****

OF

**TEZPUR UNIVERSITY**

**2020**

**DEVELOPMENT OF FEATURES AND COMPONENTS FOR ECOMMERCE WEBSITES**

*A project report submitted in partial fulfillment of the requirements for the degree*

*of*

**MASTER OF COMPUTER APPLICATION (MCA)**

*Submitted by*

**HIRAK JYOTI NATH (CSM17033)**

*Guided by*

**PRABHU SRIVASTAVA**

CHIEF TECHNOLOGICAL OFFICER

GETPY.BIZ

**Dr. SANGHAMITRA NATH**

ASSISTANT PROFESSOR

TEZPUR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**TEZPUR UNIVERSITY**

TEZPUR 784028

ASSAM.

****

**DEPARTMENT OF**

**COMPUTER SCIENCE & ENGINEERING**

**SCHOOL OF ENGINEERING**

**TEZPUR UNIVERSITY**

**NAPAAM, SONITPUR – 784028**

**ASSAM, INDIA**

**CERTIFICATE**

This is to certify that the report titled **“Development of Features and Components for E-Commerce Websites”** submitted to Tezpur University in the Department of Computer Science & Engineering, in partial fulfillment of the award of the degree of Master of Computer Application, is part of the project work carried out by **Hirak Jyoti Nath (CSM17033)** under my guidance.

**Dr. Sanghamitra Nath**

Assistant Professor

Department of CSE

Tezpur Univeristy

****

**DEPARTMENT OF**

**COMPUTER SCIENCE & ENGINEERING**

**SCHOOL OF ENGINEERING**

**TEZPUR UNIVERSITY**

**NAPAAM, SONITPUR – 784028**

**ASSAM, INDIA**

**CERTIFICATE**

This is to certify that **Hirak Jyoti Nath (CSM17033)** has prepared the project report entitled **“Development of Features and Components for E-Commerce Websites”**. This work has been done under supervision and guidance of **Dr. Sanghamitra Nath**(internal), **Mr. Prabhu Srivastava**(external). This project report is the result of his efforts and sincerity. The project report is found worthy of acceptance for the award of the degree of **Master of Computer Application (MCA)** in **Computer Science and Engineering**.

**Dr. Bhogeswar Borah**

Head of Department

Department of CSE

Tezpur University

****

**DEPARTMENT OF**

**COMPUTER SCIENCE & ENGINEERING**

**SCHOOL OF ENGINEERING**

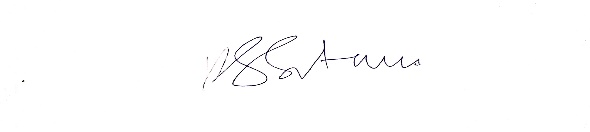
**TEZPUR UNIVERSITY**

**NAPAAM, SONITPUR – 784028**

**ASSAM, INDIA**

**CERTIFICATE**

This is to certify that the project entitled “**Development of Features and Components for E-Commerce Websites”** submitted by **Hirak Jyoti Nath (CSM17033)** in partial fulfillment of the requirements for the completion of the degree of **Master of Computer Applications,** is part of the project work carried out by him under my supervision and guidance.



**Mr. Prabhu Srivastava**

CTO, GetPy.biz

Bengaluru

**PROOF OF INTERNSHIP**

****

|  |  |
| --- | --- |
|  | **DEPARTMENT OF**  **COMPUTER SCIENCE & ENGINEERING**  **TEZPUR UNIVERSITY**  TEZPUR 784028  ASSAM. |

****

**CERTIFICATE**

This dissertation entitled **“Development of Features and Components for E-Commerce Websites”** submitted by **Hirak Jyoti Nath (CSM17033)**  in partial fulfillment of requirements for the degree of Master of Computer Application (MCA) of Tezpur University has been examined.

Internal Examiner External Examiner

Date: Date:

Place: Place:

**ACKNOWLEDGEMENT**

In developing this project, I have received immense help from a number of persons. It gives me tremendous pleasure in sincerely thanking and expressing my gratitude towards my project guides, **Mr. Prabhu Srivastava** (external), **Dr. Sanghamitra Nath** (internal) for their valuable guidance and constant encouragement throughout the project entitled “**Development of Features and Components for E-Commerce Websites**”. Without them, this project would have never been possible. They helped and supported me during the course of the entire project.

I take immense pleasure in acknowledging the effort and time put in, by my guides to get me to this stage, where I have successfully completed my project and am able to put forth this report.

**Hirak Jyoti Nath (CSM17033)**

MCA 6th Semester

Department of CSE

Tezpur University

**CONTENTS**

1. Introduction
   1. About The Organisation
   2. Project Profile
   3. About Development of Features and Components for ECommerce Websites
2. Initial System Study
   1. Problem Statement
   2. Approach
3. System Analysis
   1. Real-Time Information Sharing
   2. Asynchronous use of marketing tags
   3. E-Commerce platform in Wordpress
4. System Requirements Specification
5. System Deign
   1. Webhook in Azure Automation
6. System Implementation
   1. Webhook Implementation
   2. Installing Google Tag Manager in Wordpress Site
   3. Implementing WooCommerce Store
7. Conclusion
8. Bibliography

**1. INTRODUCTION**

**1.1 About The Organisation**

GetPY is a company based in Bengaluru. GetPy provides Sales & Marketing Intelligence Platform for businesses. It primarily provides CRM, ERP and vendor management systems, along with printing data at billing counters to provide intuitive, meaningful analytics to SMEs (Small and Medium Enterprises). GetPy has clients from various verticals like apparel, fashion, jewellery, restaurants, electronics stores, and automobiles. Currently, GetPY has over 100 clients in the UAE and 500-plus stores across India.  
  
GetPy uses CRM (Customer Relationship Management) software to perform [data analysis](https://en.wikipedia.org/wiki/Data_analysis) about customers' history with a company, to improve business relationships with customers, specifically focusing on [customer retention](https://en.wikipedia.org/wiki/Customer_retention) and ultimately driving [sales](https://en.wikipedia.org/wiki/Sales) growth. GetPy uses ERP (Enterprise Resource Planning) software to manage day-to-day business activities and make predictions on business insights.

**1.2 Project Profile**

|  |  |
| --- | --- |
| * Project Title : | “Development of Features and Components for E-Commerce Websites” |
| * Organisation : | GetPy.biz |
| * Duration : | 4 months |
| * External Guide : | Prabhu Srivastava  CTO, GetPy.biz |
| * Internal Guide : | Sanghamitra Nath,  Assistant Professor,  Department of ComputerScience & Engineering,  Tezpur University |
| * Project Trainee : | Hirak Jyoti Nath (CSM17033),  MCA 6th Semester, 2020,  Tezpur University |
| * Software Requirements : | PowerShell, CSS, PHP, Javascript |

**1.3 Introduction to Development of Features and Components for ECommerce Websites**

Ecommerce, also known as electronic commerce or internet commerce, refers to the buying and selling of goods or services using the internet, and the transfer of money and data to execute these transactions. Electronic commerce relies heavily on features such as [mobile commerce](https://en.wikipedia.org/wiki/Mobile_commerce), [internet marketing](https://en.wikipedia.org/wiki/Online_advertising), [online transaction processing](https://en.wikipedia.org/wiki/Online_transaction_processing), [inventory management systems](https://en.wikipedia.org/wiki/Inventory_management_software), automated [data collection](https://en.wikipedia.org/wiki/Data_collection) systems etc. With the internet becoming an essential requirement of everyday life, ecommerce has certainly become the most essential element of any business. The project titled **“Development of Features and Components for E-Commerce Websites”** is mainly based on three components:

**Webhook:** A webhook is a way for an application to provide other applications with real-time information.Webhooks are user defined HTTP callbacks which are triggered by some specific events. For example, such as pushing code to a repository or a comment being posted to a blog. Whenever that trigger event occurs in the source site, the webhook collects the data, and sends it to the URL specified in the form of an HTTP request.

**Google Tag Manager:** Google Tag Manager is a free tool that allows us manage and deploy marketing tags (snippets of code or tracking pixels) on our website (or mobile app) without having to modify the code.

**WooCommerce:** WooCommerce is an open source e-commerce plugin for WordPress. WooCommerce can create a basic online store and sell a variety of products and services that include physical goods, digital goods, affiliate transactions etc. WooCommerce adds in basic product management, order processing, and shopping cart functionality. The core software can manage essential ecommerce features like inventory management, sales tax management, shipping cost calculations, and coupons.

**2. INITIAL SYSTEM STUDY**

**2.1 Problem Statement**

**Real-Time Information Sharing:** There are many companies such as BigCommerce, Shopify, Magento, CS-Cart etc., that provides platforms for creating an online store. My first task was to extract real time information from BigCommerce & Shopify API. I was assigned to work with Singer[1], an open source standard for moving data between databases, web APIs, files, queues etc. The Singer uses a data extraction script called “**Tap**” and data loading script called “**Target**” to communicate using a standard JSON-based data format over standardoutput. The Singer Shopify tap or BigCommerce tap pulls data out and delivers data in a standard, JSON-based format with the help of Python Programming. But storing data (push) to a particular database requires the involvement of a third party called Stitch. As Shopify/BigCommerce are ecommerce platform providers that has a certain cost associated, involvement of another third-party Stitch makes this process quite costly. Also pulling of data through taps at frequent intervals is not an efficient way of sharing fresh, updated data (also known as a type of **polling** method).

### **Use of several marketing tags severely affects the website load time:** Information from one data source (our website) is shared with other data sources such as Google Analytics, Adwords, Facebook pixels, Twitter, Bing Ads etc with the help of their respective tags. Tags are snippets of code, which are added to a site to collect information and send it to third parties. The issue with traditional tracking tags is that if they fire synchronously, they can slow down site speeds.

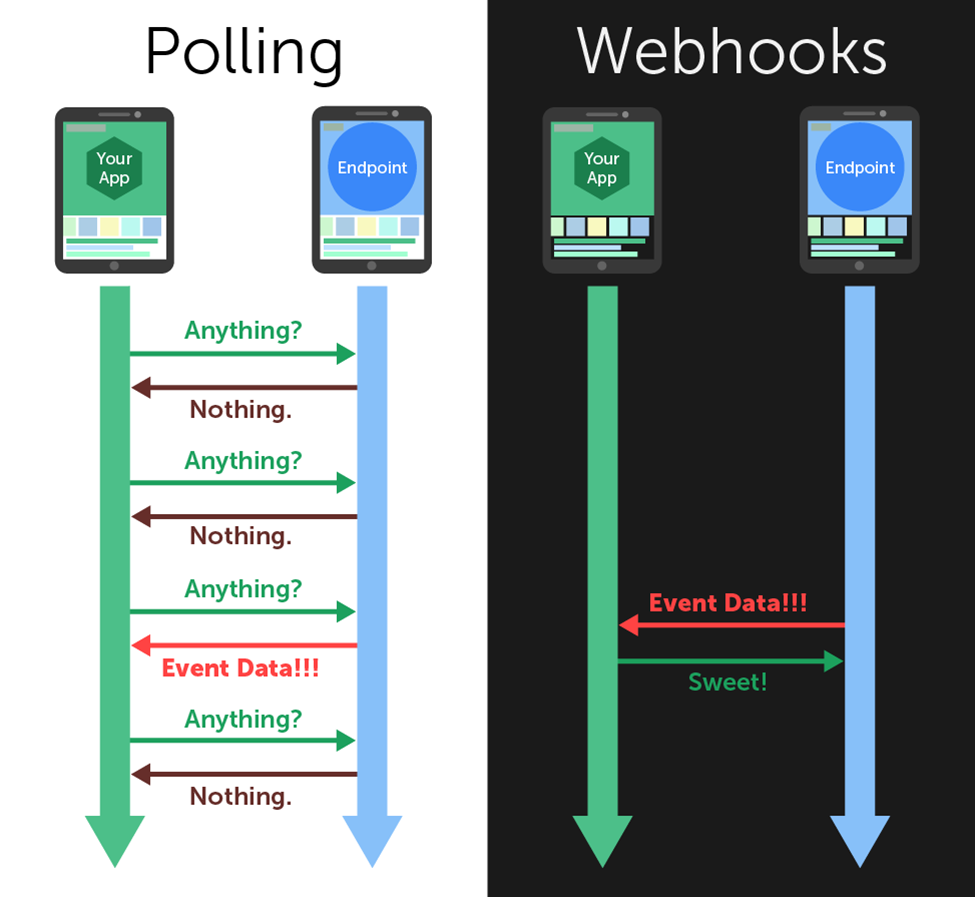
**2.2 Approach**

1. Instead of extracting data with polling methods, which needs to send repeated request to the source site at a frequent interval for any changes in events, I tried to shift my focus to webhooks.
2. Incase of online marketing tags, we tried to use a central tag manager called Google Tag Manager that will take care of all the tracking codes without affecting our website. We can use infinite number of tracking codes, be it of social media such as Facebook, Instagram, Twitter or of search engine such as Google Ads, Bing etc.

