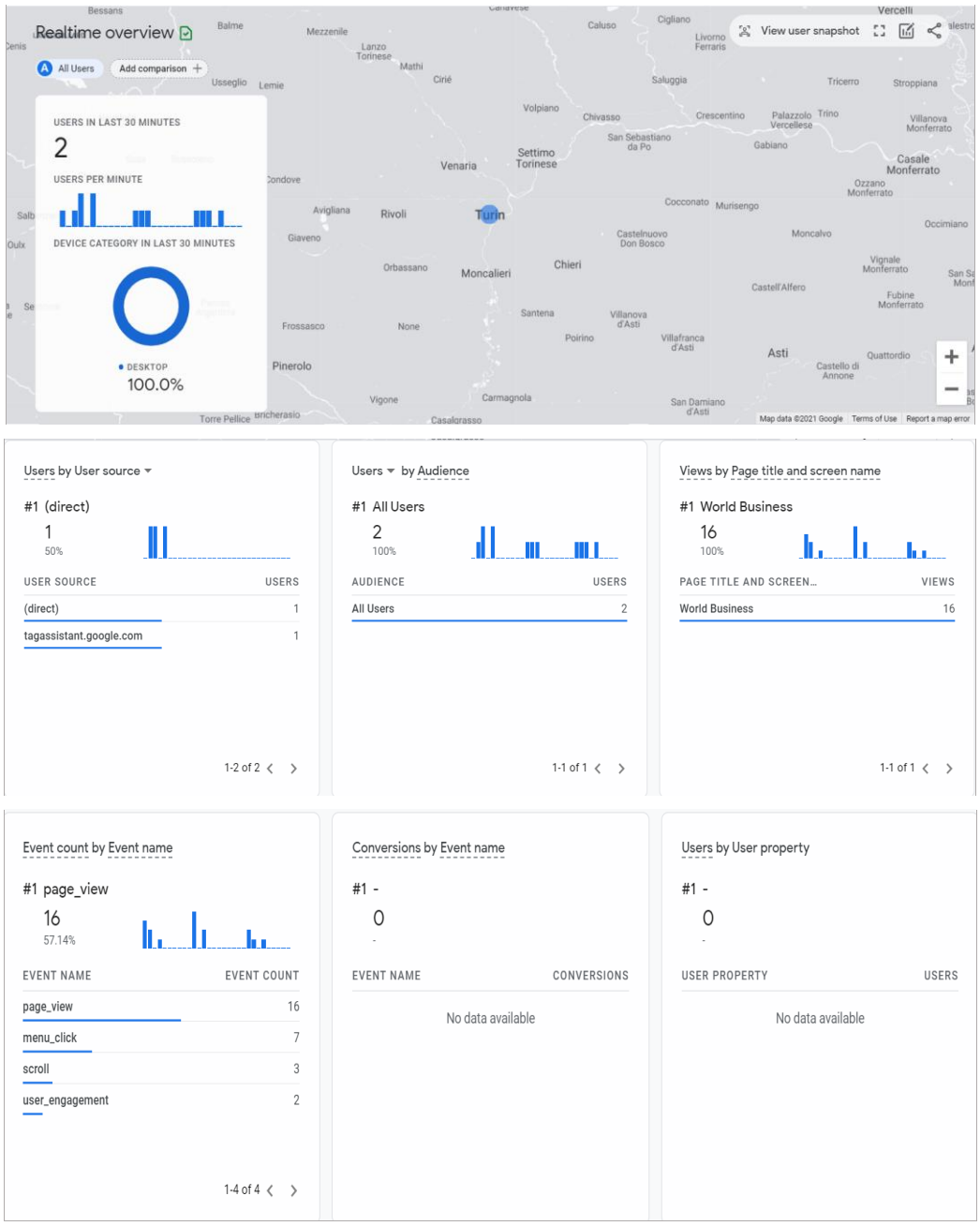


Figure3.5. Real time and events in google analytics

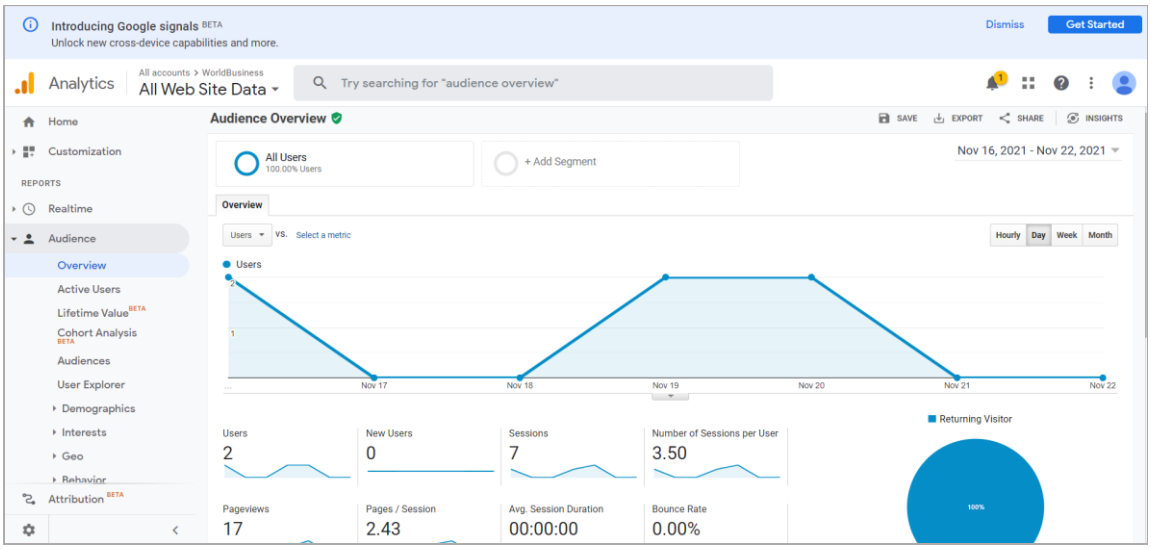


Figure3.6. dashboard in google analytics

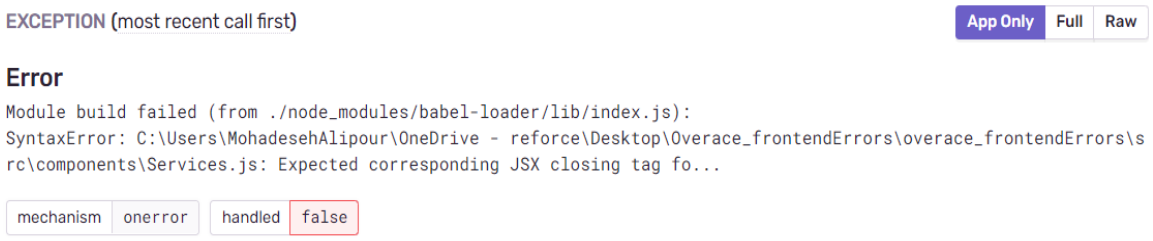
3.3 Monitoring errors tools

Sentry is a free monitoring tool that utilized to monitor the JavaScript errors, as well as for tracking and debugging production errors. Data is collected within the dashboard on Sentry's website. Sentry let us monitor 5000 errors each day. Furthermore, enable developers to quickly and effectively assess which user is impacted, and whether the issue is due to a defect or inefficient code. However, Rollbar is another error monitoring approach that assists developers in locating and correcting faults more quickly.

3.3.1 Comparison of Sentry and Rollbar:

Although the purpose of these tools is to make it easier to find, report, and repair errors in web applications, there are some differences between them.

Sentry is a free and open-source web-based platform that enable developers' team to see how their production code impact real users. In this research, data was collected in sentry dashboard, including, affected users, number of errors generated in the frontend, current location, and access mode of the user. Sentry provides a Complete Stack Trace feature that displays the whole section of code that caused the problem and, the URL and IP address of each user are revealed.



As shown in Figure3.8. and Figure3.9., Regarding the errors, the performance and latency of the transactions visualized and factors such as FCP, CLS, LCP and FID were displayed by inserting specific web vital code into the website to measure the website's health. Unfortunately, advanced sentry usage was not free which was one of the disadvantages of this approach.

However, Rollbar was also evaluated in this study. Data were collected in dashboard. It also had a complete stack trace to display exceptions in detail. The error was shown in the dashboard and viewed using the occurrence graph. The comprehensive view showed how many times each issue occurred; how many unique IP addresses were affected by it.

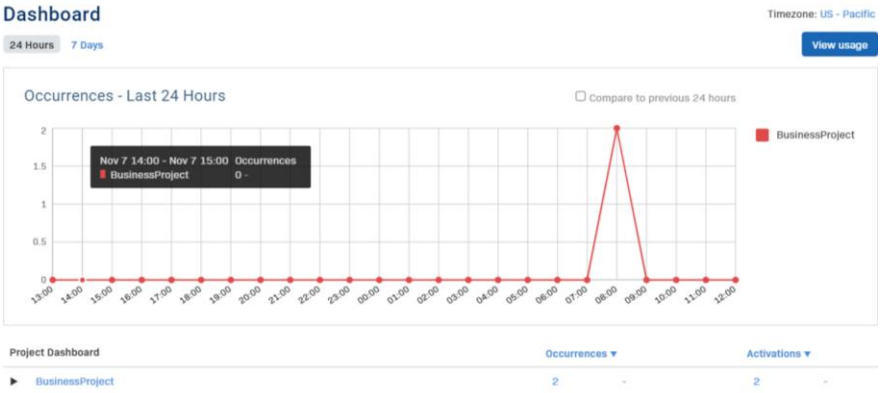


Figure3.7. Rollbar dashboard

#3 Uncaught Error: Module build failed (from ./node_modules/babel-loader/lib/index.js): SyntaxError: C:\Users\MohadesehAlipour\OneDrive - reforce\Desktop\Overace_frontendErrors\overace_frontendErrors\src\components\Services.js: Expected corresponding JSX clo

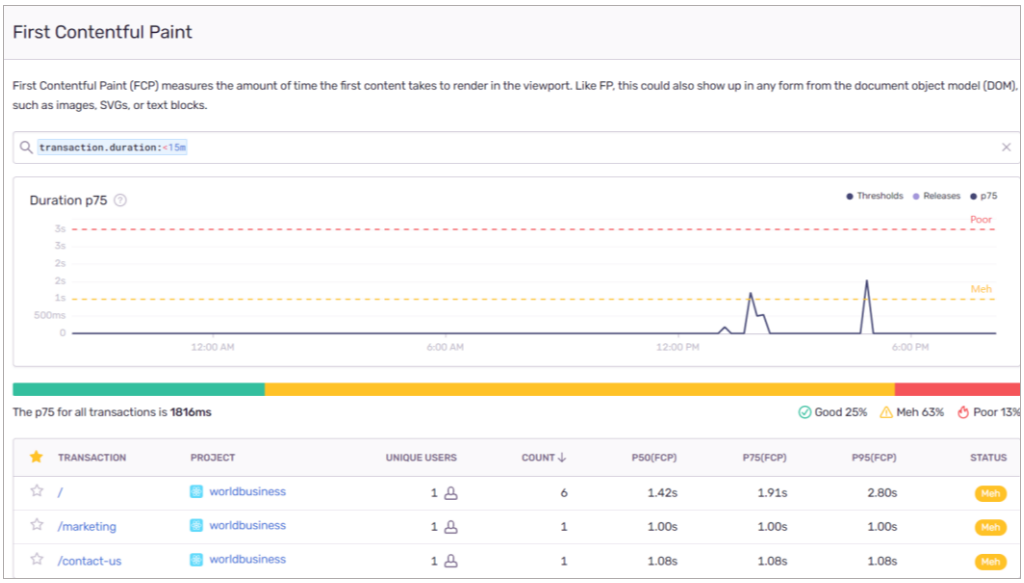


Figure3.8. FCP curve to calculate the time it takes for the first content to render on the web

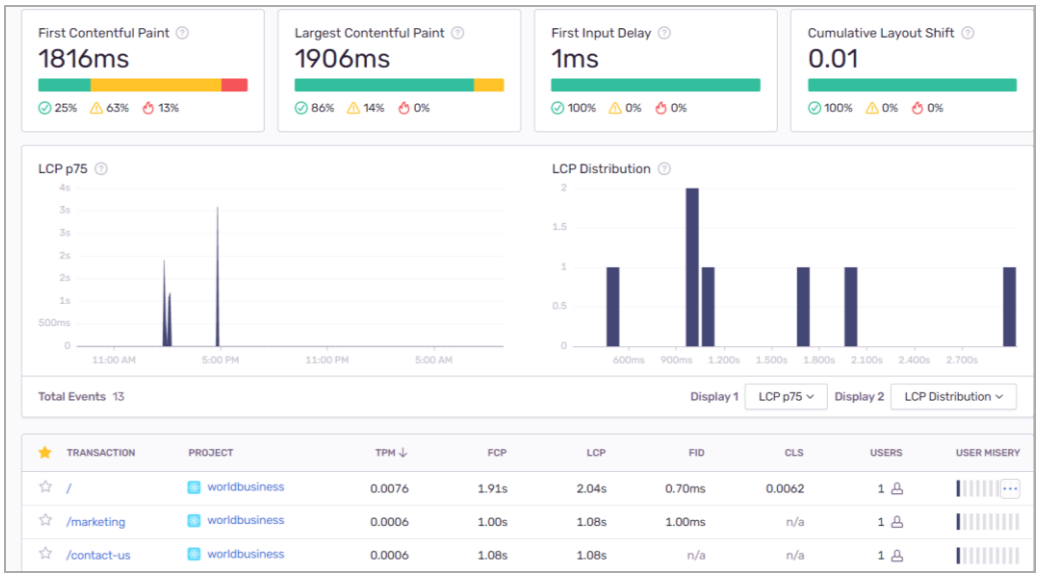


Figure3.9. LCP and other metrics

4. Tracking and monitoring Tools

Before we get into these features, let's go over some general recommended practices for integrating and utilizing these tools on any website.

