**Magento Open Known issue**

**Admin Filter Issue:**

* **Issue:**  When frequently switching between bookmark views, the saved filters in admin grids do not function as intended. The filters get mixed up, and the correct view is not saved.

UI Bookmark Component in Magento 2 is responsible for storing the state of data grids, including filters, column positions, sorting, and pagination. This component is used to manage the state of data grids and provides a way to save and restore the state of these grids

* **Steps to reproduce:** 
  + Access the Sales Order grid in the Admin.
  + Create two to three filters.
  + Verify the filter settings by switching views to ensure they are saved accurately.
  + Go to Filter1 or Filter2.
  + Refresh the page to update the displayed data.
  + Switch to a different view and notice that the filters remain unchanged.
  + Notice that the default view is now displaying filtered results, even though no specific filter was set for it.
* **Expected results:** The filters do not get interchanged and retain their original state.
* **Actual results:** When modifying a view, the filters get mixed up, and the correct view is not saved.
* **Solution:**

\* To apply the patch, use the Quality Patches Tool (QPT) 1.1.39. The patch ID is ACSD-52219.

* + You can check if the patch is compatible with your Adobe Commerce version by updating the **magento/quality-patches** package to the latest version and checking the compatibility on the Quality Patches Tool: Search for patches page.
  + In your case, since you are using Magento 2.2.8, you may need to upgrade to a compatible version or apply a custom fix.
  + Regarding your specific issue with the order grid not giving correct results on filtering by purchase date in the locale, it is related to the date format issue in the filter.
  + You can try to fix this by overriding the date format in the filter to match the Vietnamese locale.

public function prepareDataSource(array $dataSource)

{

if (isset($dataSource['data']['items'])) {

foreach ($dataSource['data']['items'] as &$item) {

$orderId = $item['entity\_id'];

$order = $this->orderRepository->get($orderId);

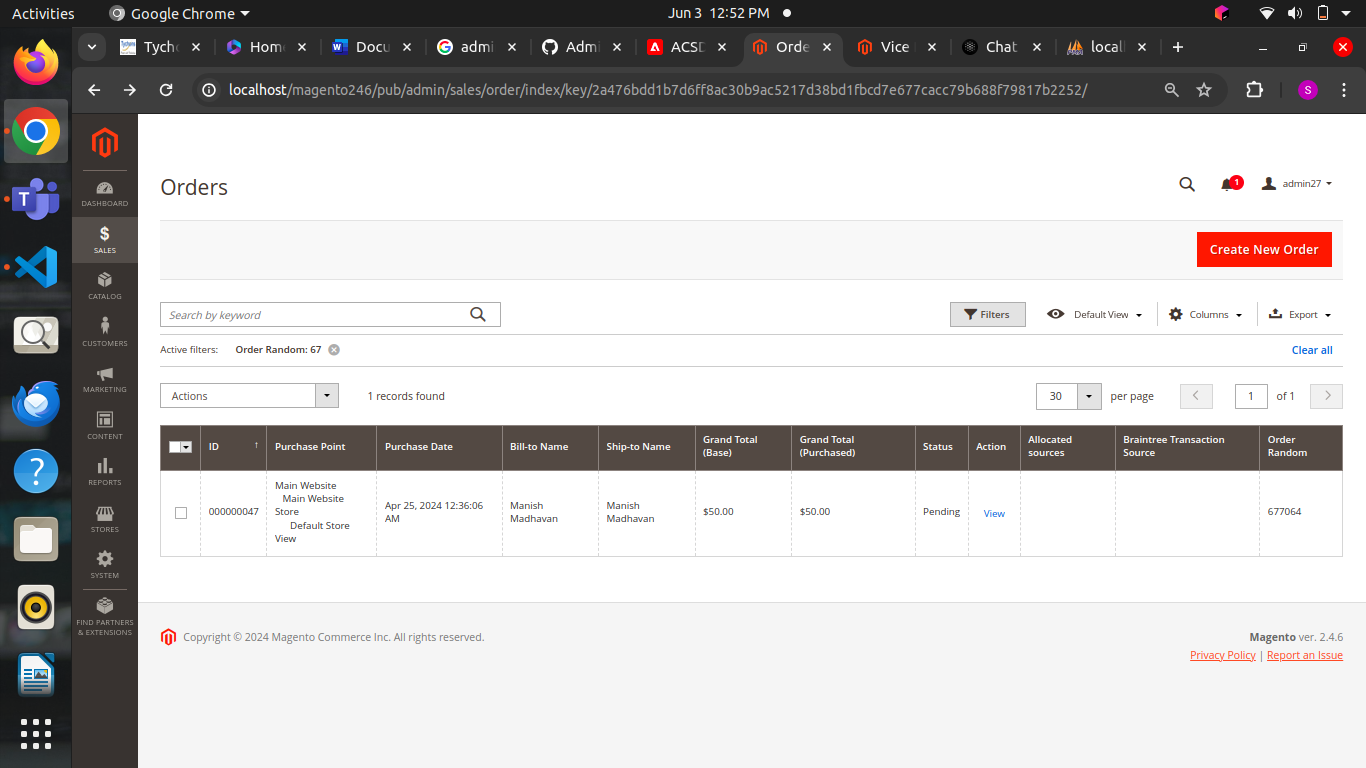
$item[$this->getData('name')] = $order->getData('order\_random');

}

}

return $dataSource;