**3. SYSTEM ANALYSIS**

**3.1 Real-Time Information Sharing**

The ultimate goal of any API (Application Programming Interface) integration is the efficient sharing of data between apps to provide greater value to users. In order to facilitate this, an integration must provide a method to detect changes, *events*, which occur in the endpoint application. There are two ways **apps can communicate** with each other to share information: ***polling***and ***webhooks***.



**Fig: Diagramatic Representation of Polling and Webhook**

**Polling:** Polling is a process in which repeated requests are send to the endpoint application for any changes in events (for eg: Create, Retrieve and Delete events) at a predetermined frequency (for ex: after every 12 hours) and wait for the endpoint to respond. If there are no changes in events in endpoint application, it will only send the same data that we have received in the previous request.

**Webhooks:** Instead of sending repeated requests for new events, we provide the endpoint with a URL. Whenever a new event occurs within the endpoint app, it posts the event data to our specified URL, updating our application in real-time. We can specify the event for which the webhook needs to be created and also a secret key to authenticate or validate the webhook. The URL is called webhook endpoint which belongs to the receiving system.

**Webhooks are superior to polling in terms of freshness of data, efficiency of communication and infrastructure costs.** A communication standpoint, Zapier did a study across 30 million poll requests made through their service, and found that 98.5% of [polls are wasted](http://resthooks.org/) and they spent [66x more resources](https://zapier.com/engineering/introducing-resthooksorg/) on polling.

### **3.2 Asynchronous use of marketing tags**

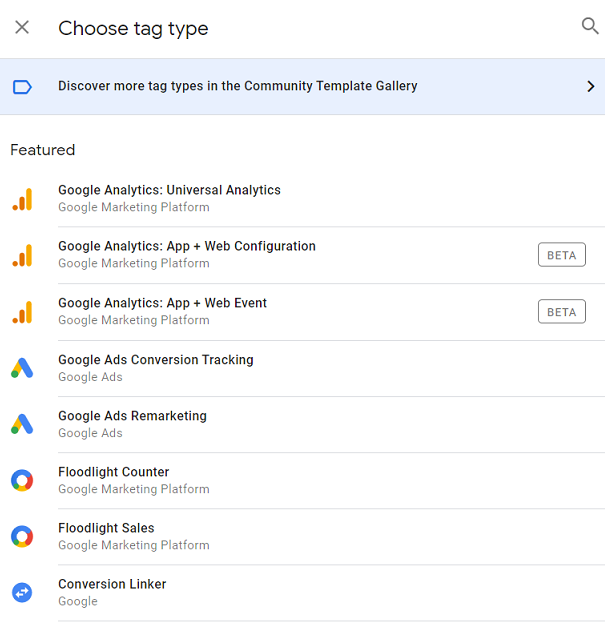
### When tags fire synchronously, one tag being slow to load slows down all the other tags that are waiting on it. And the longer a site takes to load, the more likely it is that people will leave without converting. **Google Tag Manager** (**GTM**) becomes very handy when we have lots of tags to manage because all of the code is stored in one place**. GTM** fire tags asynchronously, so they load independently from one another and donot slow down our pages. Pages can load faster and data collection can start sooner.

### There are three main parts to Google Tag Manager:

**Tags:** Tags are snippets of code or tracking pixels from third-party tools. These tags tell Google Tag Manager ***what*** to do.

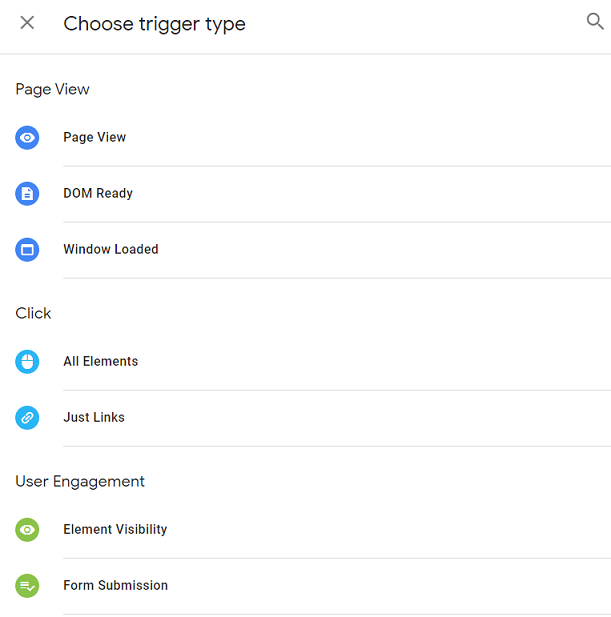
Examples of common tags within Google Tag Manager are:

* Google Analytics Universal tracking code
* Adwords Remarketing code
* Adwords Conversion Tracking code
* Facebook pixels



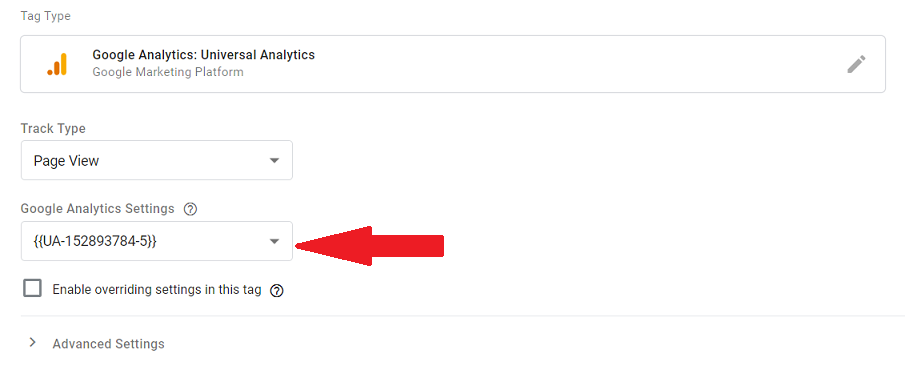
**Fig: tags in Google Tag Manager**

**Triggers:** Triggers are a way to fire the tag that we set up. They tell Tag Manager when we want to fire tags on our site, either on a page view or on a link click.

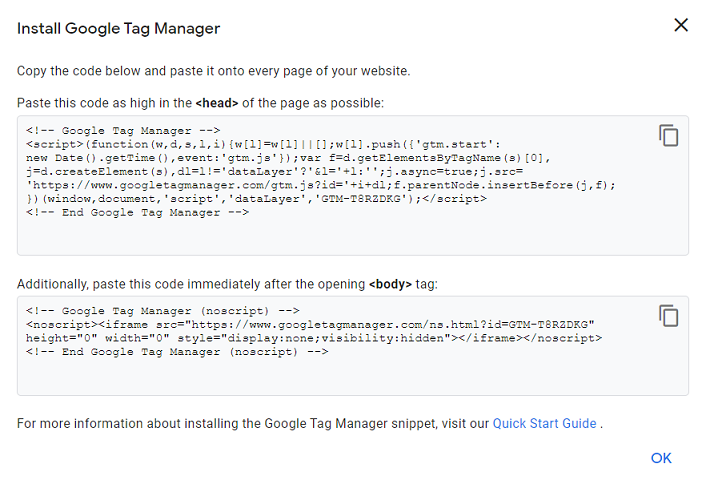


**Fig: triggers in Google Tag Manager**

**Variables:** Variables are additional information that GTM may need for our tag and trigger to work. Here are some examples of different variables.



**Fig: variable in Google Tag Manager**

****

**Fig: Google Tag Manager Tracking code**

**Advantages of Google Tag Manager:**

* It help our site load faster even after using many tags on our site.
* It works with non-Google products.
* Flexibility to play around and test out almost anything we want.
* All third-party code is in one place.
* GTM has a preview and debug mode so we can see what’s working and what’s not before we make anything live. It shows us what tags are firing on the page.

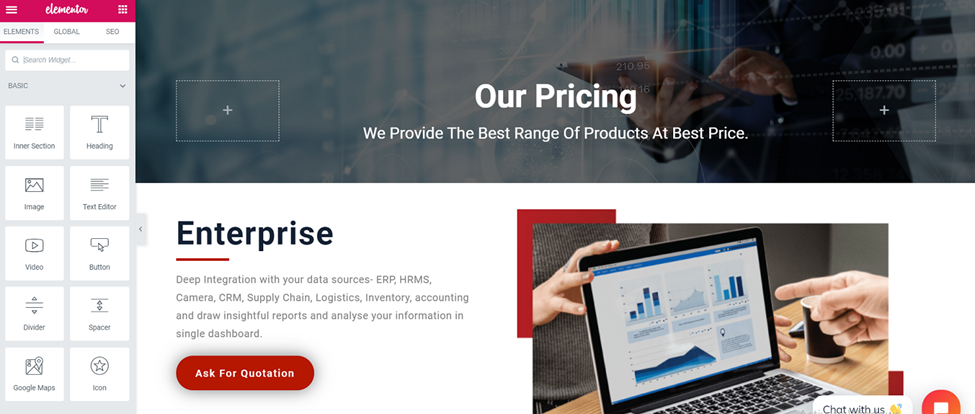
## **What can we track in GTM?**

* Events (link clicks, PDF downloads, add to cart click, remove from cart click)
* Scroll tracking
* Form abandonment
* Shopping cart abandonment
* Video views tracking
* All exit link clicks

**3.3 E-Commerce platform in Wordpress**

**WordPress** (**WordPress.org**) is a free and open-source content management system (CMS) written in PHP and paired with a MySQL or MariaDB database. Open-source means anyone can use or modify the WordPress software for free. **WordPress is the simplest, most popular way to create our own website or blog.**In fact, WordPress powers over 35.2% of all the websites on the Internet. The end result is that WordPress makes building a website accessible to anyone, even people who aren’t developers. Some of important components of Wordpress that I have used for designing are as follows:

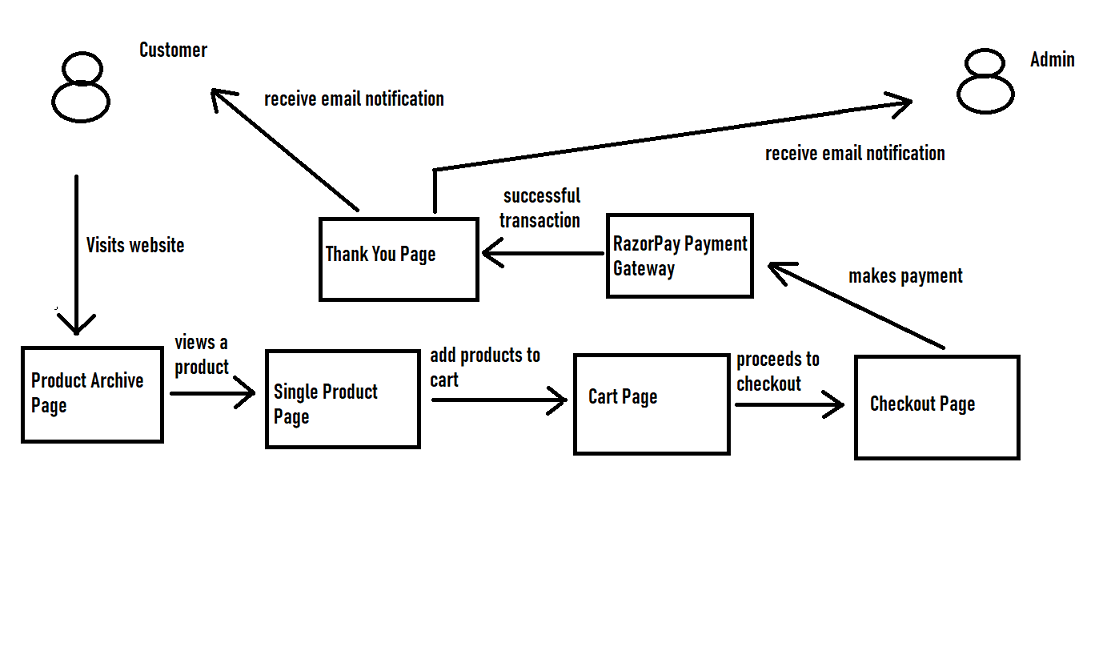
**Elementor:** Elementor is a drag-and-drop page builder for WordPress. This plugin helps us to create beautiful pages using a visual editor. It is designed to build dynamic websites quickly. The user interface is extremely friendly, so it only takes a few minutes to get the hang of it.