4.1 Google Analytics

Google Analytics is a free web analytics service provided by google that is used to track website performance and collect visitor insights.
According to Alexa reports, 85.4% of the top 10 million sites utilize Google Analytics.
However, for everything we want Google Analytics to detect and monitor, we will need to update the website and include the specific event-tracking code in the client side of the web application, which is a snippet of JavaScript code that the website owner adds to every page of the website.

Event tracking

An event is any user action on website that cannot be automatically tracked by Google Analytics, though we need to add event tracking code in <body> tag of our website to collect the data.
Indeed, we can obtain more comprehensive data about website's usage by using event tracking, which allows to see not only what web pages visitors visit, but also what they do on those sites.

4.1.1 Create a Google Analytics 4 property and data stream:

For creating a GA4 property, go to the admin page, then click "create property".

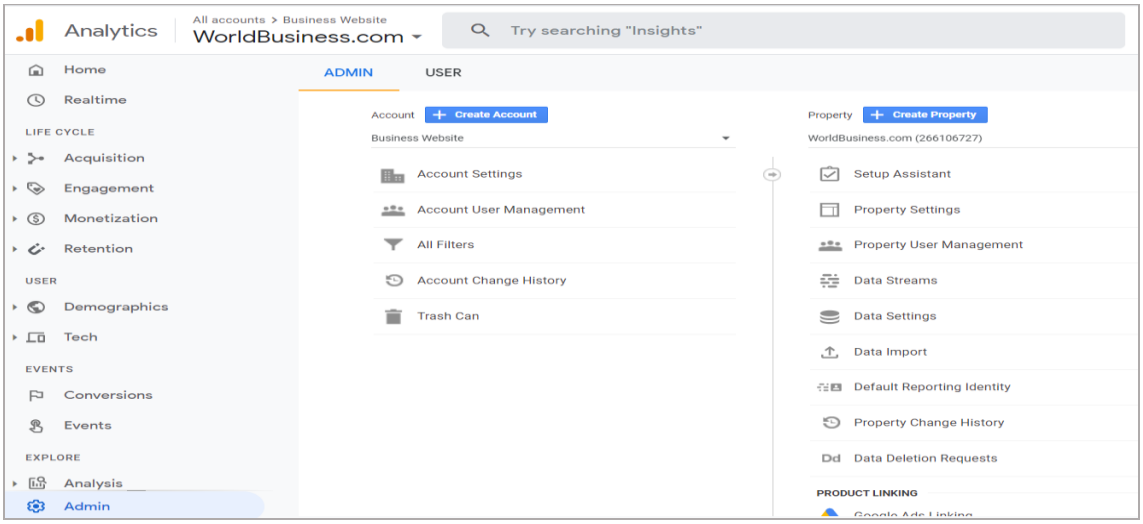


Figure 4.1. Admin page to create property

Then enter the name of our property. It might be the name of our website, our company, a brand, etc. Choose our company's country, reporting time zone. Then press Next, answer questions, click Create and then new property will be ready.

The next step to complete is to configure the first data stream. It is a data source from which events will be sent to the Google Analytics 4 property. Select Web stream. Then enter the URL of the website (for example, https://www.WorldBusiness.com).

After that, enter the name of website. Then, press the Create stream button.

After creating a data stream (web), we receive a Measurement ID.

```
<! -- Global site tag (gtag.js) - Google Analytics -->
<script        async        src="https://www.googletagmanager.com/gtag/js?id=G-
QMVTNQ9886"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag () {dataLayer.push(arguments);}
  gtag ('js', new Date ());

  gtag ('config', 'G-QMVTNQ9886');
</script>
```

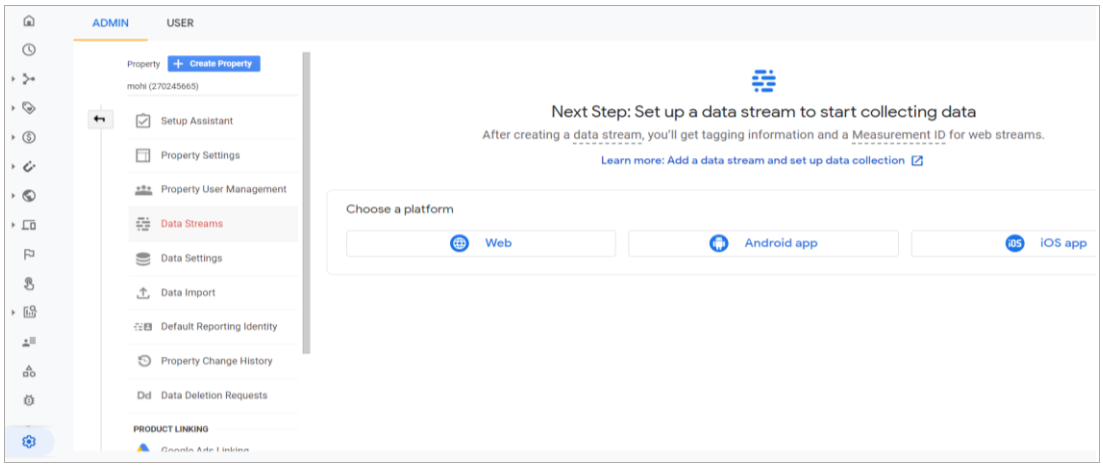


Figure 4.2. Data Streams

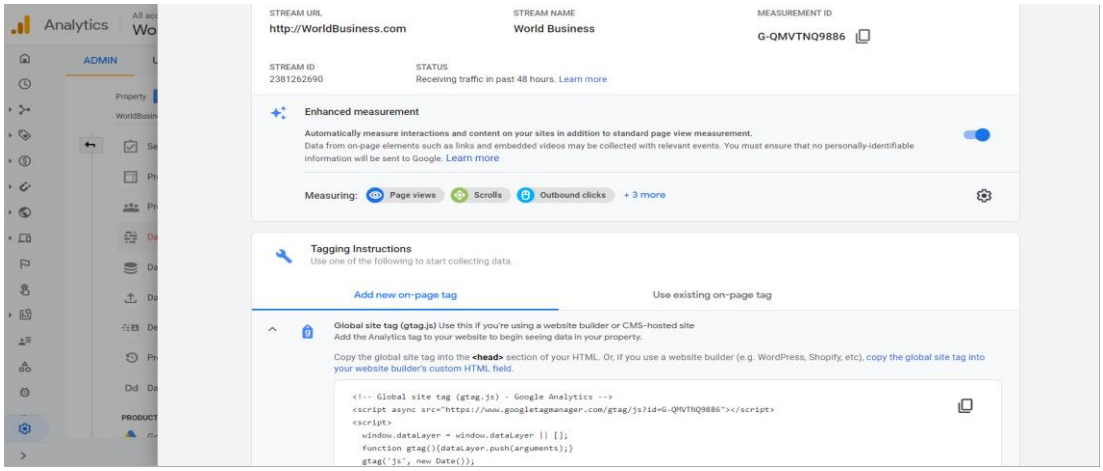


Figure 4.3 Tracking code of Google Analytics G4

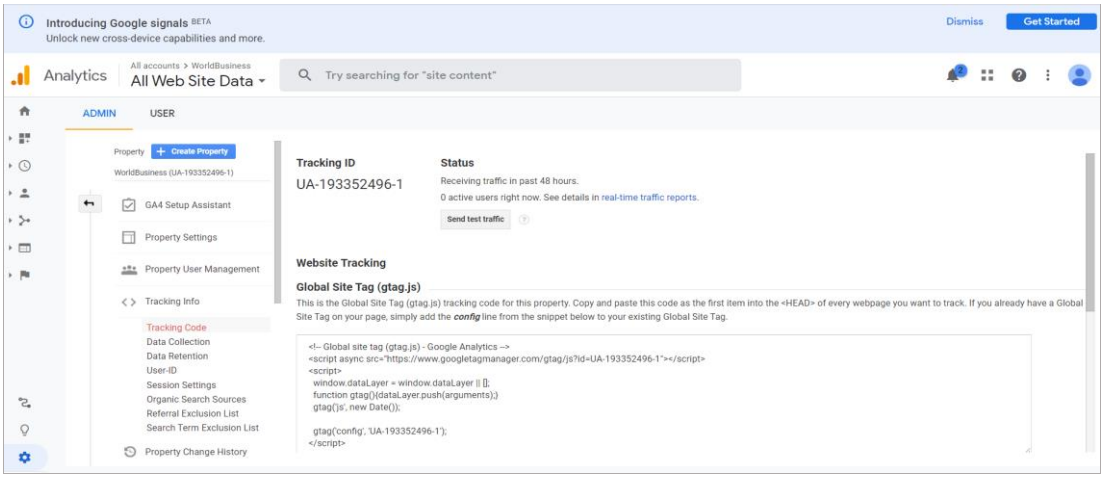


Figure 4.4 Tracking code of Google Analytics UA

```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-193352496-
1"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag () {dataLayer.push(arguments);}
  gtag ('js', new Date());

  gtag ('config', 'UA-193352496-1');
</script>
```

4.2 Google Tag Manager

GTM is a separate Google platform that utilizes a tag-based approach to inform Google Analytics what data should be collected and how it should be tracked.

Tags are snippets of code added to any website's pages. They have a range of different functions. One example is the JavaScript code tag which is vital to make Google Analytics work. That tag collects the data that Analytics needs to deliver its various reports and insights. Other tags can extract and send different information to platforms.

Google Tag Manager seamlessly integrates event-tracking functionality within its interface, which allows us to create and delete events at any time without ever directly touching website's code.

With GTM we add one piece of code to all of the pages in our website. That is called the container. We then configure GTM to fire tags on particular pages or for particular actions. So that is really the benefit of google tag manager.

It lets us centrally manage all of tags.