}



* In php my admin localhost clicks ui\_bookmark it displays all the filters which are active you can update the filter or reset the filter to clear the issue

public function prepareDataSource(array $dataSource) {

if (isset($dataSource['data']['items'])) {

foreach ($dataSource['data']['items'] as & $item) {

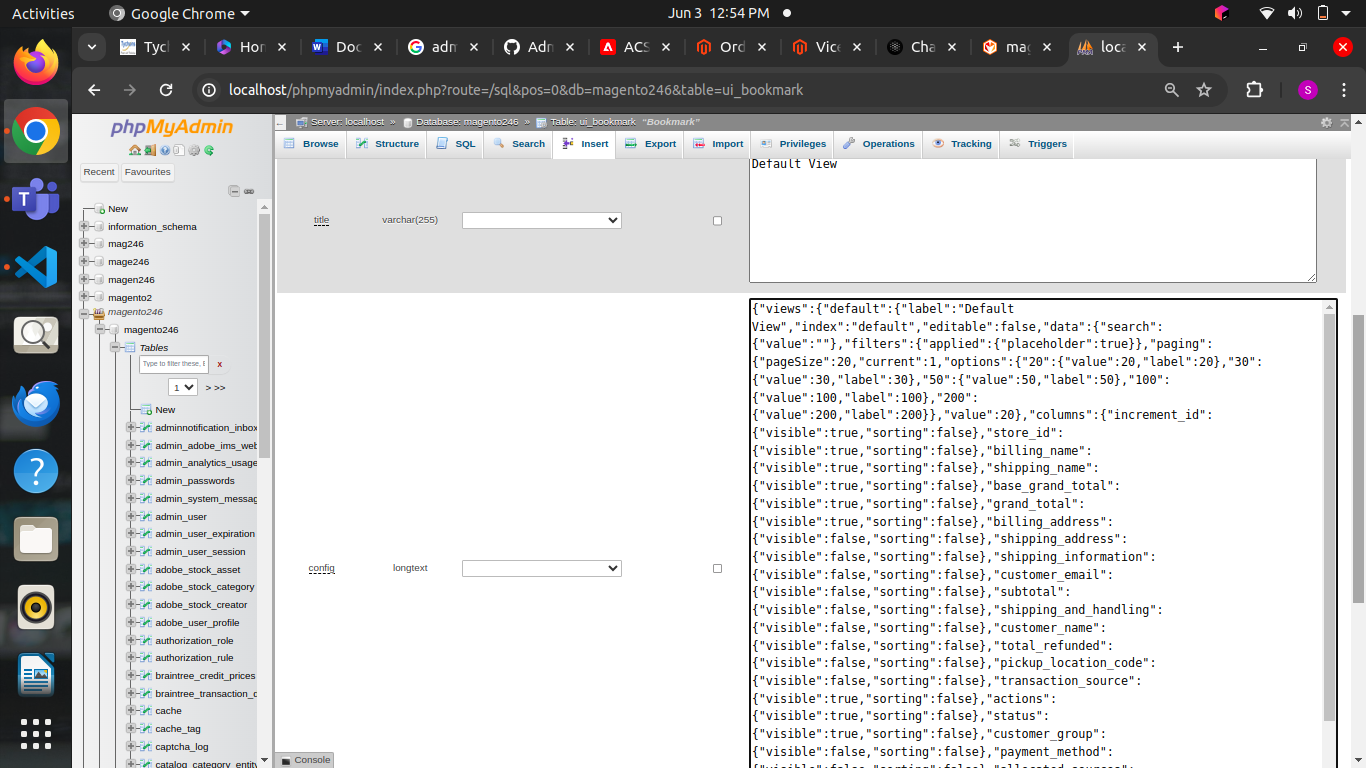
$item[**$this**->getData('name')] = **$this**->escaper->escapeHtml($item[**$this**->getData('name')]);

$item[**$this**->getData('name')] = date ('d/m/Y', strtotime($item[**$this**->getData('name')]));

}

} return $dataSource;

}



You can also update the filter   
SELECT \* FROM ui\_bookmark WHERE user\_id = 1 AND namespace LIKE '%sales\_order\_grid%';  
by providing the id details

**Apply patch using quality patches tool:**

* The ACSD-52219 patch resolves the issue of saved filters not working as expected in Magento 2 by fixing the problem where the admin grids' saved filters do not work as intended when frequently switching between bookmark views.
* The patch is available when the Quality Patches Tool (QPT) 1.1.39 is installed, and it is compatible with Adobe Commerce versions 2.4.5 - 2.4.6-p3.
* The issue occurs when frequently switching between bookmark views, the saved filters in admin grids do not function as intended. The filters get mixed up, and the correct view is not saved.
* To apply the patch, you can use the Quality Patches Tool, which allows you to apply, revert, and view general information about all individual patches that are available for the installed version of Adobe Commerce.
* composer require magento/quality-patches:1.1.39 2bin/magento quality:patch:apply ACSD-52219
* You can also apply patches like this also.

**2)Collection override issue**

* Difficulty in customizing or extending the behavior of the Fulltext\Collection class, which is used for searching products in Magento 2.
* This issue is particularly challenging because the Fulltext\Collection class is designed to be a virtual type, making it difficult to override using the standard Magento 2 dependency injection (DI) mechanism.

**Causes of collection override issue**

* The primary cause of this issue is the design of the Fulltext\Collection class as a virtual type.
* Virtual types are designed to be used as a base class for other classes, but they cannot be directly extended or overridden using the standard Dependency injection mechanism.
* This makes it difficult to customize the behavior of the Fulltext\Collection class without modifying the core code.

**Solution to clear this issue**

* **Use Plugins:** Instead of directly overriding the Fulltext\Collection class, you can use plugins to customize its behavior.