**Fig: Pricing Page of GetPy.biz website built using Elementor**

**Woo-Commerce:** By using the Woo-Commerce plugin I have designed the product archive page, single product page , cart page , checkout page and a thank you page, directed after a successful transaction.

**Razorpay plugin for WooCommerce:** Integrating an WooCommerce site with Razorpay allows us to accept payments via the Razorpay Payment Gateway.Razorpay is a payments solution which allows online businesses to accept, process and disburse digital payments through several payment modes like debit cards, credit cards, net banking, UPI and prepaid digital wallets.

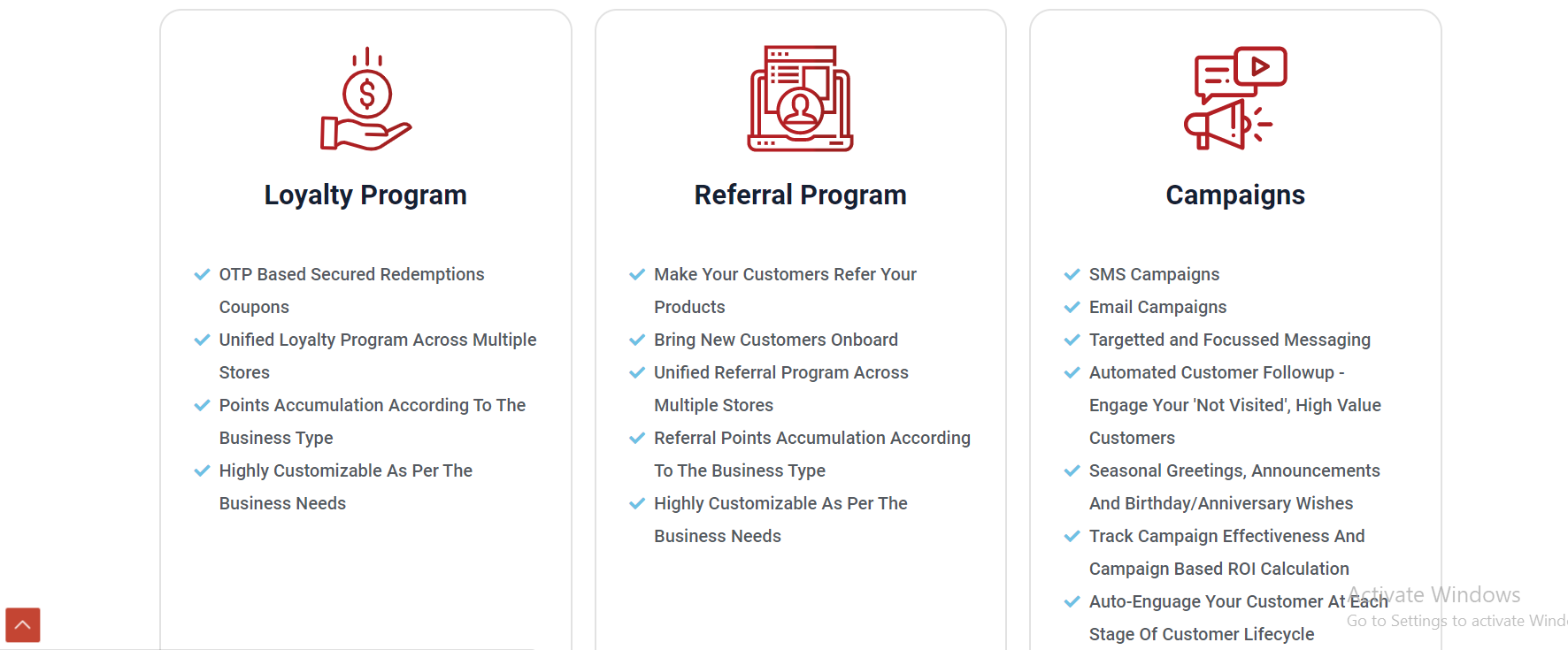


**Fig: control flow diagram of GetPy.biz website**

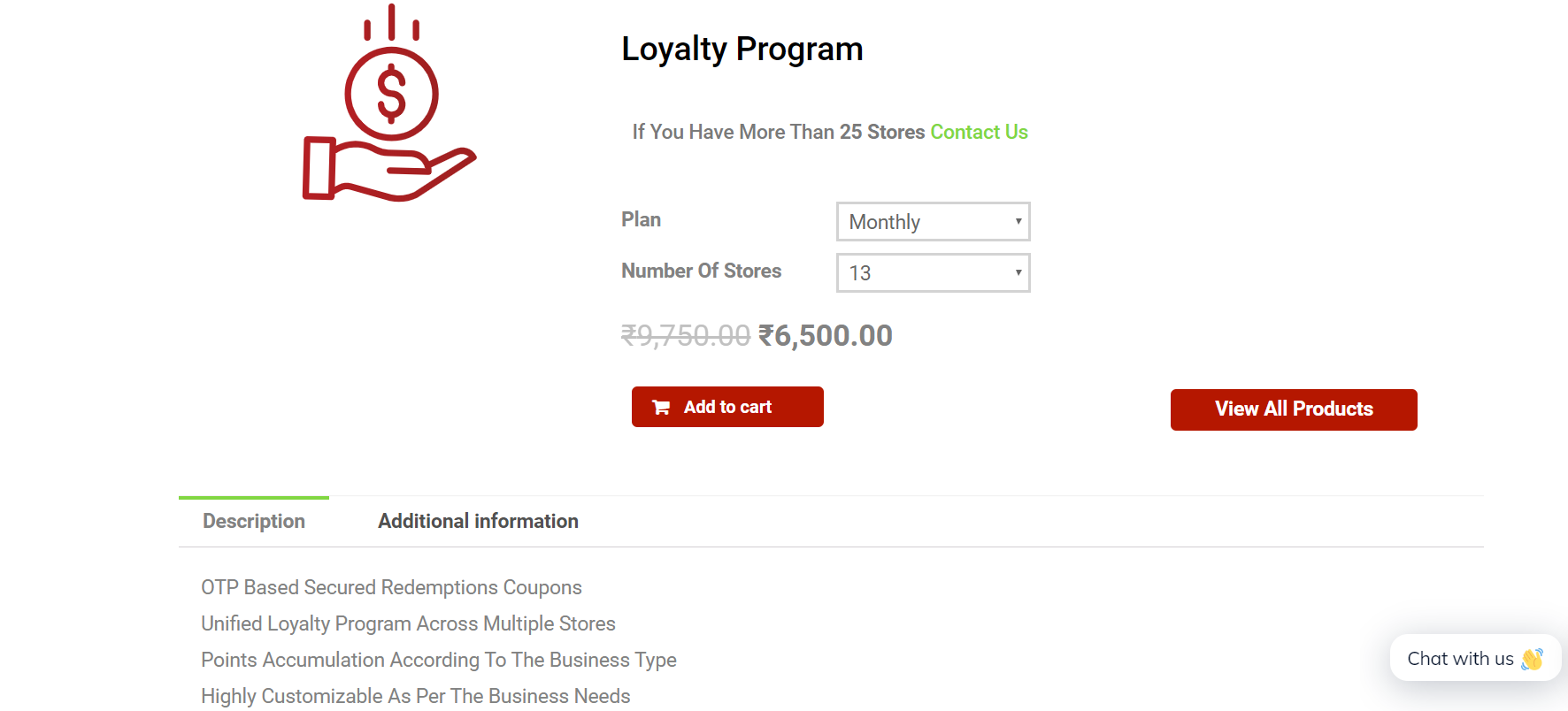
**The control flow of GetPy.biz website can be explained as follows:**

A Customer visits our website. If he wants to see our products, he will go to Product Archive Page. In the Product Archive Page, he can view all the products. If he is interested in our products, he will select a particular product, which will lead him to Single Product Page. In Single Product page, he will see product specification, cost price and add to cart button. Once the product is added to cart, he can move to Cart Page to view existing the products in the cart and the total cost. He can proceed to Checkout page to fill his personal details and shipping address. If the customer clicks on make payment button, he will be directed to the Razorpay payment gateway and once the transaction is successful, he will be redirected to a Thank You page. Once the transaction is successful, the customer will receive an email notification about processing of order. Also Wordpress Admin will receive email notification about a new order being received. Below I have shared screenshots of the webpages.

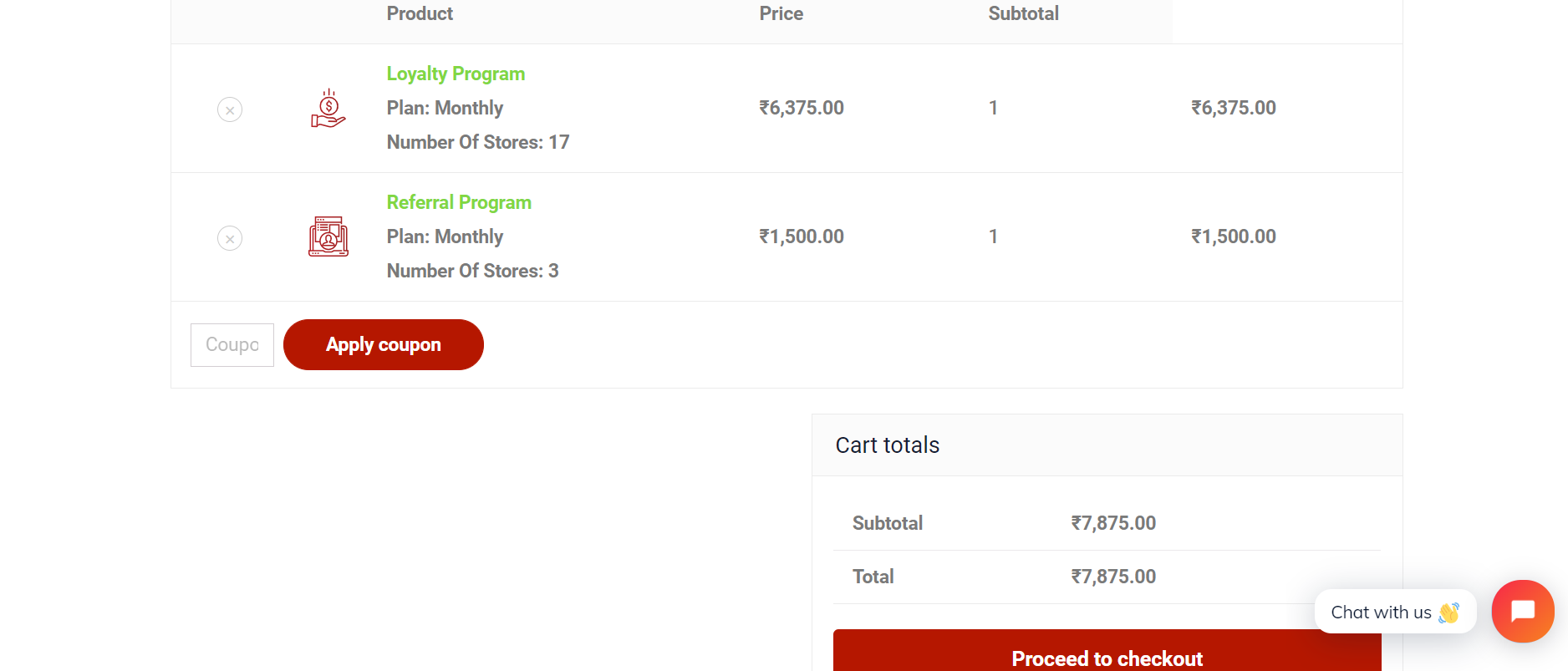
1. **Product Archive Page of GetPy.biz created using Elementor and WooCommerce**