4.2.1 Set up a Google Tag Manager account

1. Create a website for tracking
2. Sign-up for GTM at https://tagmanager.google.com/ for free.
3. Provide basic information about our website including the website address or URL
4. Obtain html coding (tracking code) from GTM
5. Insert Tracking id in source code (index.html)

Copy and paste the following code as high as possible into the <head> section of the page:

```
    <!-- Google Tag Manager -->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
  new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
  j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=

'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j
,f);
  })(window,document,'script','dataLayer','GTM-K2XFQWR');</script>
  <!-- End Google Tag Manager -->
```

Also, insert this code just after the first <body> tag:

```
    <!-- Google Tag Manager (noscript) -->
<noscript><iframe src="https://www.googletagmanager.com/ns.html?id=GTM-
K2XFQWR"
  height="0" width="0"
style="display:none;visibility:hidden"></iframe></noscript>
  <!-- End Google Tag Manager (noscript) -->
```

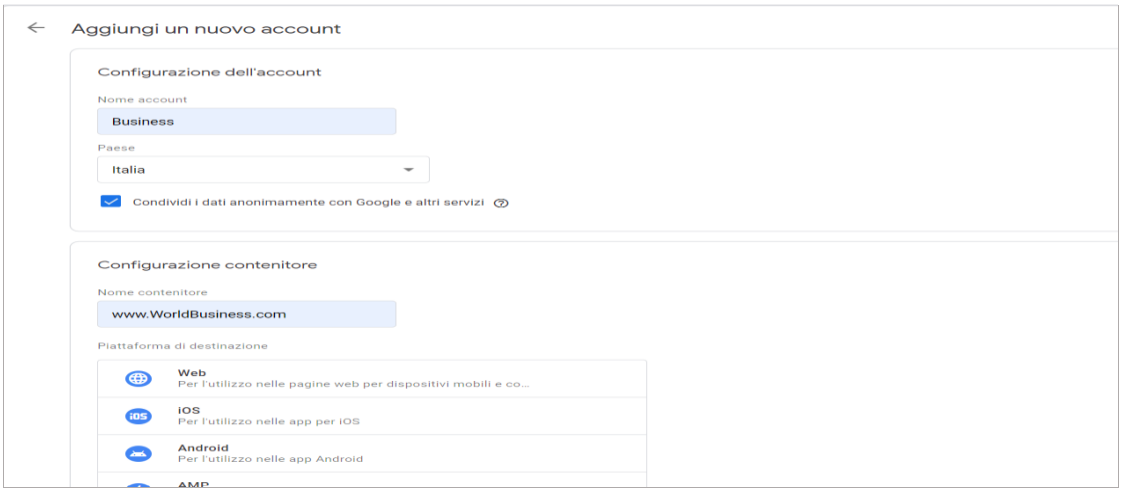


Figure 4.5. Create new account in GTM

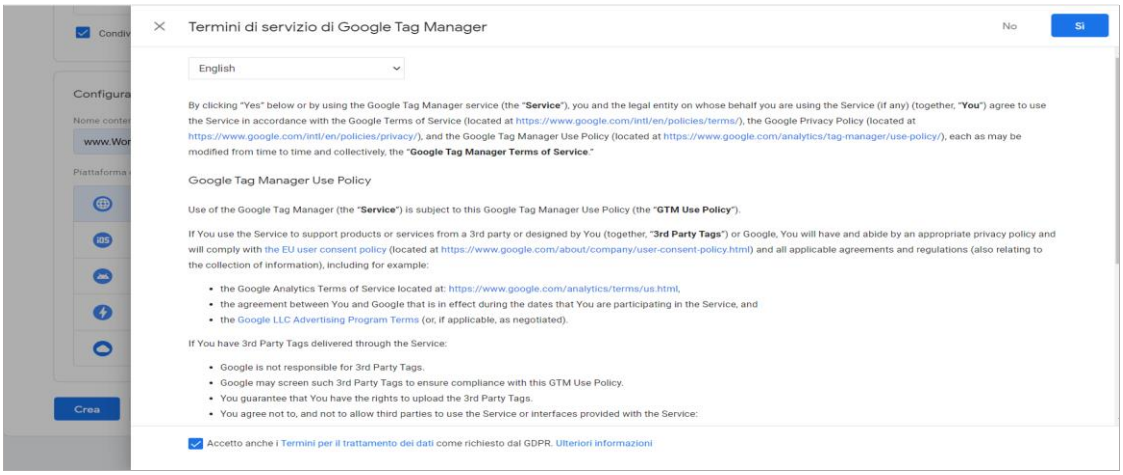


Figure 4.6. GTM terms of services agreement

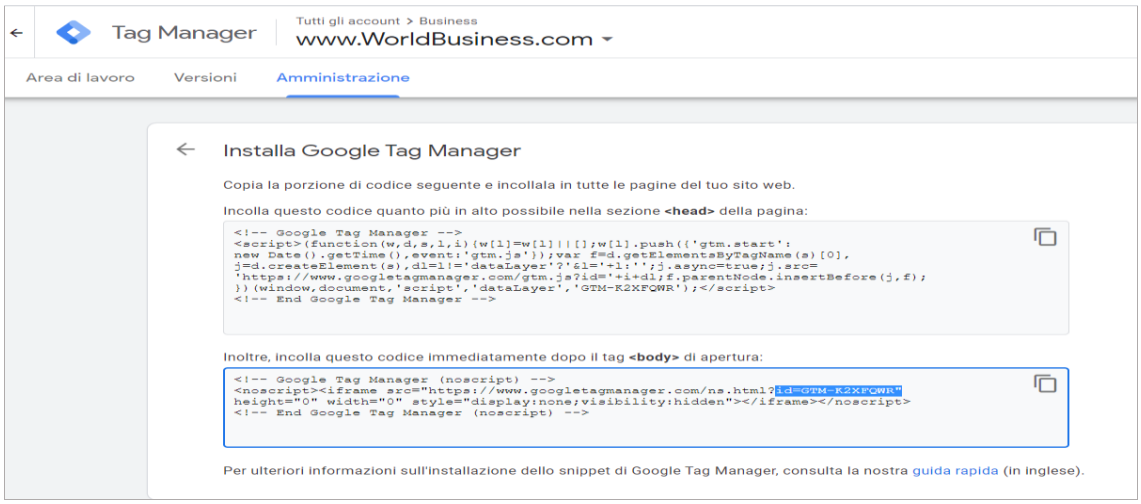


Figure 4.7. Tracking code of Google Analytics GTM

After creating a GTM account, we have access to the GTM web administrative interface, where we can create new containers. The container is an individual website or app that we are tracking. When create a new container, a GTM tracking number and tracking code are automatically generated. The GTM container snippet must get added to every web page we want to track. If website already has Google Analytics tracking code on it, that code must be removed as add the new GTM container snippet or will accidently track website's data twice in Google Analytics account. Ideally, the GTM container snippet should be placed at the opening of the <body> tag within a web page for best data collection.
Once the container is created, we have access to administrative interface then we can setup tags, triggers, and variables to tell GTM how to function.
Tag is a piece of JavaScript code; that runs on a web page and is generally associated with a particular product, such as Google Analytics. They typically collect data and then send it to a particular platform like a google analytics tag which collects information about the people viewing website or we can also use a tag to add additional functionality to our website. The trigger tells the tag when to fire (or run) and collect data on a specific web page if the variable is true. Variables are Placeholder for information.

4.2.2 Install Google Analytics 4 with Google Tag Manager:

After creating a data stream (web), we receive a Measurement ID.
Then go to your Google Tag Manager container > Tags > New and choose GA4 configuration. In the Measurement ID field, enter the ID that copied in the GA4 interface.
In the Triggering section, select All Pages and then name the tag, e.g., GA4 - Config - Pageview. Then click save > preview/test the new tag and publish the changes in Google Tag Manager container.

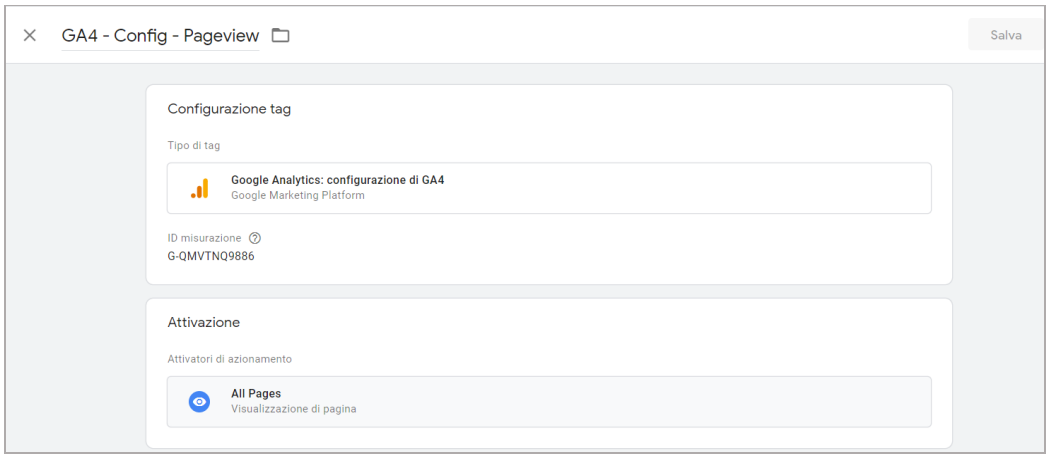


Figure4.8. Configuration tag

In GA4, events can be divided into 4 categories:

1. Automatically collected events (first_visit, session_start, user_engagement)
2. Enhanced Measurement events (scroll, file_download)

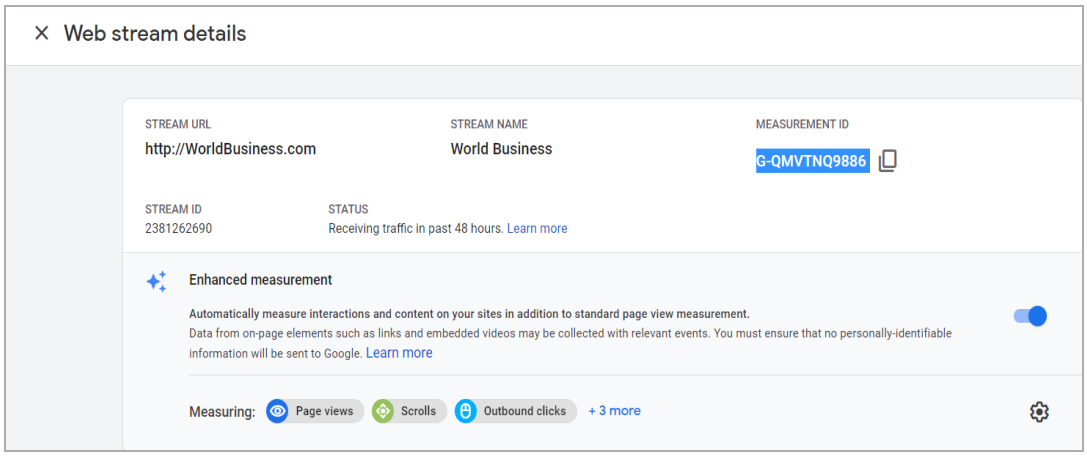


Figure4.9. Enhanced Measurement

3. Recommended events
   https://support.google.com/analytics/answer/9267735?hl=en&ref_topic=9756175
4. Custom events

4.2.3 Creating Events in Google Tag Manager

Creating Tag:

The first tag we are going to build is a Google Analytics tag, which will sync Google Analytics account with the GTM container. As a result, GTM links to our Google Analytics account. To do this, go the Tags area within our GTM container and click on the new button. Once new tag screen is appeared, name the tag to organize tags. Next, choose Google Analytics as our product. It can be either Universal Analytics (UA) or Classic Google Analytics (GA4) as mentioned in the first section, then click Continue. Finally, add Google Analytics Tracking ID number.
It should look like UA-193352496-1 or G-QMVTNQ9886 for Universal Analytics (UA) and Classic Google Analytics (GA4) respectively.

Select Page View for Track Type, Click Continue, then, by trigger options tell the tag when to run. Since we inserted the Google Analytics tracking code, we will track every web page on which the GTM container snippet appears, so select All Pages for this tag. Click the Create Tag button and publish this change on GTM.
After Publishing the tag withing GTM, we can select the Preview and Debug option to determine if the tags work as expected. When browser loads the website's web page, the GTM debug window (shown in figure 4.8, figure 4.9 and 4.10) will open at the bottom of the screen and list all tags and the tags that fired. This will only work if we have already included the GTM container snippet to our page.
One of the most significant advantages of GTM is that, in order to collect data associated with events, we have to add Google Analytics event-tracking script to each event we want to track, but GTM replaces this time-consuming process with a simple event tag that does not require direct editing of the website's code.