Plugins are a powerful tool in Magento 2 that allow you to extend the functionality of a class without modifying its core code.

* **Customize the Search Request:** Another way to customize the search behavior is to modify the search request.
* You can create a custom module and add a search\_request.xml file that defines the search request.
* Use the prepareRequest Method: The prepareRequest method is a key part of the search process in Magento 2. You can use this method to customize the search request and add custom filters.

**Prevention methods**

* Use plugins
* Customize the search request
* Use Preparerequest method

**Collection override with grid filter issue in magento 2**

The collection override in grid filter issue in Magento 2 occurs when trying to customize the behavior of the Fulltext\Collection class, which is used for searching products.  
  
you can follow the same procedures to clear the collection override issue create plugin and use preparerequest method to customize your search request.

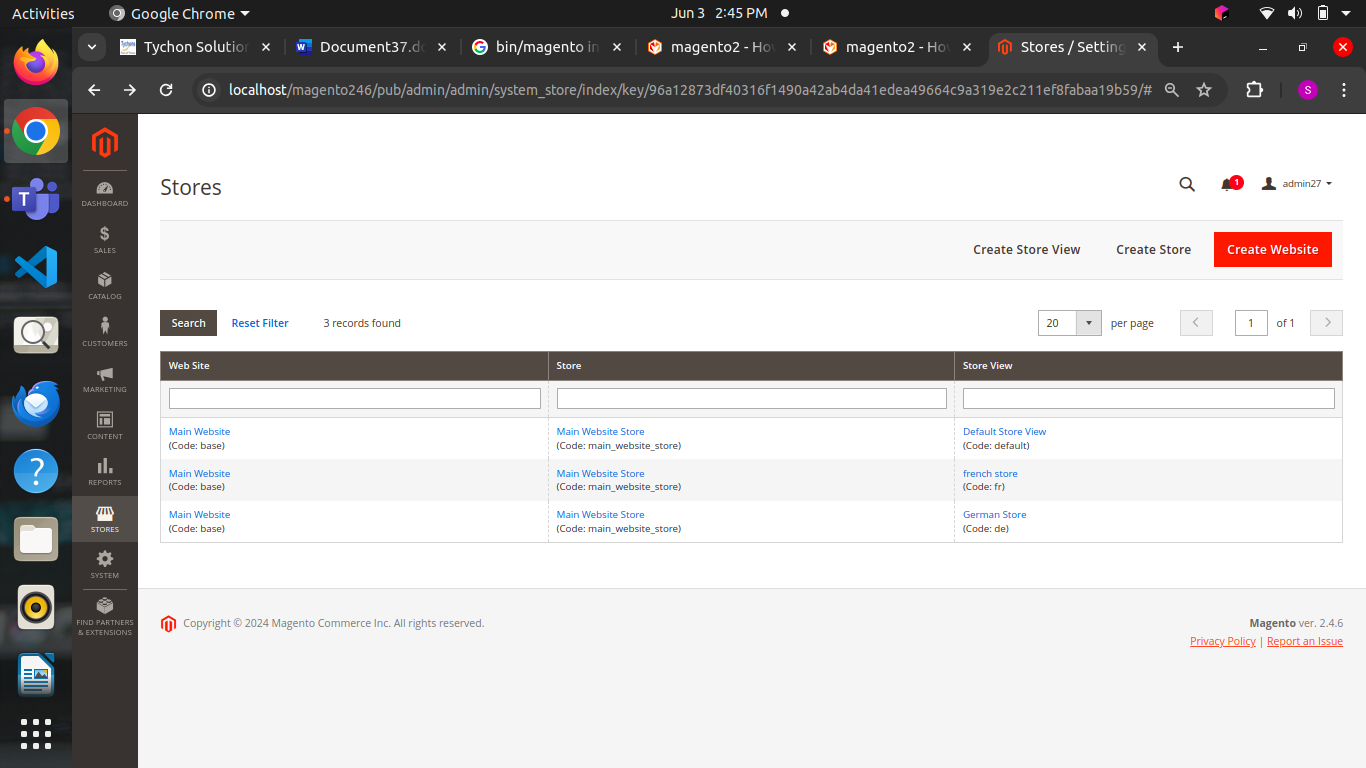
**3) Issues with multi-store configurations:**

Problem: Multi-store configurations can lead to issues such as incorrect store views, URL rewrites not working, or configuration settings not being applied correctly.

**Solution:**

* **Verify store configuration:** Navigate to Stores > All Stores.

Ensure each store view is correctly configured with the appropriate base URLs.



* **Check Url rewrites:**

Navigate to Marketing > URL Rewrites.

Ensure URL rewrites are correctly set up for each store view.

* **Set Store Specific Configurations:**

Navigate to Stores > Configuration.

Ensure store-specific settings are correctly applied under the relevant store view scope.

* **Reindex URL Rewrites:**

Open your server via SSH.