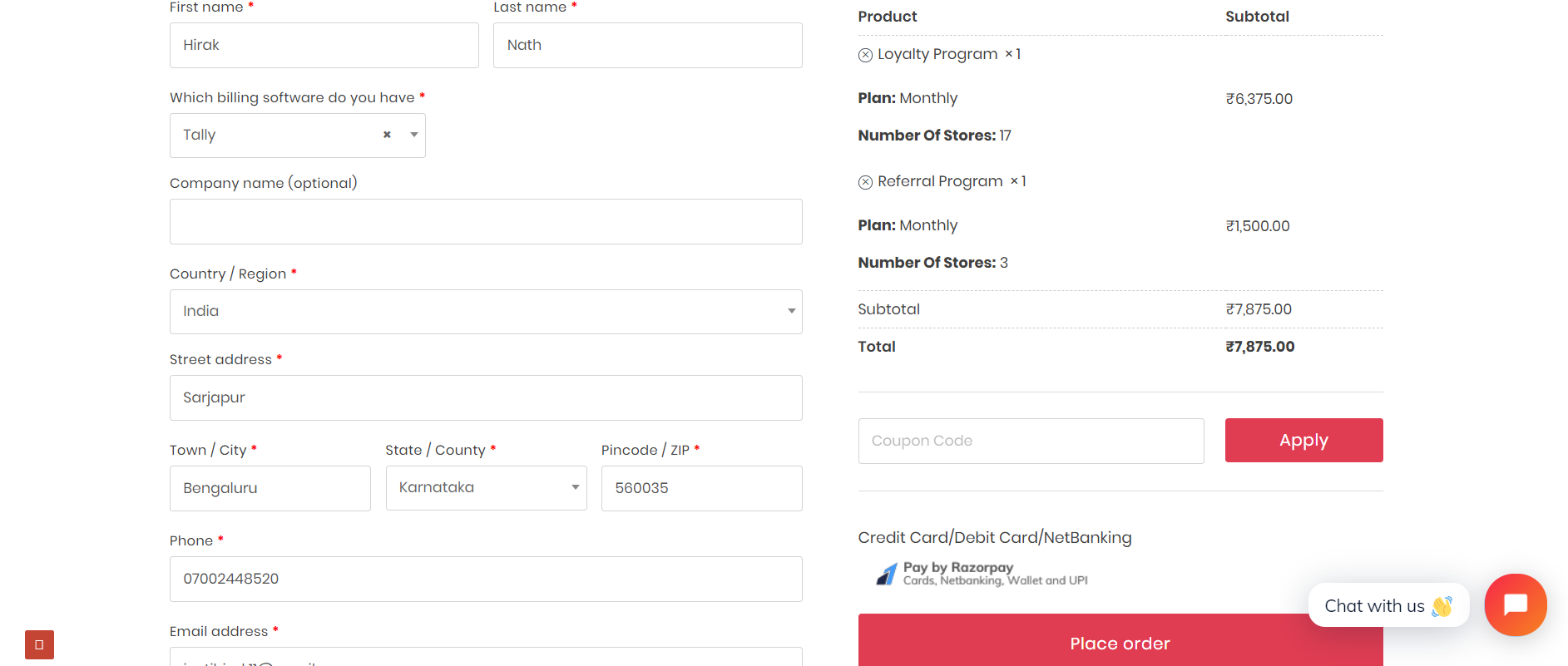
1. **Single Product Page**



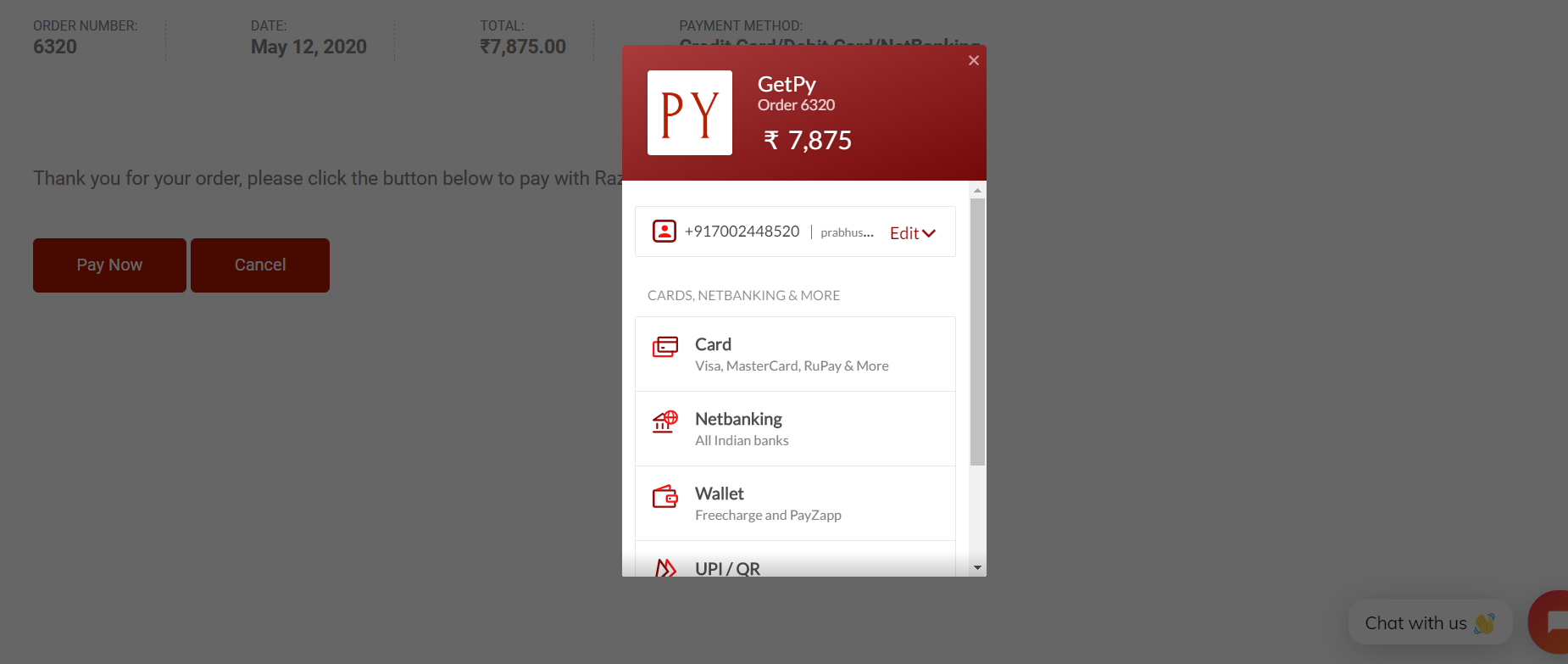
1. **Cart Page**

****

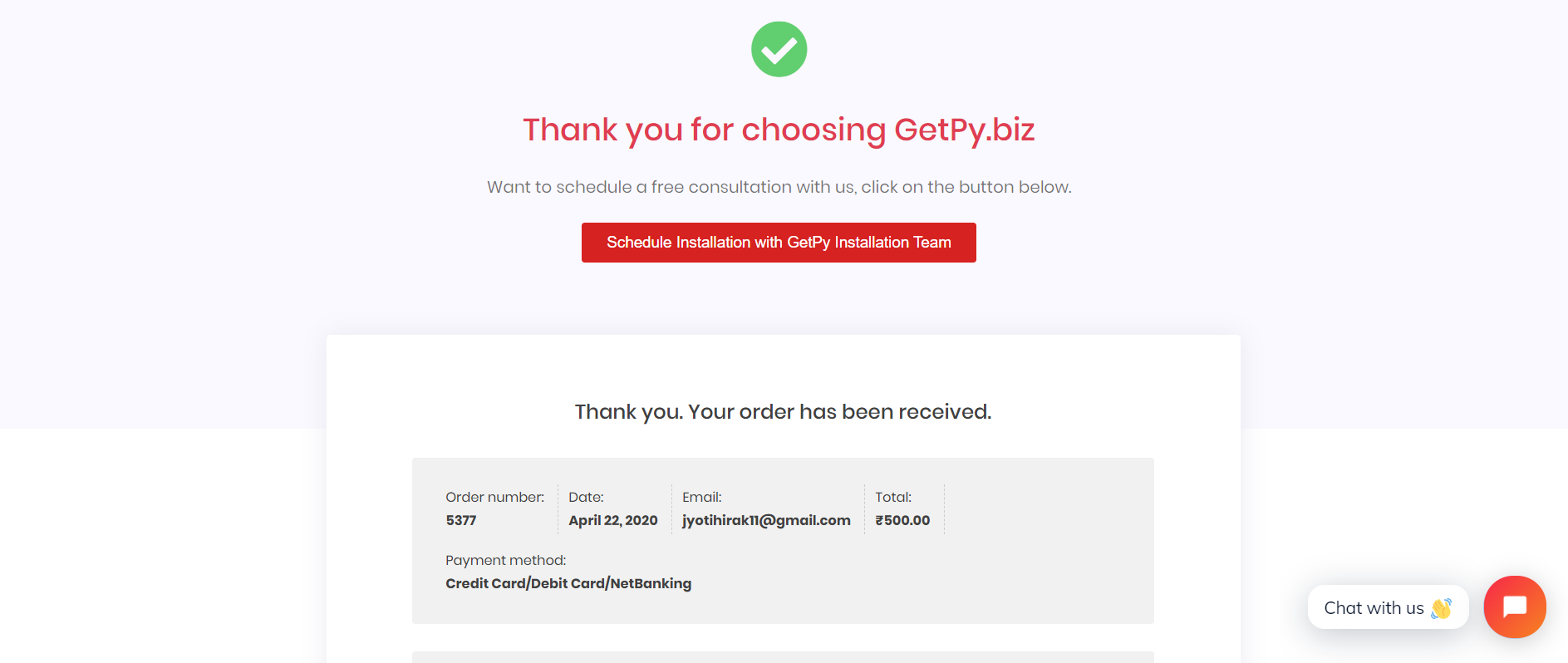
1. **Checkout Page**

****

1. **RazorPay Payment Gateway**

****

1. **Thank You Page**

****

**4. SYSTEM REQUIREMENTS SPECIFICATIONS**

1. With the use of **Webhooks**, our application will receive the updated data in real-time from **BigCommerce/Shopify API** which will provide great value to our users.
2. By using **Google Tag Manager** we can promote our products in social medias such as Instagram, Facebook, Twitter, Linkedin or search engines such as Google Ads, Bing Ads etc as using several tracking codes simultaneously will not degrade our website performance. GTM gives a great boost in Internet Marketing of our ecommerce store.
3. By using **WooCommerce** we can create an ecommerce store in our **Wordpress** site. It is highly cost-effective and can target customers around the world.

**5. SYSTEM DESIGN**

**5.1 Webhook in Azure Automation**

Azure is Microsoft’s cloud computing infrastructure and platform designed to build, deploy and manage different applications and services through a giant network of data centres managed by Microsoft. To implement webhook in Microsoft Azure we need to understand the following components of Azure:

**Azure Automation Account**: Azure Automation delivers a cloud-based automation and configuration service that supports consistent management across Azure and non-Azure environments. It comprises process automation, configuration management, update management, shared capabilities, and heterogeneous features. Out of all these features, I have used the process automation. Automation gives us complete control during deployment, operations, and decommissioning of workloads and resources.

**Process Automation:** Process Automation in Azure Automation allows us to automate frequent, time-consuming, and error-prone cloud management tasks. By reducing errors and boosting efficiency, it also helps to lower our operational costs. Process automation supports the integration of Azure services and other public systems required in deploying, configuring, and managing our end-to-end processes. The service allows us to author [runbooks](https://docs.microsoft.com/en-us/azure/automation/automation-runbook-types) graphically, in PowerShell, or using Python.

**Runbook:** Runbook is a set of tasks that perform some automated process in Azure Automation. It may be a simple process such as starting a virtual machine and creating a log entry, or we may have a complex runbook that combines other smaller runbooks to perform a complex process across multiple resources or even multiple clouds and on-premises environments.

For example, we might have an existing manual process for truncating a SQL database if its approaching maximum size that includes multiple steps such as:-

1. Connecting to the server
2. Connecting to the database
3. Get the current size of database
4. Check if the threshold has exceeded and then truncate it and notify user.

Instead of manually performing each of these steps, we could create a runbook that would perform all of these tasks as a single process. We would start the runbook, provide the required information such as the SQL servername, database name and recipient email and then sit back while the process completes.

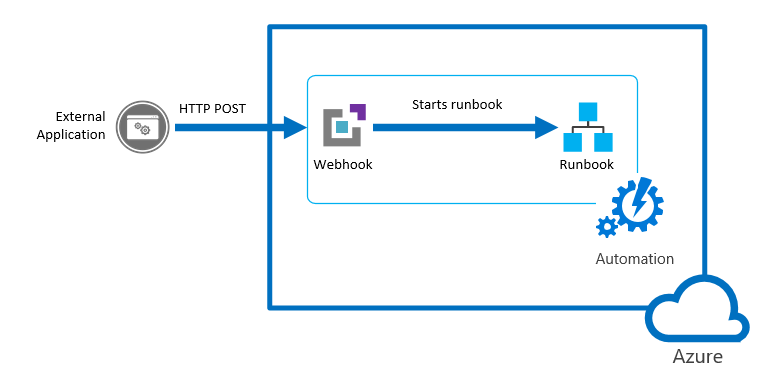
Types of runbook:

* **Graphical**: based on Windows PowerShell, created and edited completely in graphical editor in Azure portal.
* **PowerShell:** Text runbook based on Windows PowerShell script.
* **Python:** Text runbook based on Python

Runbook can be launched by different triggers such as:

* Schedule
* WebHook
* Azure Alert
* Logic Apps
* PowerShell
* Another Runbook
* Watcher Tasks

# **Starting a runbook from a webhook:** A webhook allows an external service to start a particular runbook in Azure Automation through a single HTTP request. External services include Azure DevOps Services, GitHub, Azure Monitor logs, and custom applications. Such a service can use a webhook to start a runbook without implementing the full Azure Automation API.