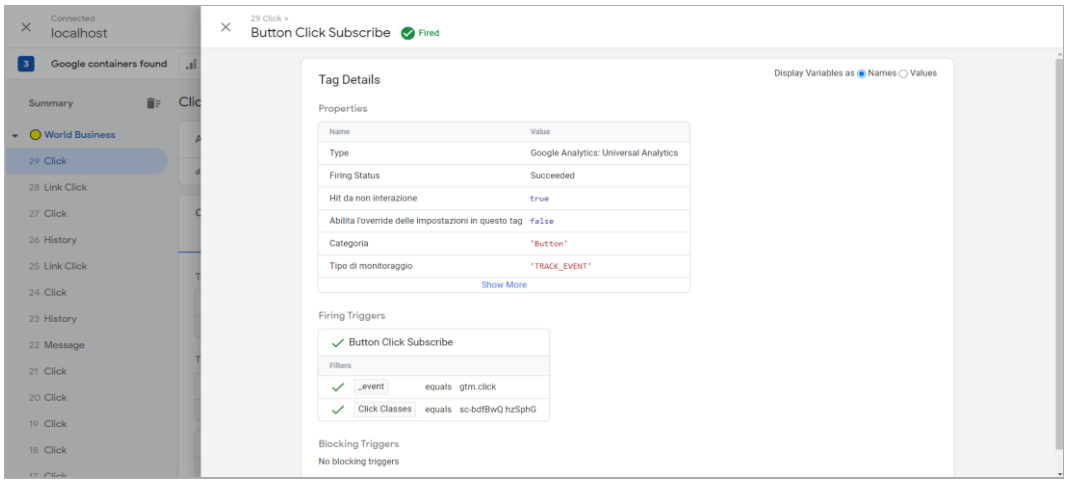


figure4.10 Button click fired

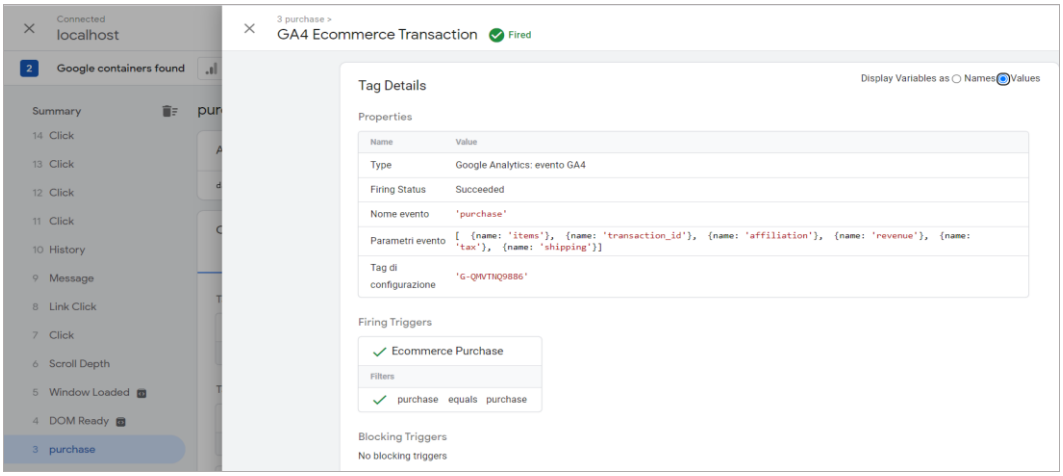


figure4.11 Transaction fired

How to make sure that data was properly sent to Google Analytics 4:

By Debug View and by clicking the preview button in GTM, we can enable debug mode.

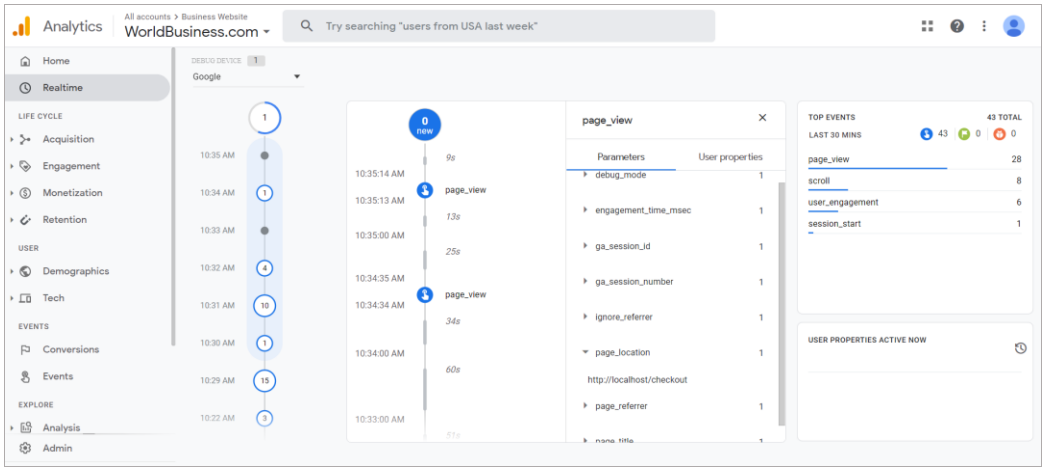


Figure4.12. Debug view

Youtube video tracking

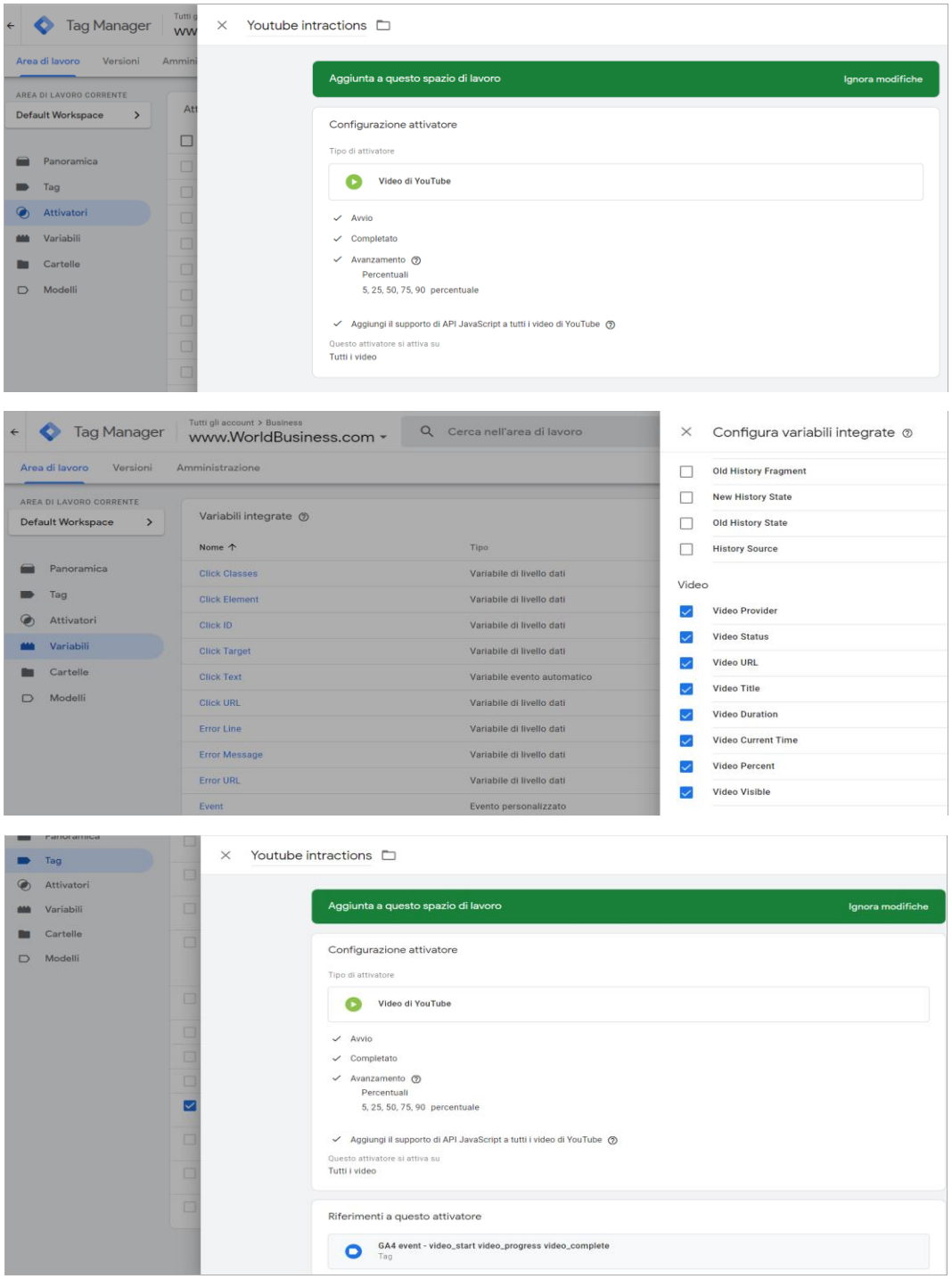


Figure4.13. Video tracking

Purchase Tracking:

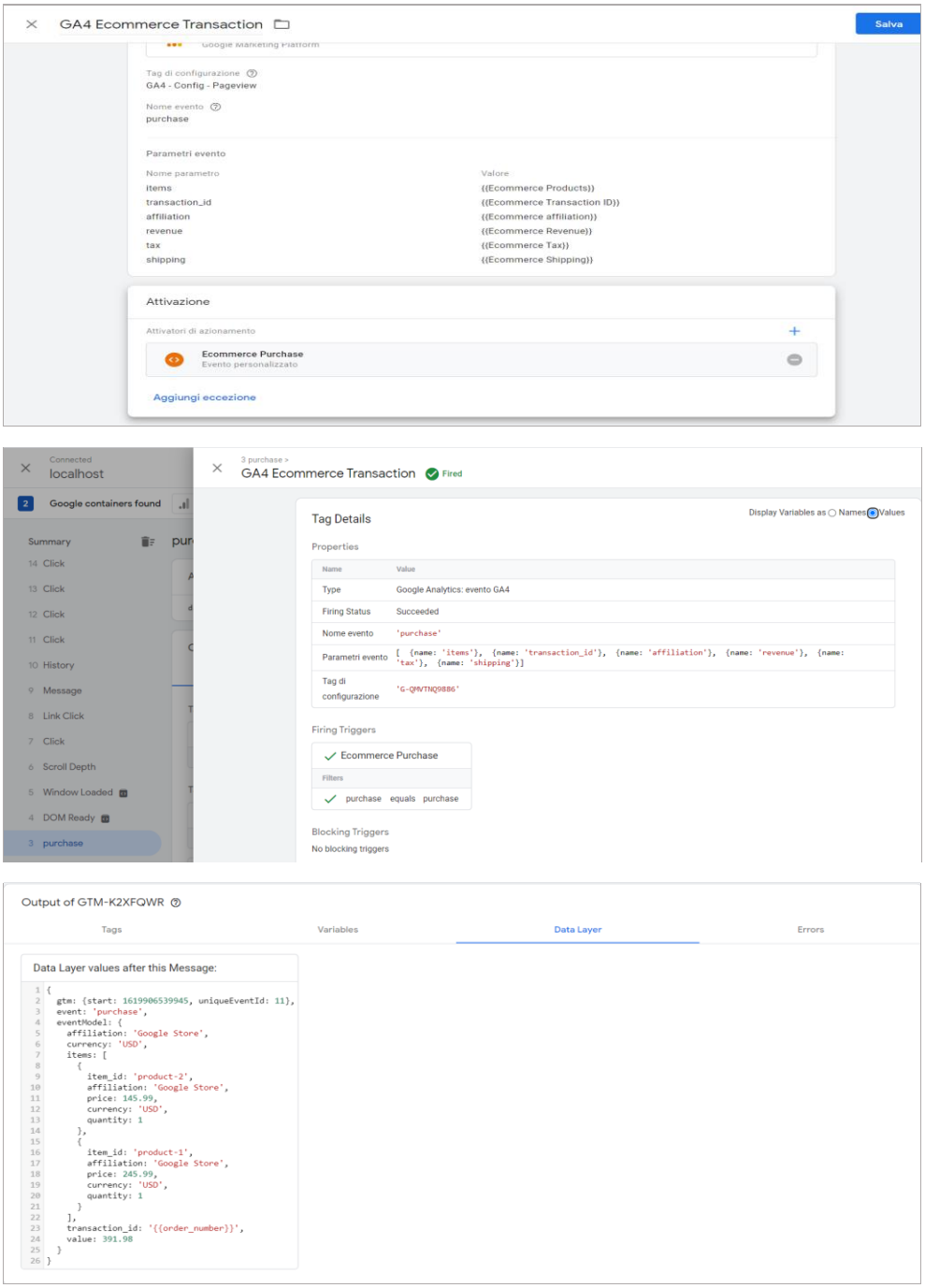
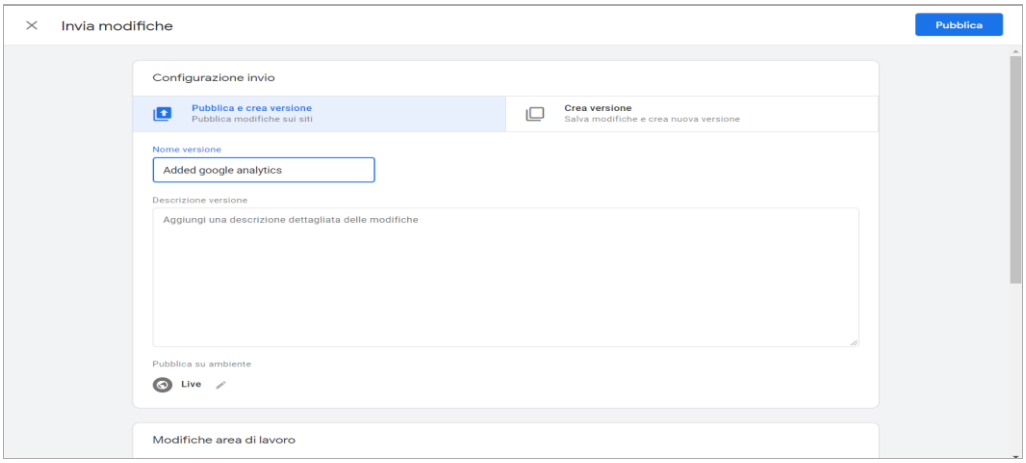
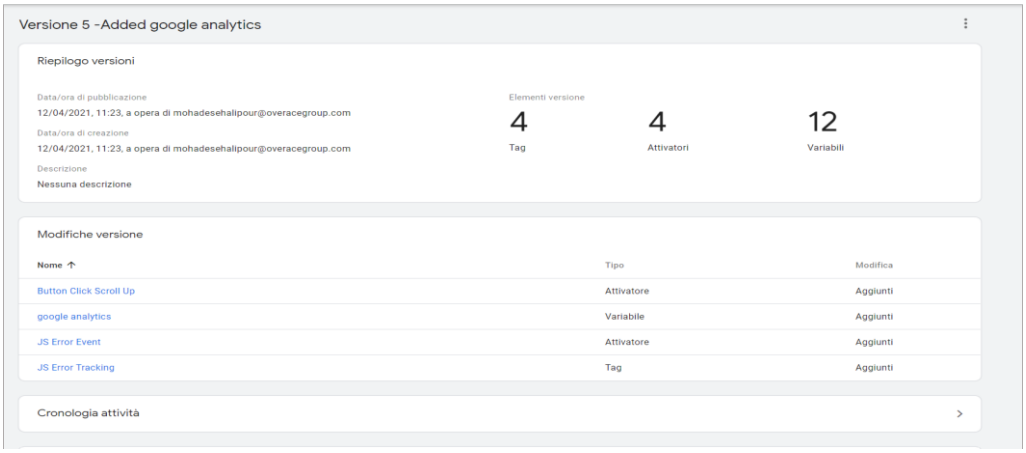