Run the reindex command for URL rewrites:

bin/magento indexer:reindex url\_rewrite

* **How to delete url rewrite**

delete from URLrewrite;  
 delete from catalog\_url\_rewrite\_product\_category;

* **Regenerate url rewrite**

For product:

php bin/magento ok:urlrewrites:regenerate

For Category:

php bin/magento ok:urlrewrites:regenerate --entity-type=category

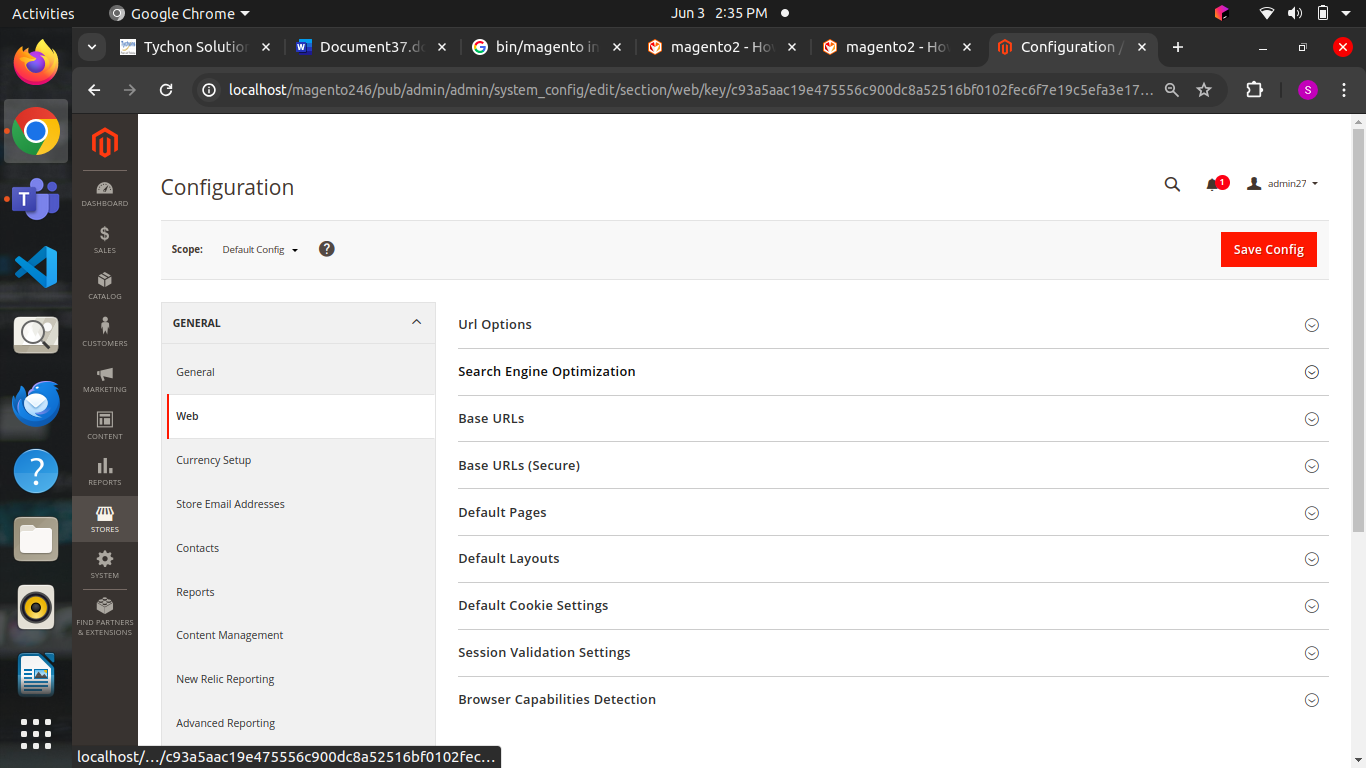
* **By admin panel we can regenerate url**

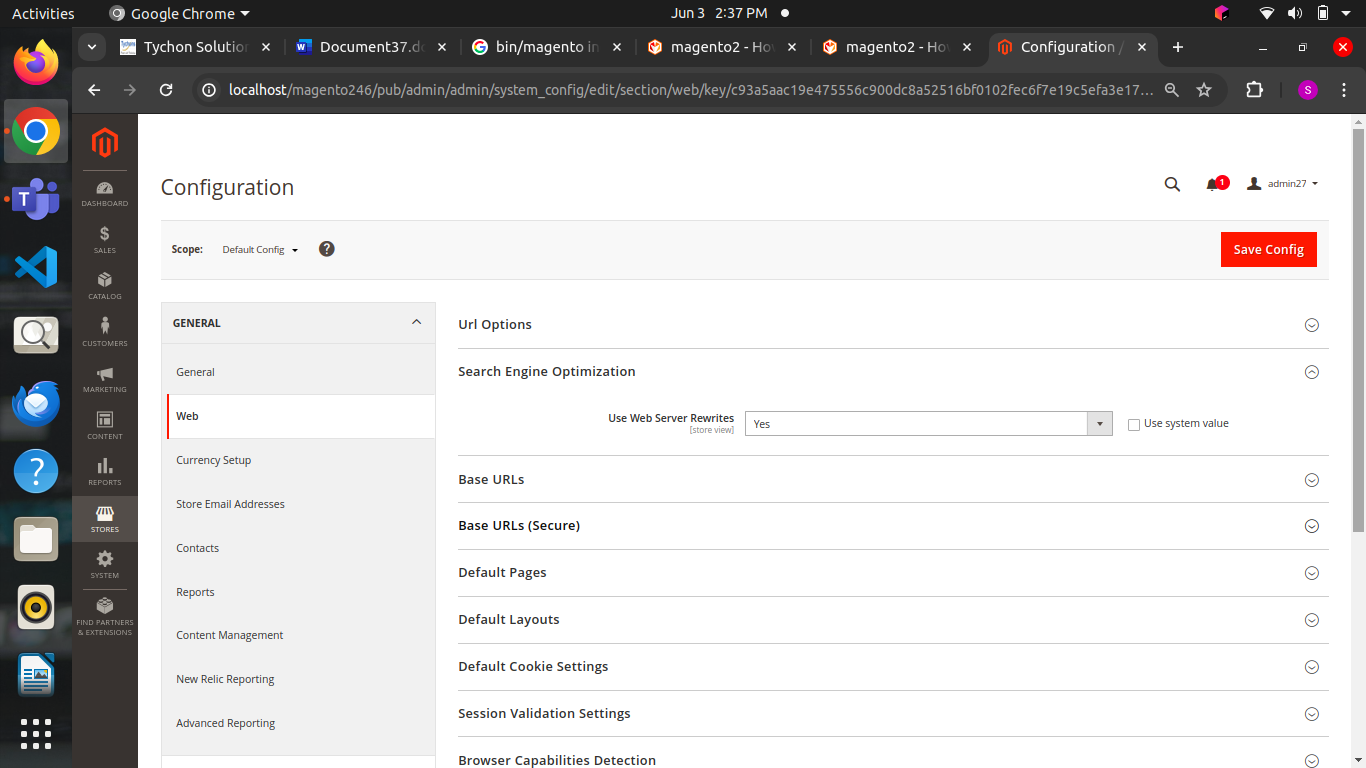
Navigate to Stores -> Configuration -> General -> Web