**Fig: Workflow Diagram of Webhook in Azure Automation**

## Parameters used when the webhook starts a runbook

## 

**Fig: WebhookData Parameter Properties**

The WebhookData parameter has the following properties:

|  |  |
| --- | --- |
| **Property** | **Description** |
| WebhookName | Name of the webhook. |
| RequestHeader | Hashtable containing the headers of  the incoming POST request. |
| RequestBody | Body of the incoming POST request. This body retains any data formatting, such as string, JSON, XML, or form-encoded. The runbook must be written in such a way that is expected to work with the data format. |

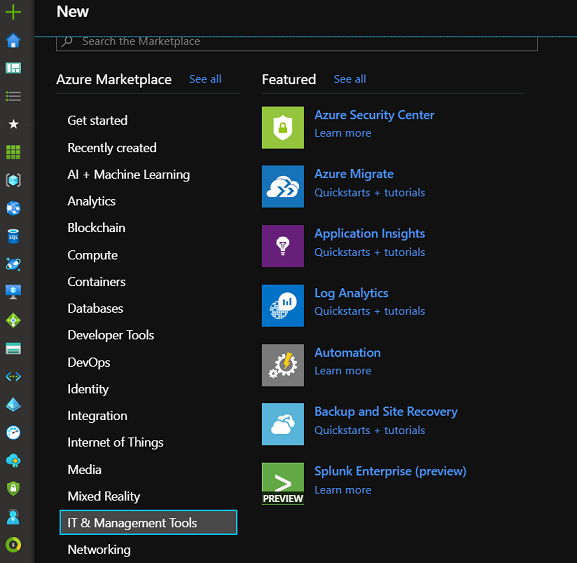
**6. SYSTEM IMPLEMENTATION**

**6.1 Webhook Implementation**

Following are the steps involved to implement webhook in Micosoft Azure.

Create a new Automation Account from the Azure portal:

1. Azure Portal 🡪 +New 🡪It & Management Tools 🡪 Automation

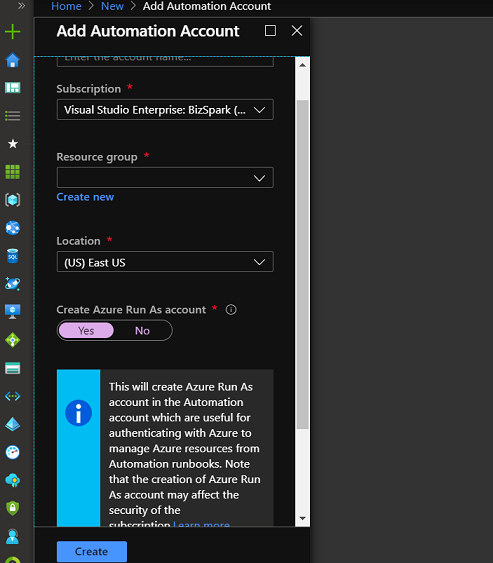
****

1. Name =”MyDemo”… , Create Azure Run As account=Yes
2. Create

When we create an automation account in the Azure portal, we automatically create two authentication entities:-

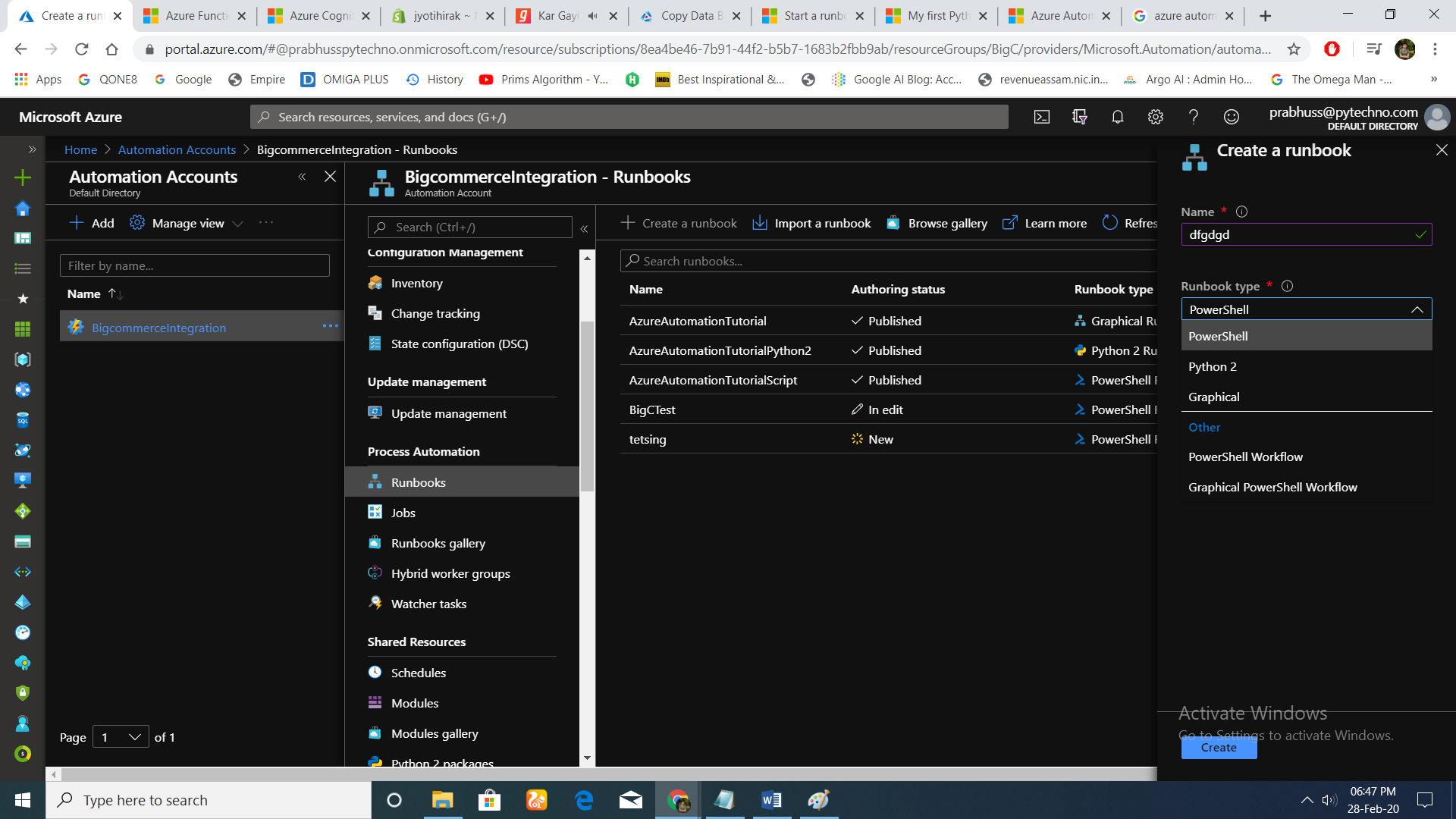
**A Run As account**: This account creates a service principalin Azure Active Directory (Azure AD) and a certificate. It also assigns the Contributor role-based access control (RBAC), which manages Resource Manager resources by using runbooks.

**A Classic Run As account:** This account uploads a management certificate, which is used to manage classic resources by using runbooks.



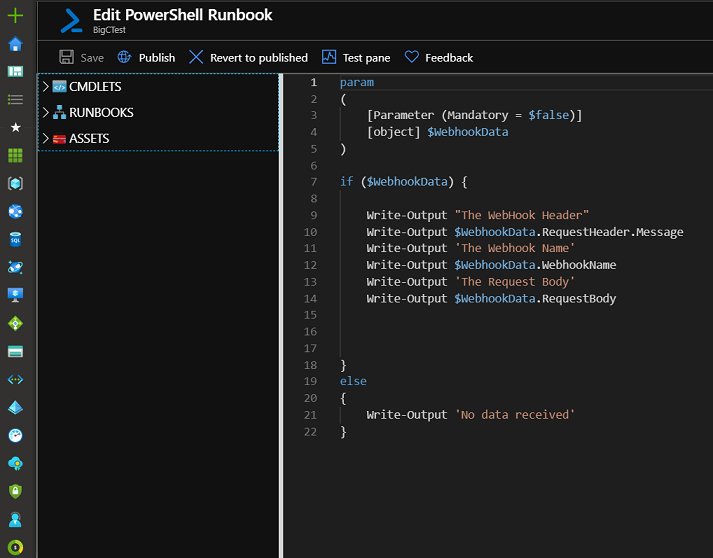
Create a new Azure Automation runbook with Azure Portal:

1. Azure Portal 🡪Automation Account 🡪Select Account 🡪Runbooks
2. Add a runbook 🡪 Create a new runbook
3. Name=”TestRun”, Type=”PowerShell” 🡪Create



Write code for runbook:

1. Select runbook
2. Go to “edit” option
3. Inside the Edit PowerShell Runbook write the mandatory parameters
4. Select the “Publish” option after the coding has finished to save the changes.



**Sample code used for my runbook in PowerShell script: (for implementing webhook)**

param

(

[Parameter (Mandatory = $false)]

[object] $WebhookData

)

if ($WebhookData) {

Write-Output "The WebHook Header"

Write-Output $WebhookData.RequestHeader.Message

Write-Output 'The Webhook Name'

Write-Output $WebhookData.WebhookName

Write-Output 'The Request Body'

Write-Output $WebhookData.RequestBody

}

else

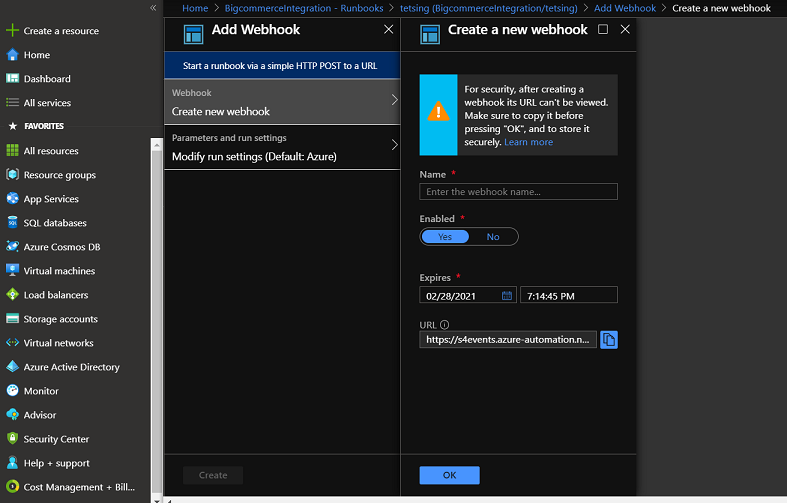
{

Write-Output 'No data received'

}

**Create a webhook for a particular application:** Runbook is a set of tasks that perform some automated process in Azure Automation. It can be also triggered by a webhook. For adding webbook to a particular runbook follow the following steps:

1. Home 🡪 Automation Accounts, select the automation account
2. In “Process Automation” section, select the “Runbooks” option
3. Select a particular runbook or create one using the “+Create a runbook” option
4. Incase we create a new runbook go to “edit” option and publish the Edit PowerShell Runbook window
5. Add webhook 🡪 Create new webhook
6. Name=”Testhook”…, set the expiry date
7. Click the copy icon and press Ctrl+C to copy the URL of the webhook. Then record it in a safe place. **Once we create the**
8. **webhook, we cannot retrieve the URL again.**
9. Click “Ok”