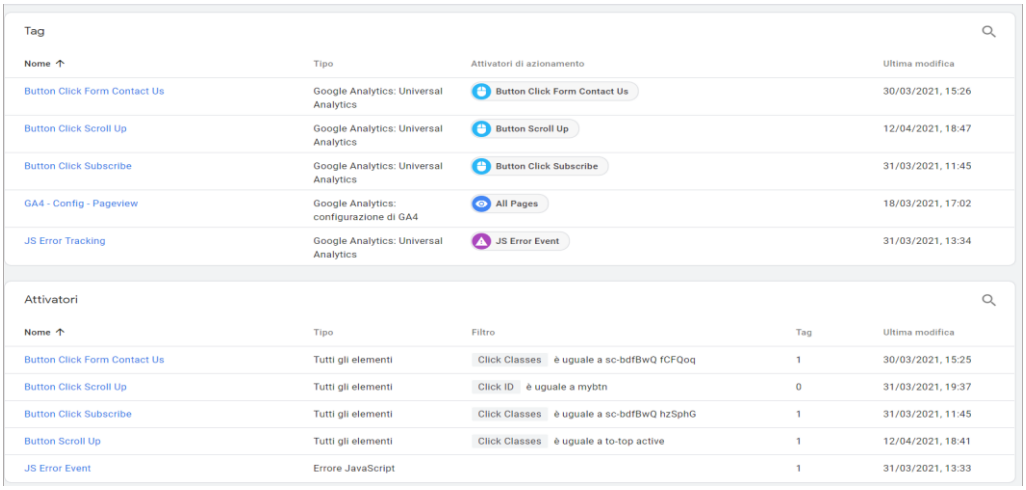
Figure4.14. Purchase tracking

How to publish tag on website:

1. Click Submit(invia)
2. Choose "version name" then click publish







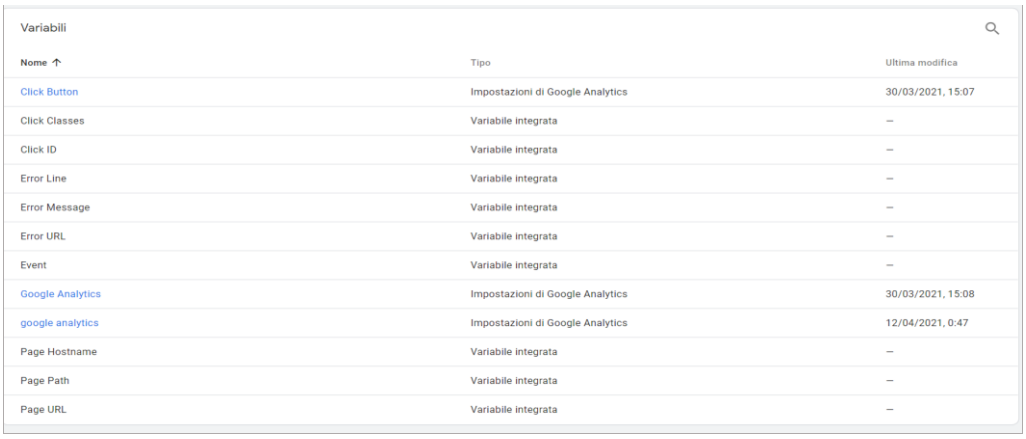


Figure4.15. Publish tag

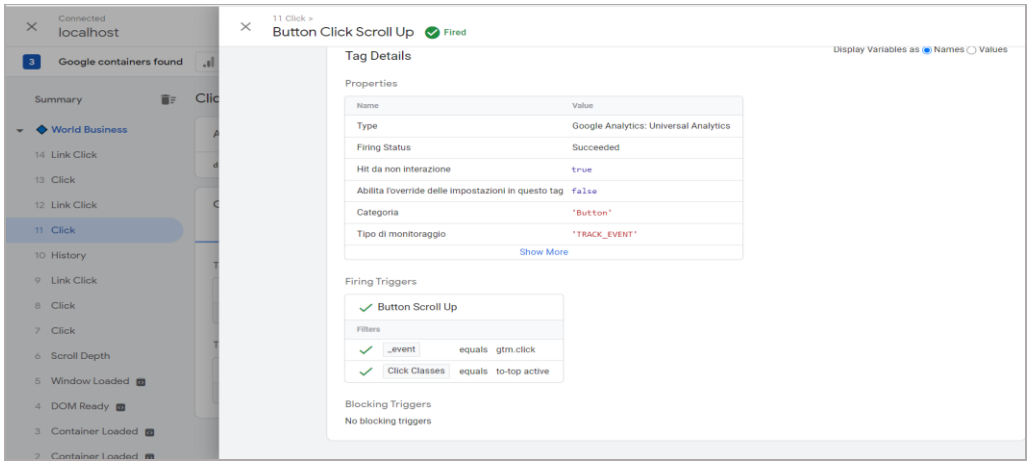
How to fire a tag:

1. Click Triggering (Attivazione)
2. Select all pages
3. Then click save

Trigger:

4. Select triggering -> click + button on top to create a new trigger to tag -> Name the trigger and click trigger configuration -> select all elements -> click some clicks -> "click classes" -> equals -> button className in our website -> save.

5. Then preview the container and check if we are tracking the button click correctly. click "preview".



We also check in google analytics to see if the tag is firing correctly.
select Realtime and events.
Then we can see that the button has been tracked as an event into google analytics.

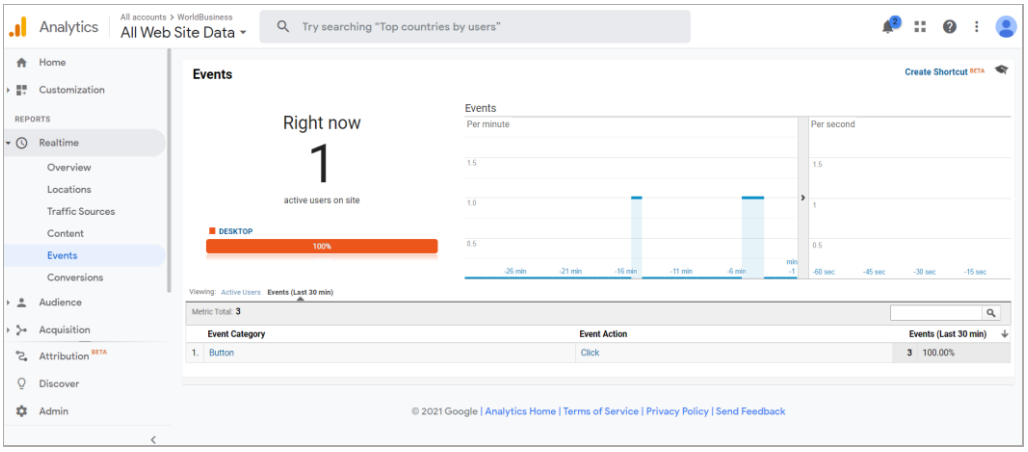


Figure4.16. Event tracking in google analytics

Error monitoring tools:

Instead of worrying about getting report problems from users, developers and administrators may focus on the important stuff, such as producing high-quality code, by employing error monitoring tools.

Error monitoring is a collection of tools for detecting and correcting errors in various applications, most of which are web-based.

The goal of error monitoring:

1. Find and report errors
2. Identify critical errors
3. Analyze the errors in our app.

4.3 Implement and Configure Sentry:

First of all, should log in to the sentry organization to create a project, then to represent our app, Sentry assigns a Data Source Name (DSN), which enter into the website's source code. (index.js).

https://docs.sentry.io/product/sentry-basics/guides/integrate-frontend/create-new-project/