Expand the Search Engine Optimization section

Set the Use Web Server Rewrites to ‘Yes’.

Click the Save Config button to save your settings.





**Multi-Store Configuration Drop Down Problem:**

* Problem: When trying to edit a secondary store, the configuration dropdown reverts to the main store.
* Solution: Try deleting the secondary store and then re-adding it. If the issue persists, it may be related to the database configuration. Ensure that the database is properly configured for multi-store setup

4) **Issues with Data Migration Error**

**Problem:** Data migration errors can occur during Magento 1 to Magento 2 migrations, often due to data inconsistencies, mapping issues, or corrupted data.

* **Verify Data Consistency:**

Ensure data integrity in the Magento 1 database. Check for and resolve any inconsistencies.

* **Review Migration Tool Logs:**

Check logs generated by the Data Migration Tool in var/log/.

Look for errors or warnings that indicate what went wrong.

* **Adjust Mapping Files:**

Modify the map.xml and map-tier-price.xml files to correctly map data from Magento 1 to Magento 2.

* bin/magento migrate:data –reset
* We can reset the migration data by using this command
* Example snippet

<field\_rules>

<field\_rule>

<source>old\_field\_name</source><destination>new\_field\_name</destination>

</field\_rule>

</field\_rules>

* Run migration commands:
* php bin/magento migrate:settings path/to/config.xml
* php bin/magento migrate:data path/to/config.xml
* **Fix Data Corruptions:**
* Manually fix any data corruptions identified during the migration process.
* Use SQL queries or scripts as needed.

**Issues we faced in migration 8.1 to 8.2 php versions**

* Some Magento extensions or custom modules may not be compatible with PHP 8.2, leading to errors or unexpected behavior in your store
* PHP 8.2 may deprecate certain functions or features that were supported in PHP 8.1. If your Magento codebase relies on these, you may need to update your code to stick to the latest PHP standards
* PHP 8.2 may introduce new error messages or warnings for code that does not stick to the latest PHP standards, which could impact your store's functionality
* Your web server and other server software must be compatible with PHP 8.2. If they are not, you will need to update or replace them to support the new PHP version
* Any customizations or theme modifications made to your Magento store may need to be reviewed and updated for compatibility with PHP 8.2
* If your Magento store integrates with third-party services or APIs, you will need to verify that they are compatible with PHP 8.2. Some integrations may require updates or adjustments to work correctly with the new PHP version
* To ensure a smooth migration, it is recommended to:
* Test the new PHP version in a development environment to identify potential issues.
* Back up your site before making any changes.
* Work with an experienced developer or Magento expert.
* Verify compatibility of all extensions and custom modules.
* Review and update customizations and theme modifications.
* Ensure server software compatibility.
* Verify compatibility of third-party integrations

Migrating from 8.1 to 8.2 the issues we will face in methods

DateTime::createFromImmutable() and DateTimeImmutable::createFromMutable():

* These methods have tentative return types, which may cause compatibility issues

SplFileObject::hasChildren() and SplFileObject::getChildren():

* These methods now enforce their signatures, which may require changes in your code

ODBC and PDO\_ODB Escaping:

* When passing a connection string, username, and password, PHP 8.2 may escape these values differently, leading to compatibility issues

FilesystemIterator's Constructor:

* In PHP 8.2, you must explicitly set the FilesystemIterator::SKIP\_DOTS flag if you want to exclude dot files (. and ..)

String Functions:

* Functions like strtolower(), strtoupper(), stristr(), stripos(), strripos(), lcfirst(), ucfirst(), ucwords(), and str\_ireplace() are no longer local-sensitive, which may affect your code

Dynamic Class Properties and String Interpolation:

* PHP 8.2 introduces dynamic class properties and ${var} and ${expr} style string interpolation, which may require adjustments in your code

Dynamic Class Properties and String Interpolation:

* PHP 8.2 introduces dynamic class properties and ${var} and ${expr} style string interpolation, which may require adjustments in your code

Readonly Properties and Classes:

* PHP 8.2 introduces readonly properties and classes, which can prevent dynamic property changes and ensure typed properties. This may impact your code if you rely on dynamic property changes

These changes can lead to compatibility issues if not properly addressed during the migration process.