****

**Link Shopify App to Azure Automation runbooks with a webhook:**

**Follow the steps:**

**1,** From our Shopify admin, go to **Settings** > **Notifications**.

**2.** Scroll down to the **Webhooks** section.

**3.** Click the **Create webhook** button. A new window should appear.

**4.** From the first drop-down, select the "Event" from the following list:

* + **Cart creation**: a webhook will be sent every time a cart is created.
  + **Cart update**: a webhook will be sent every time a cart is updated.
  + **Checkout creation**: a webhook will be sent every time a checkout is created.
  + **Checkout deletion**: a webhook will be sent every time a checkout is deleted.
  + **Checkout update**: a webhook will be sent every time a checkout is updated.
  + **Collection creation**: a webhook will be sent every time a collection is created.
  + **Collection deletion**: a webhook will be sent every time a collection is deleted.
  + **Collection update**: a webhook will be sent every time a collection is updated.
  + **Customer group creation**: a webhook will be sent every time a customer group is created.
  + **Customer group deletion**: a webhook will be sent every time a customer group is deleted.
  + **Customer group update**: a webhook will be sent every time a customer group is updated.
  + **Customer creation**: a webhook will be sent every time a new customer is created.
  + **Customer deletion**: a webhook will be sent every time a customer record is deleted.
  + **Customer disable**: a webhook will be sent every time a customer account is disabled.
  + **Customer enable**: a webhook will be sent every time a customer account is enabled.
  + **Customer update**: a webhook will be sent every time a customer record is updated.
  + **Draft order creation**: a webhook will be sent every time a draft order is created.
  + **Draft order deletion**: a webhook will be sent every time a draft order is deleted.
  + **Draft order update**: a webhook will be sent every time a draft order has been updated.
  + **Fulfillment creation**: a webhook will be sent every time a fulfillment is created on an order.
  + **Fulfillment update**: a webhook will be sent every time fulfillment is updated on an order.
  + **Order cancellation**: a webhook will be sent every time an order is canceled.
  + **Order creation**: a webhook will be sent every time an order is created.
  + **Order deletion**: a webhook will be sent every time an order is deleted.
  + **Order fulfillment**: a webhook will be sent every time an order is fulfilled.
  + **Order payment**: a webhook will be sent every time an order has been paid.
  + **Order update**: a webhook will be sent every time an order has been updated.
  + **Product creation**: a webhook will be sent every time a product has been created.
  + **Product deletion**: a webhook will be sent every time a product has been deleted.
  + **Product update**: a webhook will be sent every time a product has been updated.
  + **Refund create**: a webhook will be sent every time a refund is issued.
  + **Shop update**: a webhook will be sent every time a [shop property](https://help.shopify.com/api/reference/store_properties/shop) is updated.
  + **Theme creation**: a webhook will be sent every time a theme is created.
  + **Theme deletion**: a webhook will be sent every time a theme is deleted.
  + **Theme update**: a webhook will be sent every time a [theme property](https://help.shopify.com/api/reference/online_store/theme/) is updated.

1. From the second drop-down, choose the format in which we would like Shopify to send this information to you. The two choices are JSON or XML.
2. Under URL, enter the URL where we would like data to be stored. It is important to note that webhooks **cannot** be returned to the following URLS:
   1. Localhost
   2. Any URL ending in the word "internal" (i.e. thisshop.com/internal)
   3. "Fake" domains like www.example.com
   4. Shopify domains (i.e. shopify.com and myshopify.com)
3. Click **Add webhook**. We webhook should now appear under the "Webhooks" section.
4. We also have the ability to test our webhook, to make sure the information we want is being sent to the correct URL. To do this click the *send test notification* link. Verify at the URL that we specified that this works.

**6.2 Installing Google Tag Manager in Wordpress website**

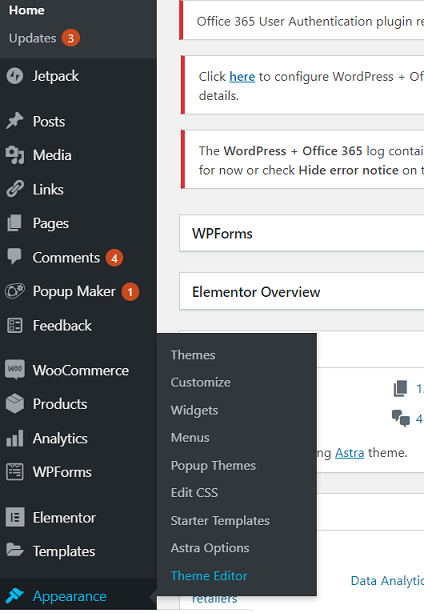
Following are the steps taken to install GTM:

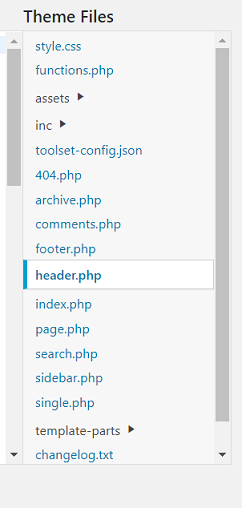
**Step1: Go to dashboard page(/wp-admin/)**

**Step2: Go to Appearance tab 🡪Theme Editor**

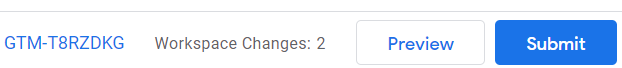
**Step3: Inside header.php paste the tracking code, one just below the head tag and other just below the body tag.**

We can check the changes by loading the homepage🡪left click🡪 view page source to see whether the GTM code has been pasted on our site

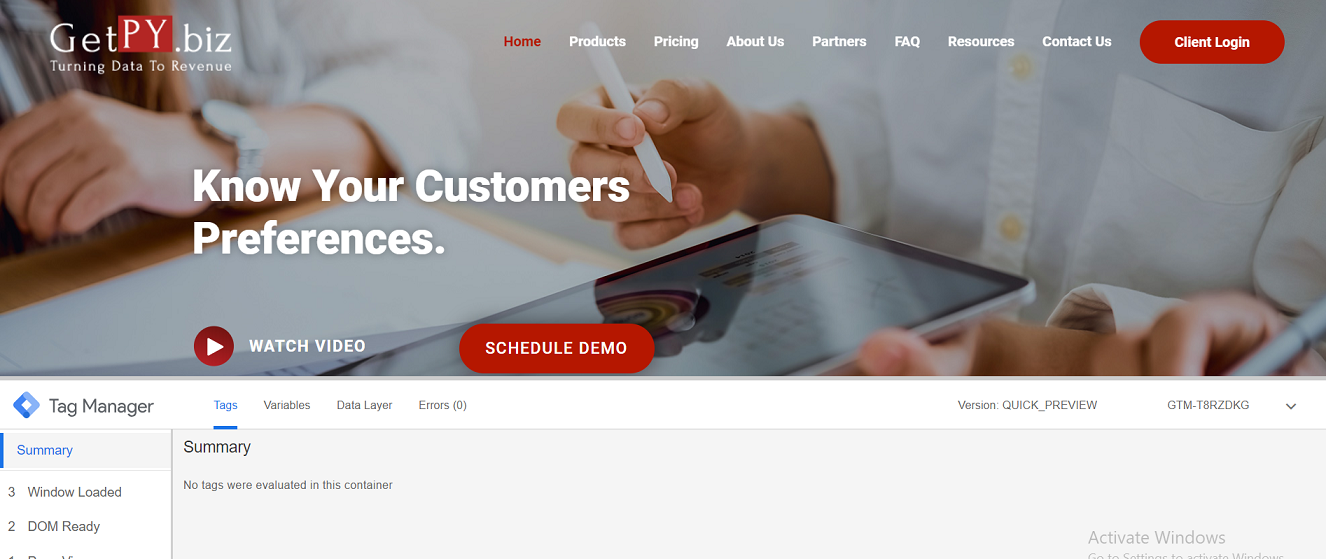
****



You can also check using the preview option in the GTM and reloading the home page to see GTM appearing on your site



**Preview option in GTM Dashboard**

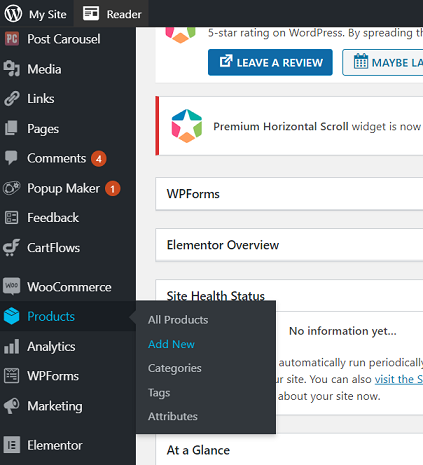


**Fig: Tag Manager appearing in the website**

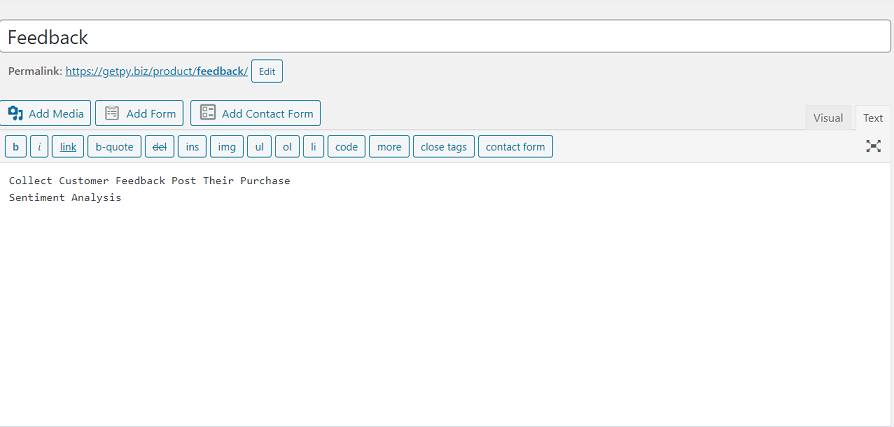
**6.3 Implementing WooCommerce Store**

For creating a new product follow the following steps:

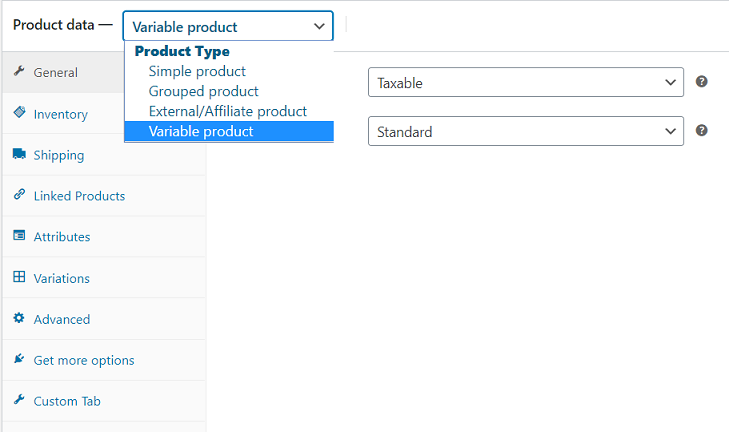
**Goto site/wp-admin page🡪Select Products🡪Add New**

****

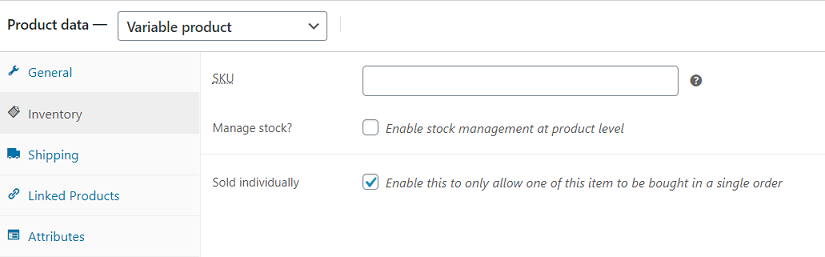
**Enter Product Name and Product Description**

****

For creating a variable product choose the **Product type** as **Variable product** in the Product data

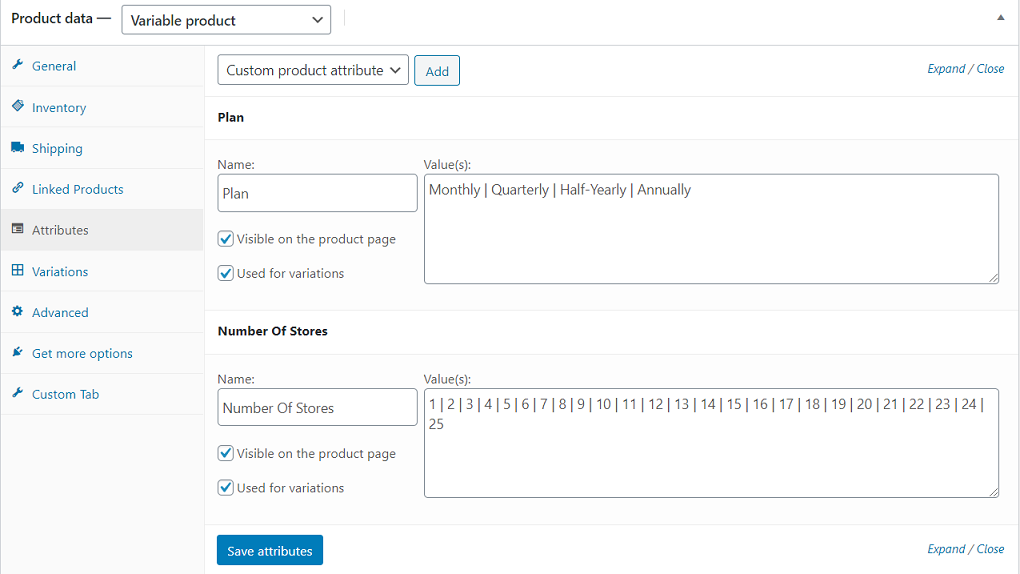


In the **Inventory** section, select the option **Sold Individually** to allow only one product to be sold individually.