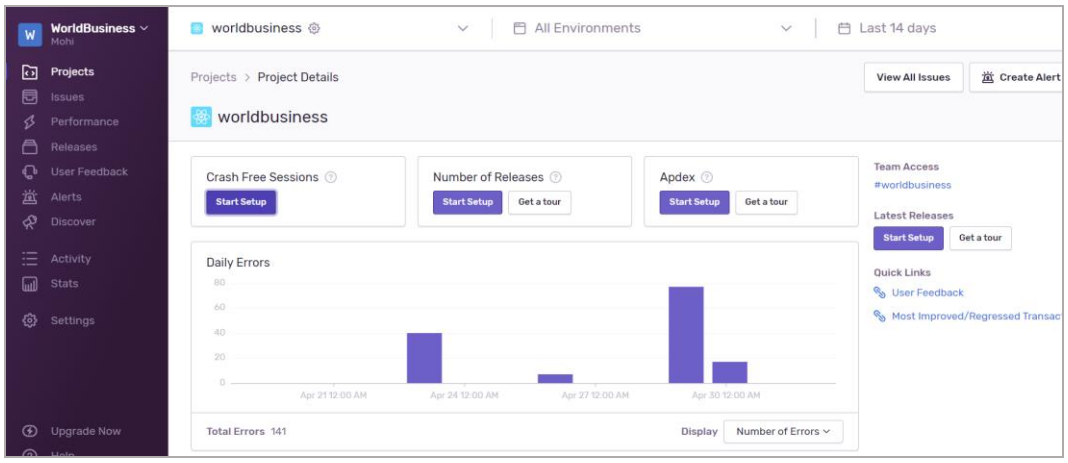


Figure4.17. New project is created

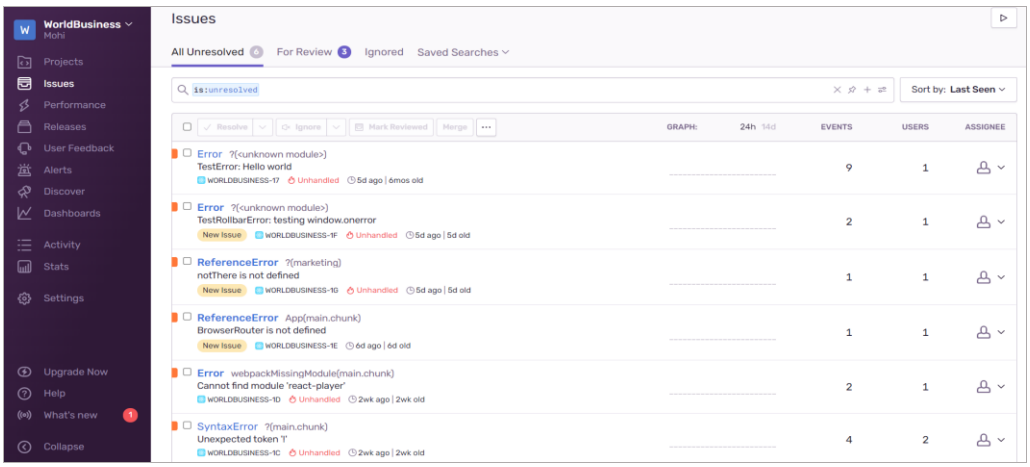
```
Sentry.init({
  dsn:
"https://40cffe687e164cb0b2ea8d5247e73ffb@o562363.ingest.sentry.io/5700808",
  integrations: [new Integrations.BrowserTracing()],

  // We recommend adjusting this value in production, or using tracesSampler
  // for finer control
  tracesSampleRate: 1.0,
  release: "worldbusiness" + version,
});
```

The DSN tells the SDK where to send the events, associating them with the project just created. Sentry uses an SDK to collect data throughout the application's execution. So, the application must be configured using the code below in (index.js).

```
import * as Sentry from "@sentry/react";
import { Integrations } from "@sentry/tracing";
```

npm install --save @sentry/react @sentry/tracing

Then, create an alert to notify developer that something went wrong. When an error occurs, this page will appear, revealing all error information so that the team that got the notification can resolve the problem.



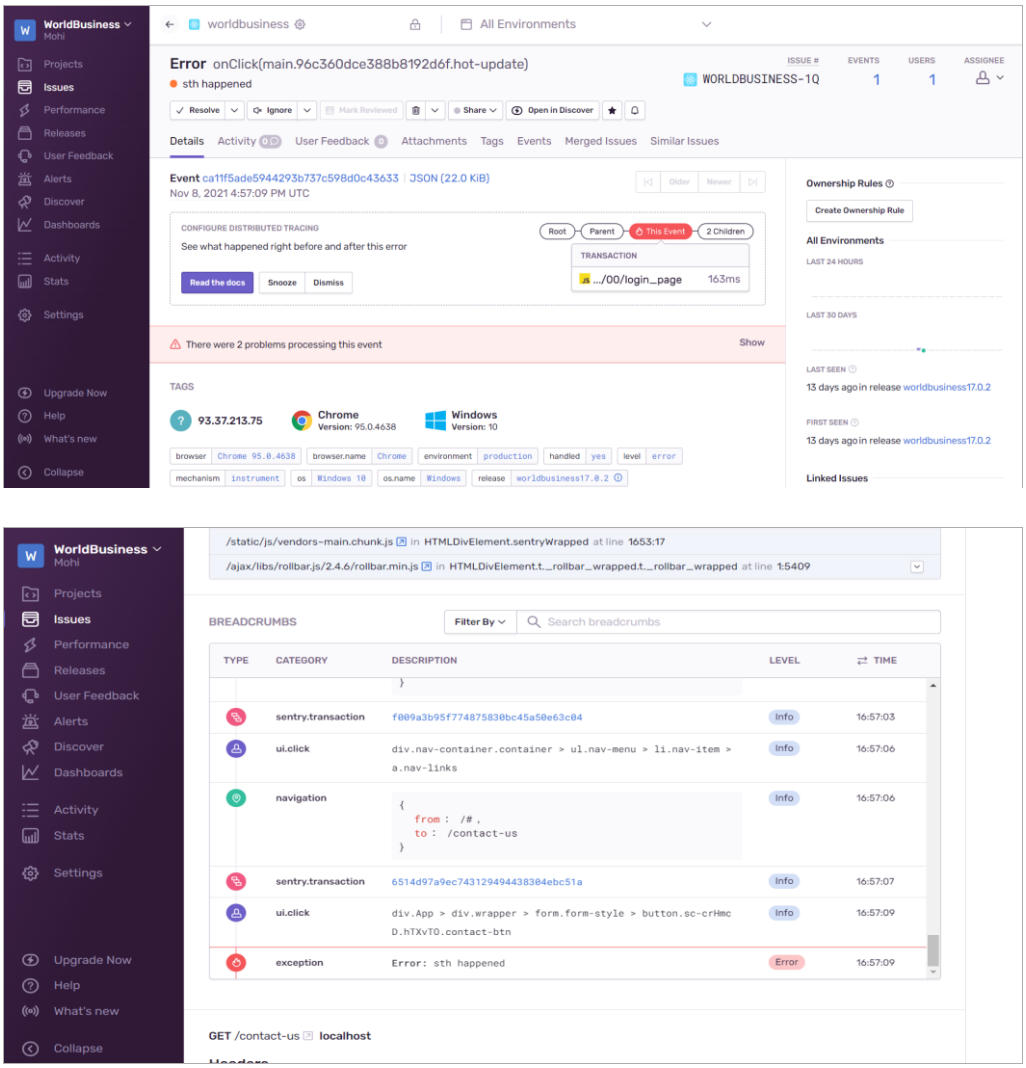


Figure4.18. Errors monitored in sentry

4.3.1 Web performance and web vitals:

https://docs.sentry.io/product/performance/web-vitals/

Using performance monitoring, sentry examines application performance, measures parameters such as latency and bandwidth, and illustrates the effect of errors. The main view in sentry is the Performance page, and it offers graphs that illustrate transactions or statistics, as well as a place for administrators to investigate or peruse transaction data.

Each transaction has a summary page that is unique to each transaction and provides a better insight of the website's overall health.

This summary page is called as "web vital" in the frontend, and it displays information on the corresponding transaction. Sentry SDKs capture Web Vitals data and apply it into frontend transactions.

```
const reportWebVitals = onPerfEntry => {
      if (onPerfEntry && onPerfEntry instanceof Function) {
      import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB })
=> {
        getCLS(onPerfEntry);
        getFID(onPerfEntry);
        getFCP(onPerfEntry);
        getLCP(onPerfEntry);
        getTTFB(onPerfEntry);
      });
      }};
      export default reportWebVitals;
```

Largest Contentful Paint (LCP) counts the time it takes for the largest content in the viewport to display. LCP assists developers in determining how long it takes for the user to view the primary content on the website.

For every unexpected element change throughout the rendering process, Cumulative Layout Shift (CLS) is the total of individual layout change scores.

The time it takes for the first content to render in the viewport is measured by First Contentful Paint (FCP). This could be in the form of photos, SVGs, or text blocks from the document object model (DOM). FCP aids developers in determining how long it takes for a user to notice a change in the page's content.

First Input Delay (FID) is a metric that indicates how quickly the user reacts while interacting with the viewport that Clicking a button, link, or other custom JavaScript controller is an example of an action. FID gives vital information about successful and failed interactions on an application page.

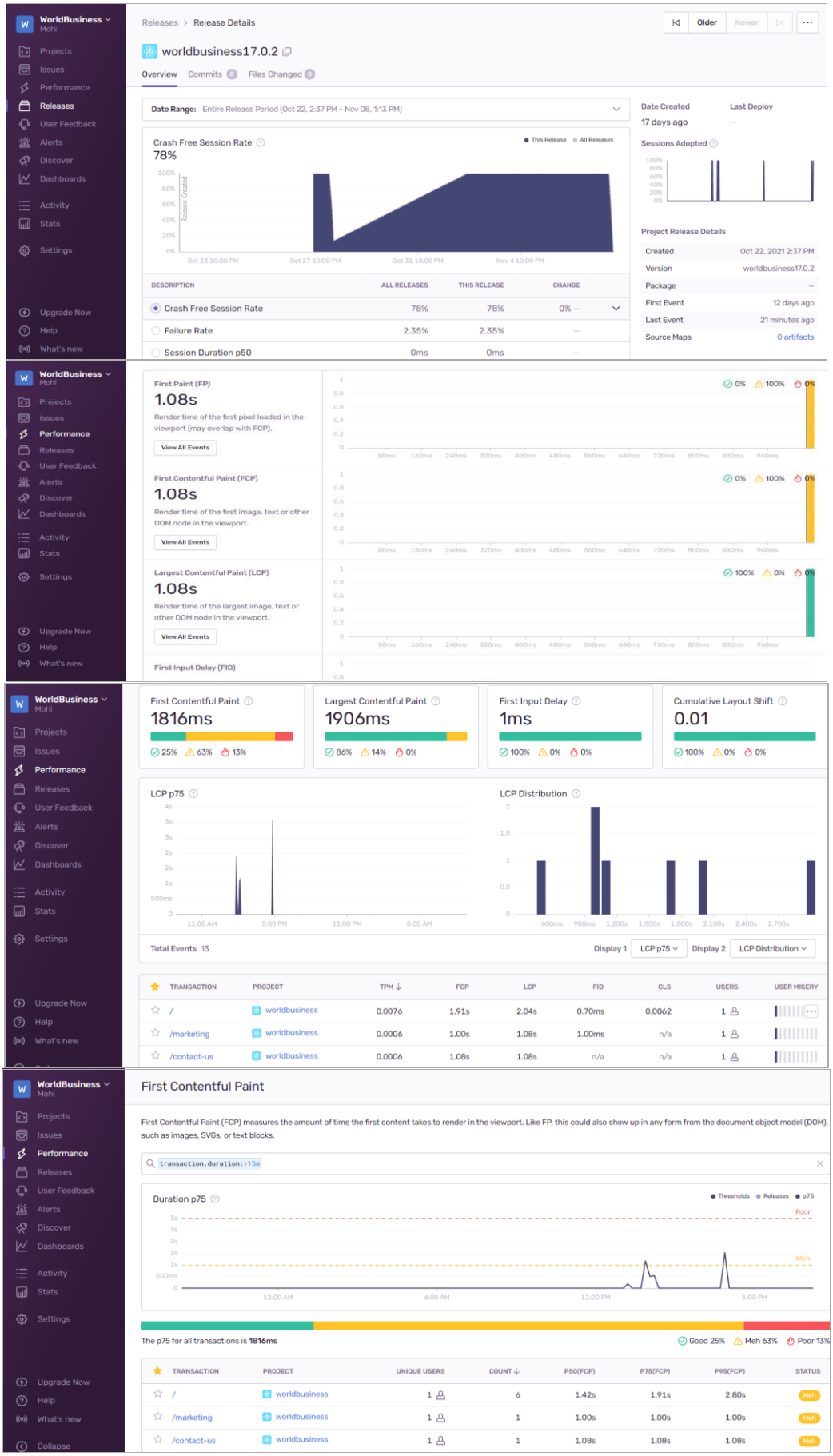


Figure4.19. Web performance and web vital

4.4 create and setup Rollbar:

https://docs.rollbar.com

To establish a project in Rollbar, first log in to the Rollbar organization, then select the programming language in which the website has be built. Finally, get this React snippet code, which insert in <head> tag of index.html to collect the website's data.

```
    <script>
      var _rollbarConfig = {
        accessToken: '55a6bbe6b2e6486b8f0487b4aac09af3',
        captureUncaught: true,
        captureUnhandledRejections: true,
        payload: {
          environment: 'production',
        },
      }
      // Rollbar Snippet
      !(function (r) {
        function e(n) {
          if (o[n]) return o[n].exports
          var t = (o[n] = { exports: {}, id: n, loaded: !1 })
          return r[n].call(t.exports, t, t.exports, e), (t.loaded = !0),
t.exports
        }
        var o = {}
        return (e.m = r), (e.c = o), (e.p = ''), e(0)
      })([
        function (r, e, o) {
          'use strict'
          var n = o(1),
            t = o(4)
          ;(_rollbarConfig = _rollbarConfig || {}),
            (_rollbarConfig.rollbarJsUrl =
              _rollbarConfig.rollbarJsUrl ||

'https://cdnjs.cloudflare.com/ajax/libs/rollbar.js/2.4.6/rollbar.min.js'),
            (_rollbarConfig.async =
              void 0 === _rollbarConfig.async || _rollbarConfig.async)
          var a = n.setupShim(window, _rollbarConfig),
            l = t(_rollbarConfig)
          ;(window.rollbar = n.Rollbar),
            a.loadFull(window, document, !_rollbarConfig.async,
_rollbarConfig, l)
        },
```

```
function (r, e, o) {
  'use strict'
  function n(r) {
    return function () {
      try {
        return r.apply(this, arguments)
      } catch (r) {
        try {
          console.error('[Rollbar]: Internal error', r)
        } catch (r) {}
      }
    }
  }
  function t(r, e) {
    ;(this.options = r), (this._rollbarOldOnError = null)
    var o = s++
    ;(this.shimId = function () {
      return o
    }),
      'undefined' != typeof window &&
        window._rollbarShims &&
        (window._rollbarShims[o] = { handler: e, messages: [] })
  }
  function a(r, e) {
    if (r) {
      var o = e.globalAlias || 'Rollbar'
      if ('object' == typeof r[o]) return r[o]
      ;(r._rollbarShims = {}), (r._rollbarWrappedError = null)
      var t = new p(e)
      return n(function () {
        e.captureUncaught &&
          ((t._rollbarOldOnError = r.onerror),
          i.captureUncaughtExceptions(r, t, !0),
          i.wrapGlobals(r, t, !0)),
          e.captureUnhandledRejections &&
            i.captureUnhandledRejections(r, t, !0)
        var n = e.autoInstrument
        return (
          e.enabled !== !1 &&
            (void 0 === n ||
              n === !0 ||
              ('object' == typeof n && n.network)) &&
            r.addEventListener &&
            (r.addEventListener('load', t.captureLoad.bind(t)),
            r.addEventListener(
              'DOMContentLoaded',
              t.captureDomContentLoaded.bind(t)
            )),
          (r[o] = t),
```

```
        t
      )
    })()
  }
}
function l(r) {
  return n(function () {
    var e = this,
      o = Array.prototype.slice.call(arguments, 0),
      n = { shim: e, method: r, args: o, ts: new Date() }
    window._rollbarShims[this.shimId()].messages.push(n)
  })
}
var i = o(2),
  s = 0,
  d = o(3),
  c = function (r, e) {
    return new t(r, e)
  },
  p = d.bind(null, c)
;(t.prototype.loadFull = function (r, e, o, t, a) {
  var l = function () {
      var e
      if (void 0 === r._rollbarDidLoad) {
        e = new Error('rollbar.js did not load')
        for (var o, n, t, l, i = 0; (o = r._rollbarShims[i++]); )
          for (o = o.messages || []; (n = o.shift()); )
            for (t = n.args || [], i = 0; i < t.length; ++i)
              if (((l = t[i]), 'function' == typeof l)) {
                l(e)
                break
              }
      }
      'function' == typeof a && a(e)
    },
    i = !1,
    s = e.createElement('script'),
    d = e.getElementsByTagName('script')[0],
    c = d.parentNode
  ;(s.crossOrigin = ''),
    (s.src = t.rollbarJsUrl),
    o || (s.async = !0),
    (s.onload = s.onreadystatechange = n(function () {
      if (
        !(
          i ||
          (this.readyState &&
            'loaded' !== this.readyState &&
            'complete' !== this.readyState)
```

```
        )
      ) {
        s.onload = s.onreadystatechange = null
        try {
          c.removeChild(s)
        } catch (r) {}
        ;(i = !0), l()
      }
    })),
    c.insertBefore(s, d)
}),
  (t.prototype.wrap = function (r, e, o) {
    try {
      var n
      if (
        ((n =
          'function' == typeof e
            ? e
            : function () {
                return e || {}
              }),
        'function' != typeof r)
      )
        return r
      if (r._isWrap) return r
      if (
        !r._rollbar_wrapped &&
        ((r._rollbar_wrapped = function () {
          o && 'function' == typeof o && o.apply(this, arguments)
          try {
            return r.apply(this, arguments)
          } catch (o) {
            var e = o
            throw (
              (e &&
                ('string' == typeof e && (e = new String(e)),
                (e._rollbarContext = n() || {}),
                (e._rollbarContext._wrappedSource = r.toString()),
                (window._rollbarWrappedError = e)),
              e)
            )
          }
        }),
        (r._rollbar_wrapped._isWrap = !0),
        r.hasOwnProperty)
      )
        for (var t in r)
          r.hasOwnProperty(t) && (r._rollbar_wrapped[t] = r[t])
      return r._rollbar_wrapped
```

```
            } catch (e) {
              return r
            }
          })
        for (
          var u =
'log,debug,info,warn,warning,error,critical,global,configure,handleUncaughtExc
eption,handleUnhandledRejection,captureEvent,captureDomContentLoaded,captureLo
ad'.split(
              ','
            ),
            f = 0;
          f < u.length;
          ++f
        )
          t.prototype[u[f]] = l(u[f])
        r.exports = { setupShim: a, Rollbar: p }
      },
      function (r, e) {
        'use strict'
        function o(r, e, o) {
          if (r) {
            var t
            'function' == typeof e._rollbarOldOnError
              ? (t = e._rollbarOldOnError)
              : r.onerror &&
                !r.onerror.belongsToShim &&
                ((t = r.onerror), (e._rollbarOldOnError = t))
            var a = function () {
              var o = Array.prototype.slice.call(arguments, 0)
              n(r, e, t, o)
            }
            ;(a.belongsToShim = o), (r.onerror = a)
          }
        }
        function n(r, e, o, n) {
          r._rollbarWrappedError &&
            (n[4] || (n[4] = r._rollbarWrappedError),
            n[5] || (n[5] = r._rollbarWrappedError._rollbarContext),
            (r._rollbarWrappedError = null)),
            e.handleUncaughtException.apply(e, n),
            o && o.apply(r, n)
        }
        function t(r, e, o) {
          if (r) {
            'function' == typeof r._rollbarURH &&
              r._rollbarURH.belongsToShim &&
              r.removeEventListener('unhandledrejection', r._rollbarURH)
            var n = function (r) {
```

```
                var o, n, t
                try {
                  o = r.reason
                } catch (r) {
                  o = void 0
                }
                try {
                  n = r.promise
                } catch (r) {
                  n = '[unhandledrejection] error getting `promise` from
event'
                }
                try {
                  ;(t = r.detail), !o && t && ((o = t.reason), (n =
t.promise))
                } catch (r) {
                  t = '[unhandledrejection] error getting `detail` from event'
                }
                o || (o = '[unhandledrejection] error getting `reason` from
event'),
                  e &&
                    e.handleUnhandledRejection &&
                    e.handleUnhandledRejection(o, n)
              }
              ;(n.belongsToShim = o),
                (r._rollbarURH = n),
                r.addEventListener('unhandledrejection', n)
            }
          }
          function a(r, e, o) {
            if (r) {
              var n,
                t,
                a =
'EventTarget,Window,Node,ApplicationCache,AudioTrackList,ChannelMergerNode,Cry
ptoOperation,EventSource,FileReader,HTMLUnknownElement,IDBDatabase,IDBRequest,
IDBTransaction,KeyOperation,MediaController,MessagePort,ModalWindow,Notificati
on,SVGElementInstance,Screen,TextTrack,TextTrackCue,TextTrackList,WebSocket,We
bSocketWorker,Worker,XMLHttpRequest,XMLHttpRequestEventTarget,XMLHttpRequestUp
load'.split(
                  ','
                )
              for (n = 0; n < a.length; ++n)
                (t = a[n]), r[t] && r[t].prototype && l(e, r[t].prototype, o)
            }
          }
          function l(r, e, o) {
            if (e.hasOwnProperty && e.hasOwnProperty('addEventListener')) {
              for (
```

```
              var n = e.addEventListener;
              n._rollbarOldAdd && n.belongsToShim;

            )
              n = n._rollbarOldAdd
            var t = function (e, o, t) {
              n.call(this, e, r.wrap(o), t)
            }
            ;(t._rollbarOldAdd = n),
              (t.belongsToShim = o),
              (e.addEventListener = t)
            for (
              var a = e.removeEventListener;
              a._rollbarOldRemove && a.belongsToShim;

            )
              a = a._rollbarOldRemove
            var l = function (r, e, o) {
              a.call(this, r, (e && e._rollbar_wrapped) || e, o)
            }
            ;(l._rollbarOldRemove = a),
              (l.belongsToShim = o),
              (e.removeEventListener = l)
          }
        }
        r.exports = {
          captureUncaughtExceptions: o,
          captureUnhandledRejections: t,
          wrapGlobals: a,
        }
      },
      function (r, e) {
        'use strict'
        function o(r, e) {
          ;(this.impl = r(e, this)), (this.options = e), n(o.prototype)
        }
        function n(r) {
          for (
            var e = function (r) {
                return function () {
                  var e = Array.prototype.slice.call(arguments, 0)
                  if (this.impl[r]) return this.impl[r].apply(this.impl, e)
                }
              },
              o =
'log,debug,info,warn,warning,error,critical,global,configure,handleUncaughtExc
eption,handleUnhandledRejection,_createItem,wrap,loadFull,shimId,captureEvent,
captureDomContentLoaded,captureLoad'.split(
                ','
```

```
            ),
            n = 0;
          n < o.length;
          n++
        )
          r[o[n]] = e(o[n])
      }
      ;(o.prototype._swapAndProcessMessages = function (r, e) {
        this.impl = r(this.options)
        for (var o, n, t; (o = e.shift()); )
          (n = o.method),
            (t = o.args),
            this[n] &&
              'function' == typeof this[n] &&
              ('captureDomContentLoaded' === n || 'captureLoad' === n
                ? this[n].apply(this, [t[0], o.ts])
                : this[n].apply(this, t))
        return this
      }),
        (r.exports = o)
    },
    function (r, e) {
      'use strict'
      r.exports = function (r) {
        return function (e) {
          if (!e && !window._rollbarInitialized) {
            r = r || {}
            for (
              var o,
                n,
                t = r.globalAlias || 'Rollbar',
                a = window.rollbar,
                l = function (r) {
                  return new a(r)
                },
                i = 0;
              (o = window._rollbarShims[i++]);

            )
              n || (n = o.handler),
                o.handler._swapAndProcessMessages(l, o.messages)
            ;(window[t] = n), (window._rollbarInitialized = !0)
          }
        }
      }
    },
  ])
  // End Rollbar Snippet
</script>
```