Steps to clear the issue   
 \* Identify Deprecated Functions and Features

\* Update Code to stick to PHP 8.2 Standards

\* Verify Compatibility of Extensions and Modules

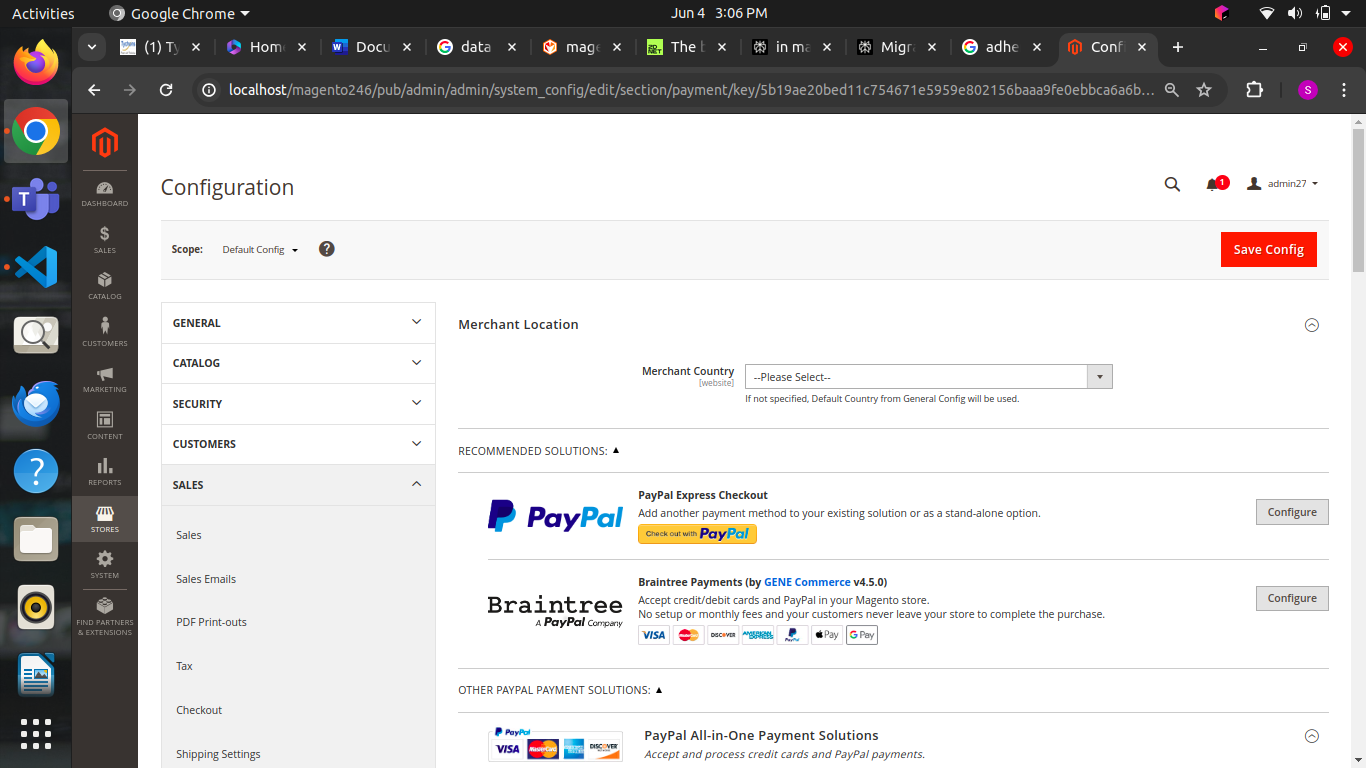
\* Perform Thorough Testing

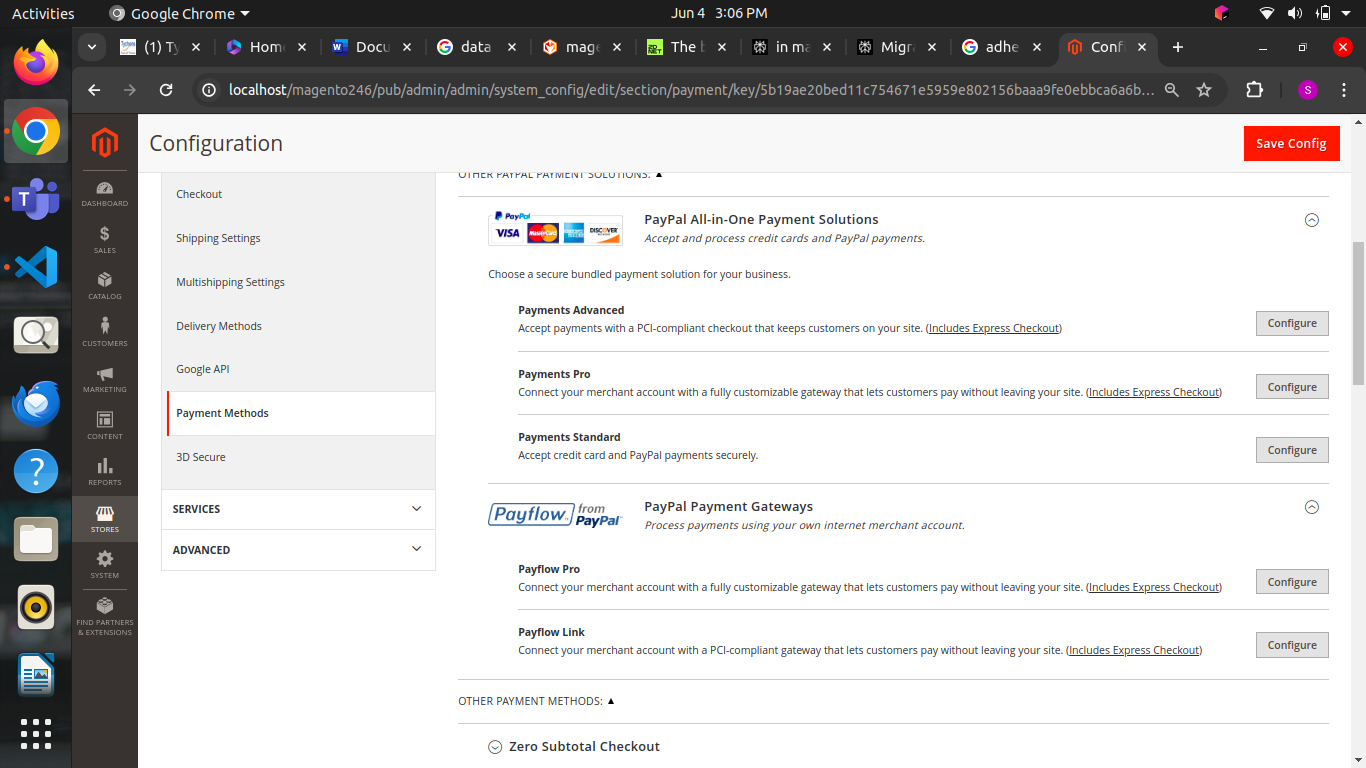
**5) Payment Method Not Displaying in Order Details**