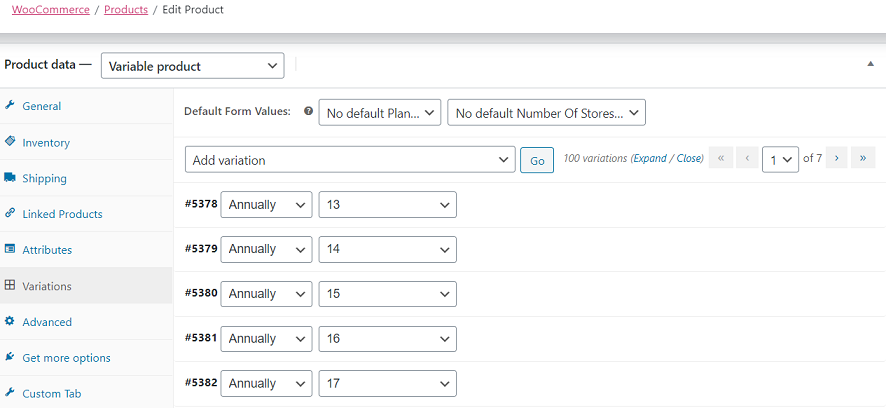


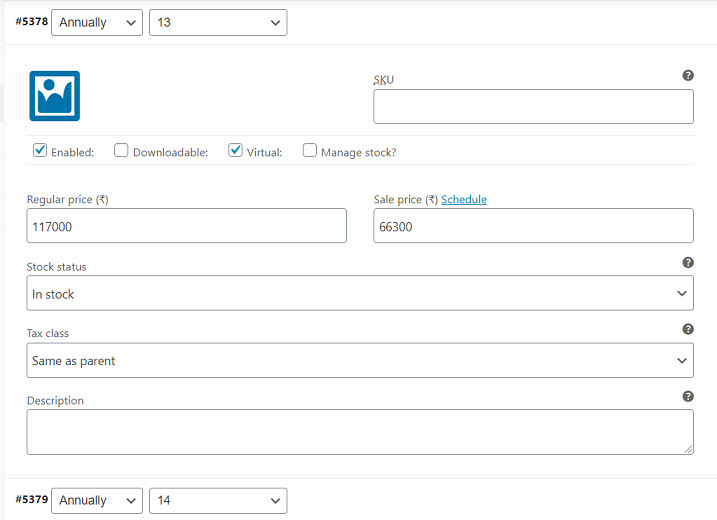
For creating custom attributes for the product go to **Attribute** section🡪**Add**. For examples I have created two custom attributes:

* **Plan**
* **Number Of Stores**

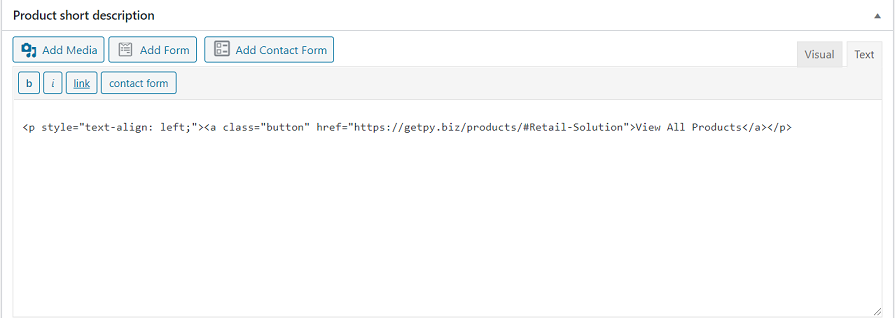
****

In the **Variations** tab add new variation and enter its pricedetails

****

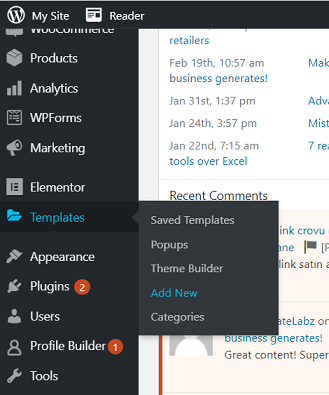
****

In the **Product short description** section we can give a briefdescripton of the Product**.** I have added the **View All Products** button in this section.

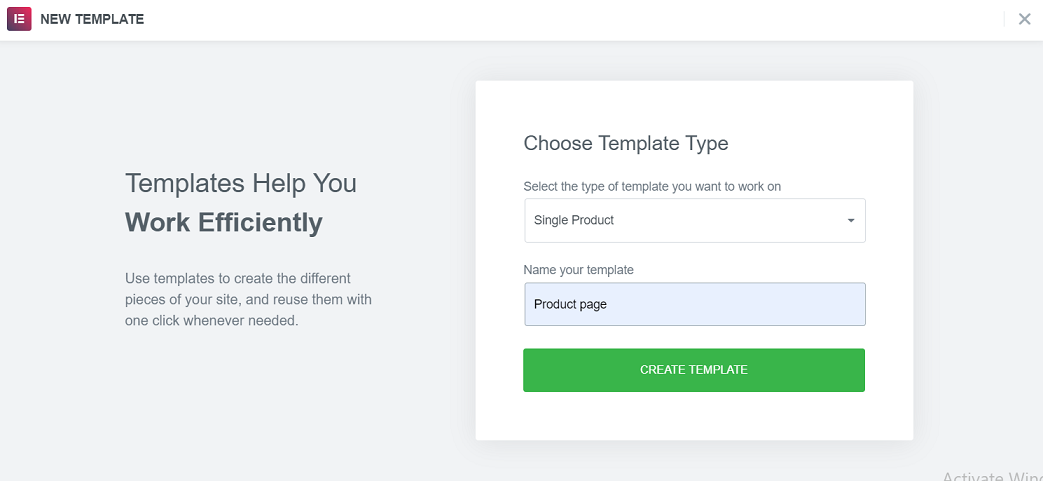
****

**For designing the Single Product Page**

Go to **Templates tab🡪Add New**

****

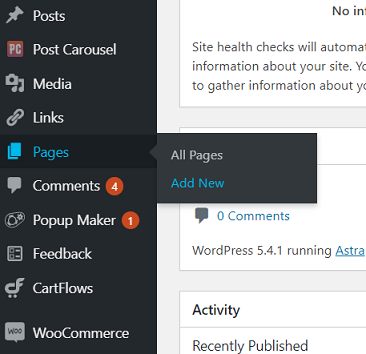
Select the **Template** type as **Single Product** and make the design in elementor.



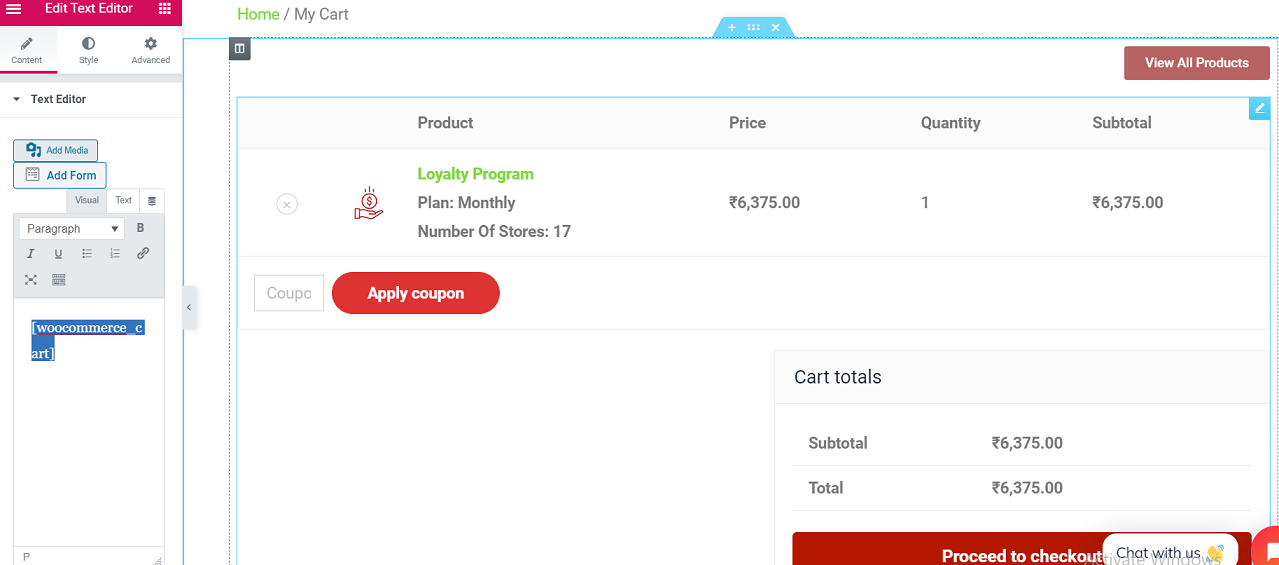
**Woocommerce Cart Page**

For creating woocommerce cart page

**Go to Pages tab🡪Add New**

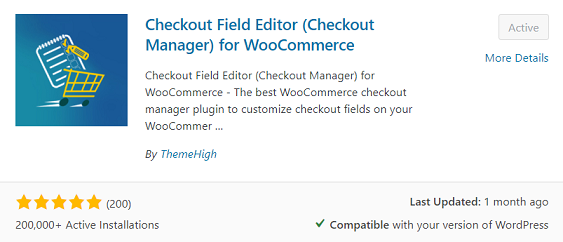
****

Add the shortcode **[woocommerce\_cart**] to automatically create the default cart page.

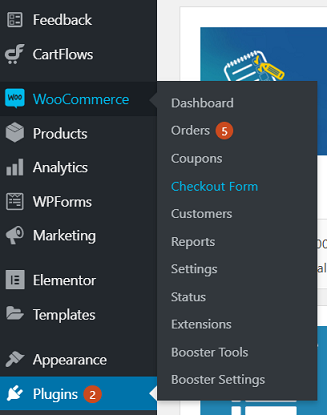


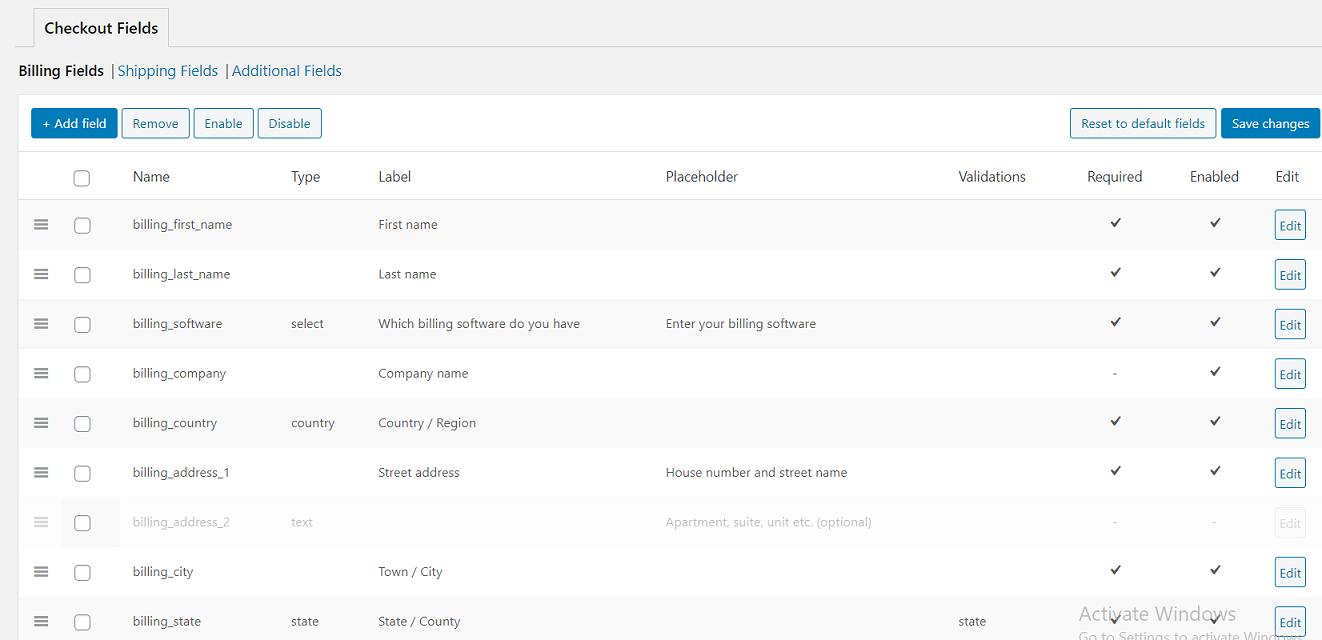
**Woocommerce Checkout Page**

Similarly, like the cart page, add the shortcode **[woocommerce\_checkout]** in a new page to create Checkout Page. For adding new fields or modifying existing fields in the checkout form, I have installed the plugin **Checkout Field Editor for WooCommerce**.