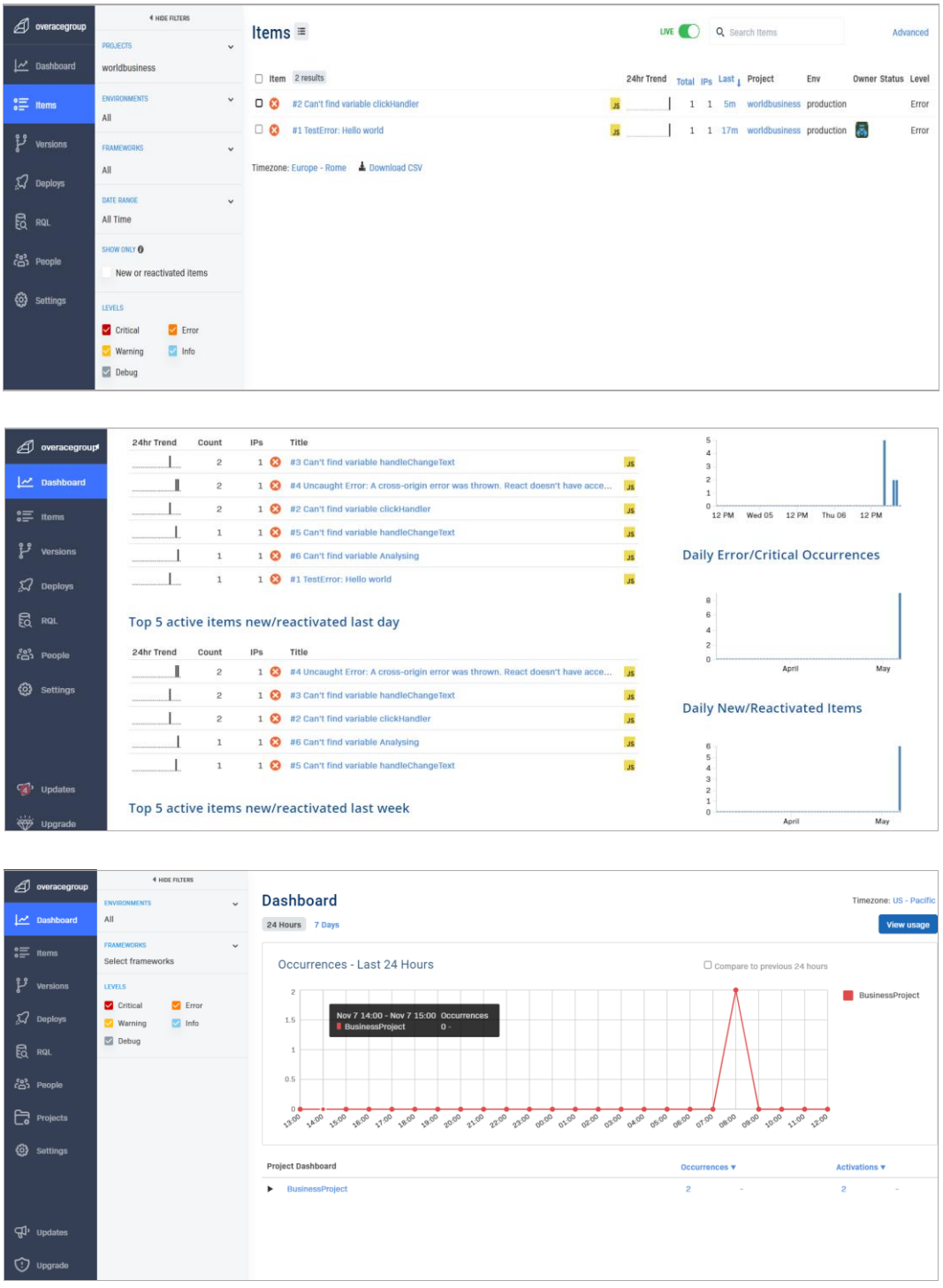


Figure 4.20 Complete view of Rollbar

5. Conclusion and future work

This research gave a comprehensive comparative assessment of this topic, Modern technological and analytical approaches in the monitoring of errors generated by the frontend layers of web applications, in which four free-to-use tools were presented, two of which are used to track and evaluate user behavior on the website, and the other two are used to monitor errors generated in the frontend layers of web applications, with recommendations on how they can increase user satisfaction and the company's reputation.

The system employed in this study was a web application that contained a home page where visitors may scroll around the website and view a business video, as well as subscribe to the website by entering their email address and then clicking on buttons to access the other pages. Users may also shop on the services and marketing sections. By selecting a product, the desired item is added to the product basket, where it may be checked against the list of selected items before buying or canceling the final transaction. Finally, there is a contact us page where new visitors may ask questions and send messages to a member of the organization.

The benefits and drawbacks of employing these strategies are identified on the mentioned website. As a result of the investigation, the finest tools for displaying problems and tracking user activity are sentry and GTM, which are used by the majority of administrators to enhance the quality and performance of their websites.

In this regard, useful findings were acquired using GTM, such as the number of times a person accesses the site, known as a session. The number of transactions they have made, as well as the duration of time the user remains on the site and navigates to other sites and also the users' geographical location was revealed. This is done independently for each user, which was thoroughly explained in Chapter 4.

Furthermore, sentry was one of the most powerful tools for reporting issues, and by using it, the number of transactions and failures associated with each user were collected. (Errors encountered by each individual user). The type of issues and the user's URL and IP address are logged, which the website owner might utilize to solve these errors.

However, this study focused more on monitoring tools, like other investigations, there were certain limitations to this study. First and primarily, because this research was completed on a

website developed by author and not a real website, it was unable to reach a large variety of users. As a result, a small statistical population was analyzed as a sample and as well as based on previous works, and the benefits and drawbacks were examined. Therefore, as the future work, further user analyses would be considered, and more tools will be investigated, such as, TrackJS and Pingdom.

Finally, because the tool comparison was limited to a single website and the programming languages used were JavaScript and React, while other websites can develop in a variety of languages, the specifics and findings of the study are not relevant outside of the study. However, the study's practical information may be used in future research.

References

1. A Study about Online Transactions, prepared for TeaLeaf Technology Inc. by Harris Interactive, October 2005
2. Azim, M., & Hasan, N. (2018, February). Web Analytics Tools Usage among Indian Library Professionals. In *2018 5th International Symposium on Emerging Trends and Technologies in Libraries and Information Services (ETTLIS)* (pp. 31-35). IEEE.
3. Bai, B., Law, R., & Wen, I. (2008). The impact of website quality on customer satisfaction and purchase intentions: Evidence from Chinese online visitors. *International journal of hospitality management*, *27*(3), 391-402.
4. Belair Gagnon, Valerie & Holton, Avery. (2019). The Two Faces of Janus: Web Analytics Companies and the Shifting Culture of News. Journalism Practice. 13. 993-997. 10.1080/17512786.2019.1642132.
5. Burby, J., Brown, A., & WAA Standards Committee. (2007). Web analytics definitions. *Washington DC: Web Analytics Association*.
6. Chyrun, L., Burov, Y., Rusyn, B., Pohreliuk, L., Oleshek, O., Gozhyj, A., & Bobyk, I. (2019). Web Resource Changes Monitoring System Development. In *MoMLeT* (pp. 255-273).
7. Chyrun, L., Gozhyj, A., Yevseyeva, I., Dosyn, D., Tyhonov, V., & Zakharchuk, M. (2019). Web Content Monitoring System Development. In *COLINS* (pp. 126-142).
8. Conrad, S. (2015). Using Google Tag Manager and Google Analytics to track DSpace metadata fields as custom dimensions.
9. Croll, A., & Power, S. (2009). *Complete web monitoring: watching your visitors, performance, communities, and competitors*. " O'Reilly Media, Inc.".
10. Fagan, J. C. (2014). The suitability of web analytics key performance indicators in the academic library environment. *The Journal of Academic Librarianship*, *40*(1), 25-34.
11. Farney, T. (2016). Getting the best Google analytics data for your library. *Library Technology Reports*, *52*(7), 5-8.
12. Farney, T. (2016). Optimizing Google Analytics for LibGuides. *Library Technology Reports*, *52*(7), 26-30.
13. Farney, T. (2016). Using Google tag manager in your library. *Library Technology Reports*, *52*(7), 9-13.
14. Farney, T., & McHale, N. (2013). Introducing google analytics for libraries. *Library technology reports*, *49*(4), 5-8.
15. Fenstermacher, K. D., & Ginsburg, M. (2003). Client-side monitoring for Web mining. *Journal of the American Society for Information Science and Technology*, *54*(7), 625-637.
16. Filipe, R. Â. S. (2020). *Client-Side Monitoring of Distributed Systems* (Doctoral dissertation, 00500: Universidade de Coimbra).
17. Filipe, R., & Araujo, F. (2016, October). Client-side monitoring techniques for web sites. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)* (pp. 363-366). IEEE.

18. Filipe, R., & Araujo, F. (2019, December). Client-Side Monitoring of HTTP Clusters Using Machine Learning Techniques. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)* (pp. 282-286). IEEE.

19. Filipe, R., Paiva, R. P., & Araujo, F. (2017, October). Client-side black-box monitoring for web sites. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)* (pp. 1-5). IEEE.

20. Hootsuite. 2019. Digital 2019 - Essential Insights into How People around the World Use the Internet, Mobile Devices, Social Media, and E-Commerce.

21. Jansen, B. J., Jung, S. G., & Salminen, J. (2020). Data Quality in Website Traffic Metrics: A Comparison of 86 Websites Using Two Popular Analytics Services.

22. Jansen, B. J., Jung, S. G., & Salminen, J. (2020). *Data Quality in Website Traffic Metrics: A Comparison Of 86 Websites Using Two Popular Analystics Services*. Tech Report 2020. Available online: http://www. bernardjjansen. com/uploads/2/4/1/8/24188166/traffic_ analytics_comparison. pdf (accessed on 28 May 2021).

23. Jeong, M., Oh, H., & Gregoire, M. (2003). Conceptualizing web site quality and its consequences in the lodging industry. *International Journal of Hospitality Management*, *22*(2), 161-175.

24. Kiciman, E., & Livshits, B. (2007). AjaxScope: a platform for remotely monitoring the client-side behavior of Web 2.0 applications. *ACM SIGOPS Operating Systems Review*, *41*(6), 17-30.

25. King, A. B. (2008). *Website optimization*. " O'Reilly Media, Inc.".

26. Kinnunen, M. (2020). Evaluating and improving Web performance using free-to-use tools.

27. Kirk, M., Morgan, R., Tonkin, E., McDonald, K., & Skirton, H. (2012). An objective approach to evaluating an internet-delivered genetics education resource developed for nurses: using Google Analytics™ to monitor global visitor engagement. *Journal of Research in Nursing*, *17*(6), 557-579.

28. Ledford, J. L., & Tyler, M. E. (2007). *Google Analytics 2.0*. John Wiley & Sons.

29. Ledford, J. L., Teixeira, J., & Tyler, M. E. (2011). *Google analytics*. John Wiley and Sons.

30. Li, W., & Gorton, I. (2010, October). Analyzing Web Logs to Detect User-Visible Failures. In *SLAML*.

31. Macbeth, S. (2016). Tracking and Online Banking: A Survey.

32. Mehta, N., & Bharadwaj, A. (2015). Knowledge integration in outsourced software development: The role of sentry and guard processes. *Journal of Management Information Systems*, *32*(1), 82-115.

33. O'Brien, P., Young, S. W., Arlitsch, K., & Benedict, K. (2018). Protecting privacy on the web: a study of HTTPS and Google Analytics implementation in academic library websites. *Online Information Review*.

34. Ocariza Jr, F. S., Pattabiraman, K., & Zorn, B. (2011, November). JavaScript errors in the wild: An empirical study. In *2011 IEEE 22nd International Symposium on Software Reliability Engineering* (pp. 100-109). IEEE.

35. Open for Business? Real Availability Is Focused on Users, Not Applications, A TeaLeaf Technology Inc. white paper, October 2003

36. Pertet, S., & Narasimhan, P. (2005). *Causes of failure in web applications* (Vol. 92). Technical Report CMU-PDL-05-109, Carnegie Mellon University.

37. Sawyer, S., Guinan, P. J., & Cooprider, J. (2010). Social interactions of information systems development teams: a performance perspective. *Information Systems Journal*, *20*(1), 81-107.

38. Sentry brings performance monitoring to the developer. (2020, Jul 14). *Business Wire* Retrieved from https://www.proquest.com/wire-feeds/sentry-brings-performance-monitoring-developer/docview/2423467474/se-2?accountid=193930

39. Sentry delivers full suite of application monitoring capabilities to JavaScript developers. (2021, Jan 27). *Business Wire* Retrieved from https://www.proquest.com/wire-feeds/sentry-delivers-full-suite-application-monitoring/docview/2481058214/se-2?accountid=193930

40. Sentry expands capabilities as demand increases for mobile application monitoring. (2020, May 19). *Business Wire* Retrieved from https://www.proquest.com/wire-feeds/sentry-expands-capabilities-as-demand-increases/docview/2404293515/se-2?accountid=193930

41. Song, M. J., Ward, J., Choi, F., Nikoo, M., Frank, A., Shams, F., ... & Krausz, M. (2018). A process evaluation of a web-based mental health portal (WalkAlong) using google analytics. *JMIR mental health*, *5*(3), e50.

42. Trinh, T. D., Vu, T. H. G., & Le, V. M. (2019, December). Browser Extension-based Crowdsourcing Model for Website Monitoring. In *Proceedings of the Tenth International Symposium on Information and Communication Technology* (pp. 465-472).

43. Yeager, H. J. (2017). Using EZproxy and Google Analytics to evaluate electronic serials usage. *Serials Review*, *43*(3-4), 208-215.