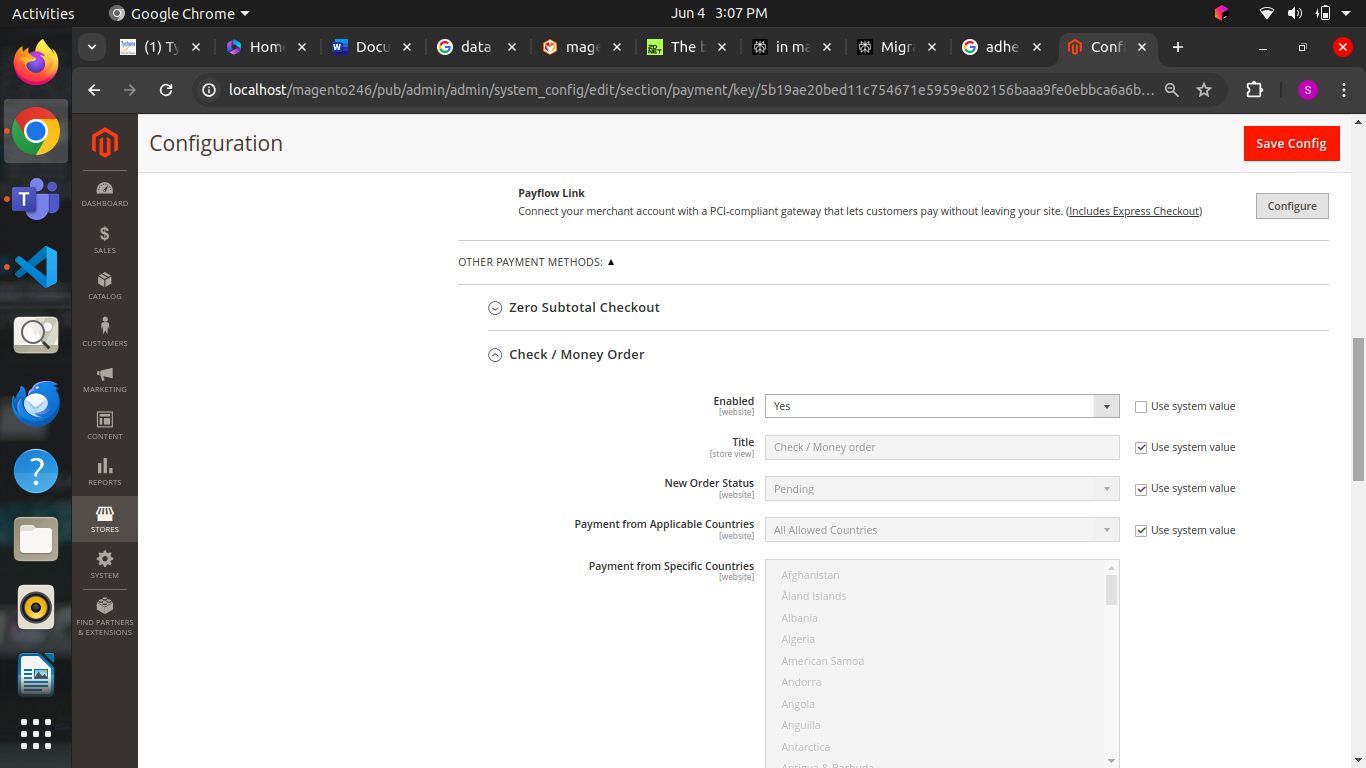
Problem: The payment method is not displayed in the order details, causing issues with order tracking.