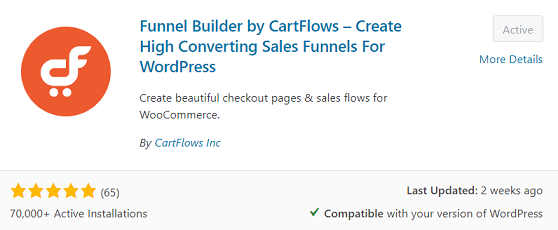


After activating the plugin, go to the **Woocommerce** tab 🡪 **Checkout Form** to modify theexisting fields in the checkout form.



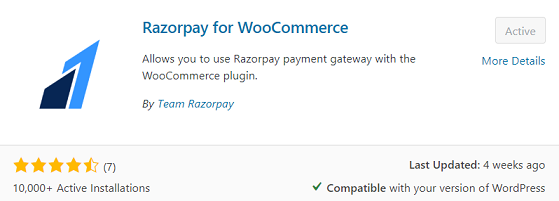


For designing the Checkout page and Thank you page I have installed the **CartFlows** plugin.

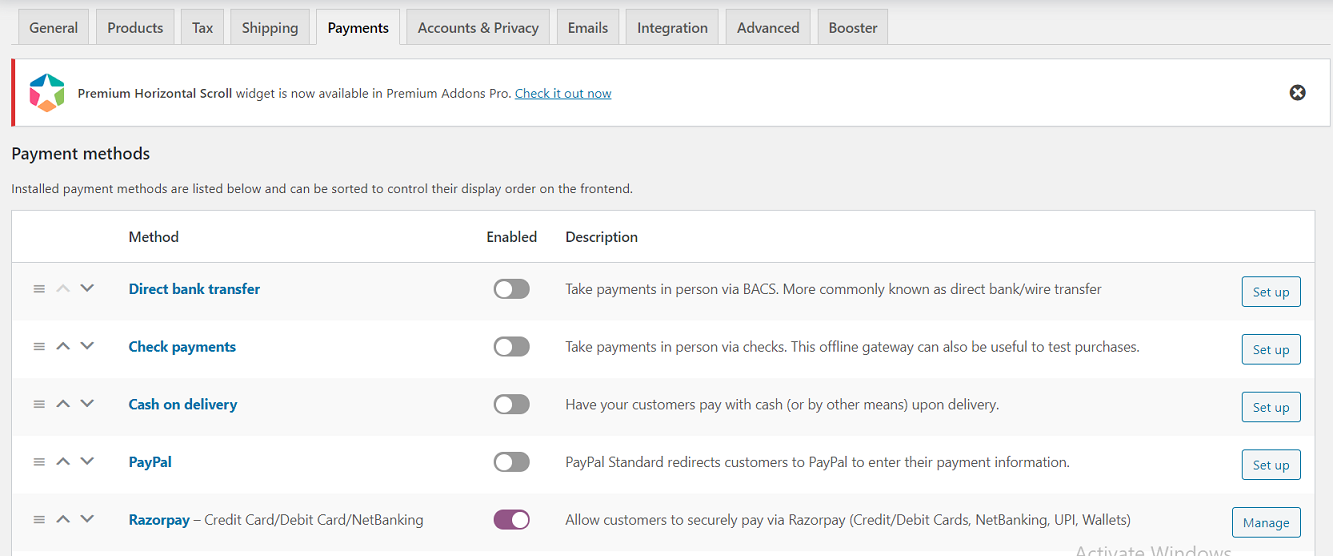


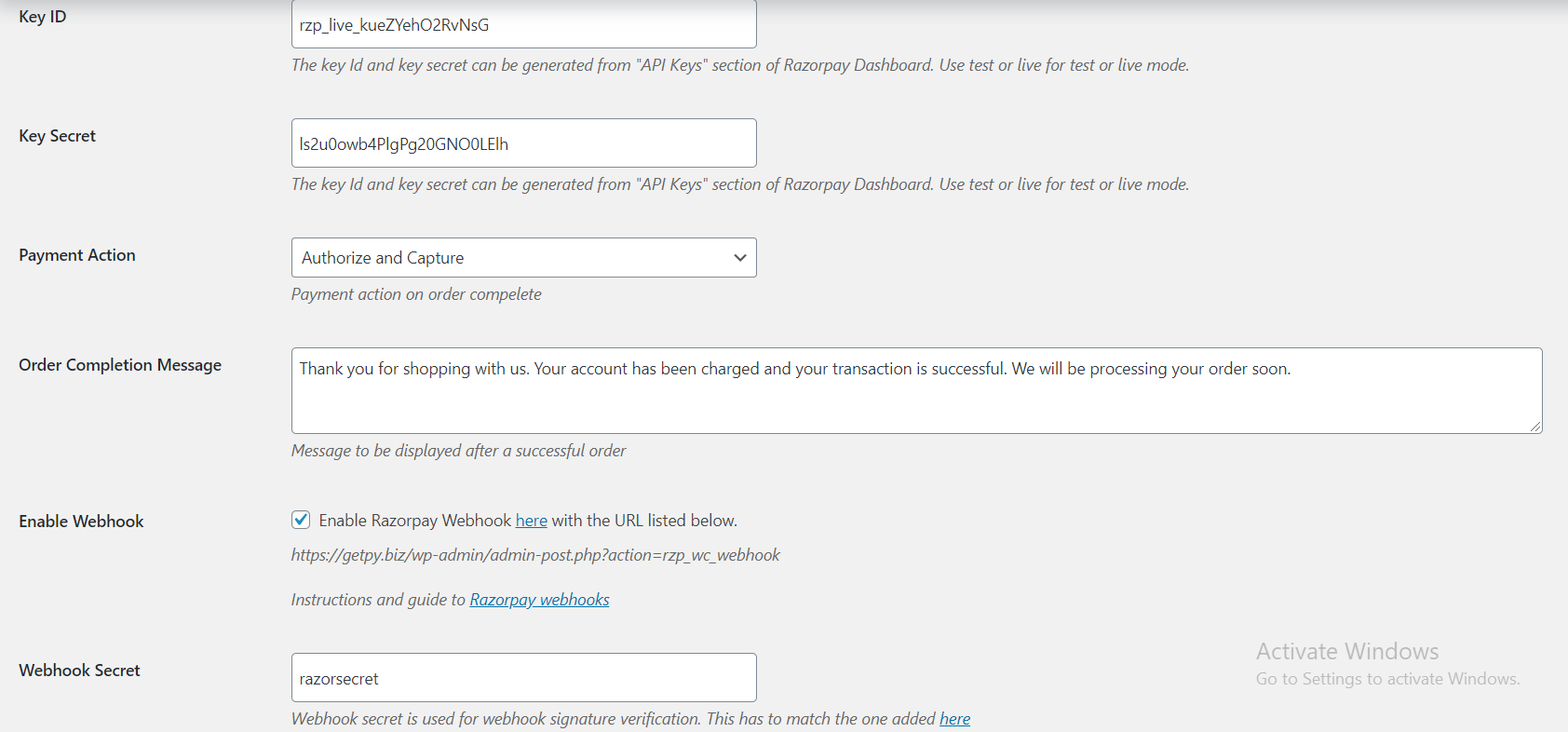
Go to **CartFlows** tab🡪 **Flows** to view and edit the Checkout and Thank You page with elementor.

**Razorpay Integration:** For integrating Razorpay in Woocommerce Checkout Page I have installed the **Razorpay for WooCommerce** plugin.

****

Go to **Woocommerce** Tab🡪**Settings** and Select the **Payments** tab. Here set the **Razorpay** Method as enabled and click the **Manage** button to set the Key ID, Key Secret, Enable Webhook and Webhook Secret option.





In the Razorpay website go to Settings tab and configure Live/Test Webhooks and API keys(Key Id and Key Secret).

**Adding Additional Codes for Customizing Page Design**

Go to **Appearance** tab **🡪 Theme Editor** and Select the **functions.php** file.

****

**Code to adjust the Related Products size in Product Page**

add\_filter( 'wp\_head' , 'related\_products\_style' );

function related\_products\_style() {

if( is\_product() ) :

?>

<style>

.woocommerce .related ul.products li.product,.woocommerce .related ul li.product, .woocommerce .upsells.products ul.products li.product, .woocommerce .upsells.products ul li.product, .woocommerce-page .related ul.products li.product, .woocommerce-page .related ul li.product, .woocommerce-page .upsells.products ul.products li.product, .woocommerce-page .upsells.products ul li.product {

width: 40% !important;

}

**Code to make the Related Products product title appear in a**

**Single line**

.woocommerce ul.products li.product .woocommerce-loop-product\_\_title, .woocommerce ul.products li.product h3 {

overflow: hidden;

text-overflow: ellipsis;

white-space: nowrap;

}

</style>

<?php

endif;

}

**Code to change the display of variable products price format**

add\_filter( 'woocommerce\_variable\_price\_html', 'bbloomer\_variation\_price\_format\_min', 9999, 2 );

function bbloomer\_variation\_price\_format\_min( $price, $product ) {

$prices = $product->get\_variation\_prices( true );

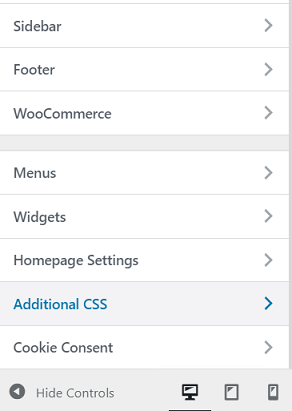
$min\_price = current( $prices['price'] );

$price = sprintf( \_\_( 'From: %1$s', 'woocommerce' ), wc\_price( $min\_price ) );

return $price;

}

Go to **Apperance** tab🡪 **Customise** andSelect the **Additional CSS** option. Here also we can add additional css code to modify webpages

****

**Code to hide the update cart button in Cart Page**

button[name="update\_cart"] {

display: none !important;

}

**Code to change the design of Place Order button in Checkout Page**

button#place\_order {

color: #fff !important;

background-color: #000000;

font-size: 20px;

padding: 15px 32px;

border-radius:5px;

text-align:center

}

**Code to change the design of Add to Cart Button in Single Product Page and Product Archive Page**

.single-product .product .single\_add\_to\_cart\_button.button{

border-radius: 5px;

}

.woocommerce .product .add\_to\_cart\_button.button{

border-radius: 5px;

}

**7. CONCLUSION**

Ecommerce covers a vast area of topics such as [internet marketing](https://en.wikipedia.org/wiki/Online_advertising), [online transaction processing](https://en.wikipedia.org/wiki/Online_transaction_processing), [inventory management systems](https://en.wikipedia.org/wiki/Inventory_management_software), automated [data collection](https://en.wikipedia.org/wiki/Data_collection) systems etc. Covering all the areas of ecommerce and implementing them would take a huge amount of time. In the four months of my internship period, I have explored only a small portion of it. I have tried to meet the requirements of the project that I was assigned to within the limited period of time and with little experience. Working in this project has been an unique experience.

**8. BIBLIOGRAPHY**

**1.** <https://www.singer.io/>

**2.** <https://github.com/singer-io/getting-started>

**3.** <https://docs.microsoft.com/en-us/azure/automation/automation-webhooks>

**4.** <https://blog.cloud-elements.com/webhooks-vs-polling-youre-better-than-this>

**5.** <https://www.youtube.com/watch?v=umr-ID4QzCc>

**6.** <https://rudrastyh.com/woocommerce/custom-checkout-validation.html>