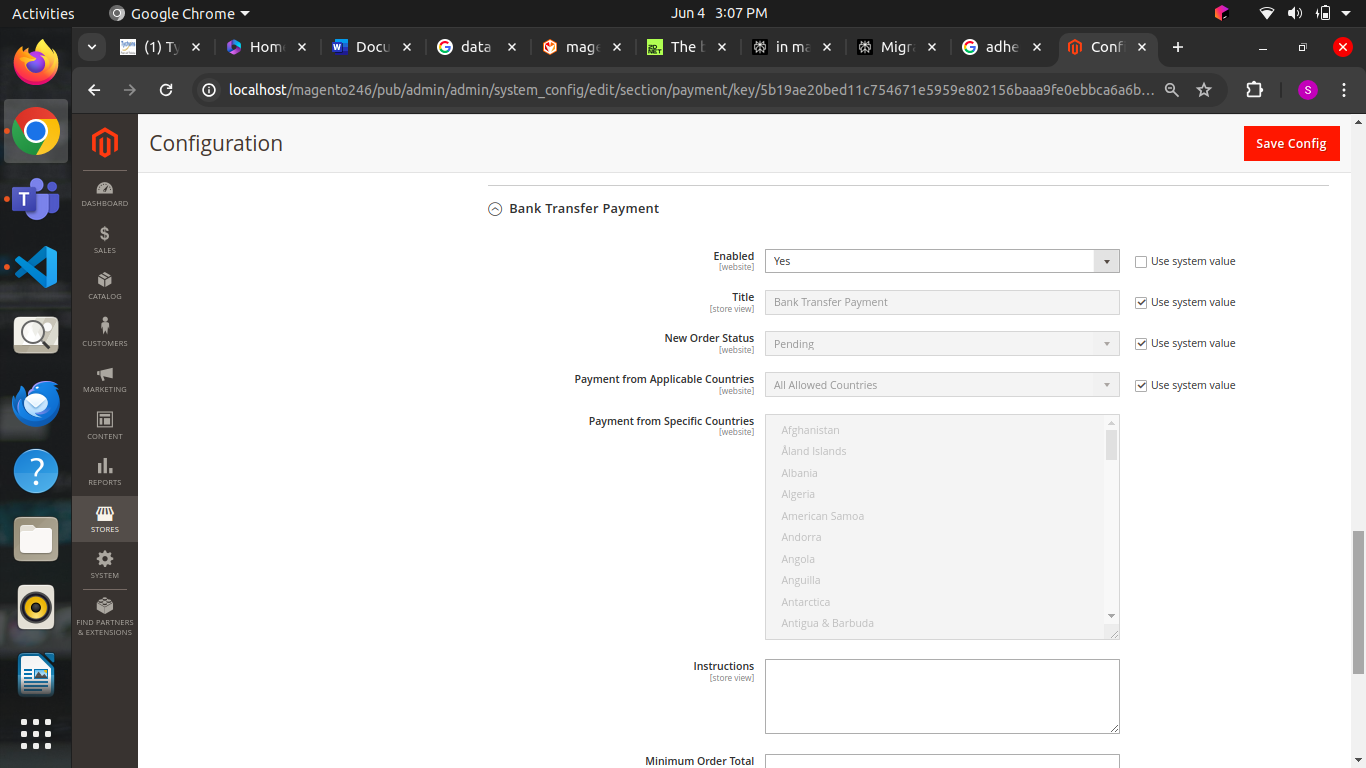
Solution:

1. Log in to the Magento 2 admin panel.
2. Go to Stores > Configuration > Sales > Payment Methods.
3. Ensure that the payment method is enabled and configured correctly.
4. Go to Stores > Configuration > Sales > Checkout > Payment Methods.
5. Verify that the payment method is listed under Available Payment Methods.









**Steps to Clear the Issue Programmatically:**

**Get the Payment Method Configuration:**

$paymentMethodConfig = \Magento\Payment\Model\Config:getInstance()->getMethodInstance('your\_payment\_method\_code');

**Check if the Payment Method is Enabled:**

if (!$paymentMethodConfig->isActive()) {

// Enable the payment method

$paymentMethodConfig->setEnabled(true); }

Clear cache

**Check Payment Method Configuration:**

if (!$paymentMethodConfig->isAvailable()) {

// Set the payment method as available

$paymentMethodConfig->setAvailable(true); }

**Check Payment Method Order Status:**

$order = \Magento\Sales\Model\Order::load($orderId); if ($order->getState() !== \Magento\Sales\Model\Order::STATE\_PROCESSING) { // Set the order state to processing $order->setState(\Magento\Sales\Model\Order::STATE\_PROCESSING);

}

**specific method names that commonly cause errors in Magento 2 payment modules**

* **authorizeAndCapture():**

This method is responsible for authorizing and capturing payments. It is a critical method in the payment workflow and can cause errors if not implemented correctly.

* **getInfo():**

This method is used to retrieve payment information from the payment gateway. It can cause errors if the payment gateway is not configured correctly or if there are issues with the payment gateway API.

* **getForm():**

This method is used to generate the payment form. It can cause errors if the form is not generated correctly or if there are issues with the form submission.

* **getPaymentInfo():**

This method is used to retrieve payment information from the payment gateway. It can cause errors if the payment gateway is not configured correctly or if there are issues with the payment gateway API.

* **placeOrder():**

This method is used to place an order. It can cause errors if there are issues with the order creation or if the payment method is not configured correctly.

* **capturePayment():**

This method is used to capture the payment. It can cause errors if there are issues with the payment capture or if the payment method is not configured correctly.

* **refundPayment():**

This method is used to refund a payment. It can cause errors if there are issues with the refund or if the payment method is not configured correctly.

* **CancelPayment():**

This method is used to cancel a payment. It can cause errors if there are issues with the cancellation or if the payment method is not configured correctly.

* **getPaymentStatus():**

This method is used to retrieve the payment status. It can cause errors if there are issues with the payment status or if the payment method is not configured correctly.

* **getPaymentMethod():**

This method is used to retrieve the payment method. It can cause errors if there are issues with the payment method or if the payment method is not configured correctly

* + **Prevent conflicts between payment related modules in magento 2**
  1. Disable any third-party modules
  2. Check for conflicts
  3. Inspect Database
  4. Check payment method configuration and gateway integration
  5. Check modules dependencies to avoid conflicts.