**Permission Implementation Flow for New Forms/Tabs**

**Step 1: Identify Required Permissions**

Look at your form/tab and identify what actions users can perform:

# Example: For a "Fees Management" form, you might need:

# - create\_fee

# - edit\_fee

# - delete\_fee

# - view\_fee\_reports

# - process\_payments

# - export\_fee\_data

**Step 2: Add Permissions to Database**

Insert the permissions for each role in your role\_permissions table:

-- Admin gets all permissions

INSERT INTO role\_permissions (role\_id, permission, created\_at) VALUES

('1', 'create\_fee', NOW()),

('1', 'edit\_fee', NOW()),

('1', 'delete\_fee', NOW()),

('1', 'view\_fee\_reports', NOW()),

('1', 'process\_payments', NOW()),

('1', 'export\_fee\_data', NOW());

-- Headteacher gets most permissions (maybe no delete)

INSERT INTO role\_permissions (role\_id, permission, created\_at) VALUES

('2', 'create\_fee', NOW()),

('2', 'edit\_fee', NOW()),

('2', 'view\_fee\_reports', NOW()),

('2', 'process\_payments', NOW()),

('2', 'export\_fee\_data', NOW());

-- Finance role gets fee-specific permissions

INSERT INTO role\_permissions (role\_id, permission, created\_at) VALUES

('5', 'create\_fee', NOW()),

('5', 'edit\_fee', NOW()),

('5', 'view\_fee\_reports', NOW()),

('5', 'process\_payments', NOW()),

('5', 'export\_fee\_data', NOW());

-- Teacher gets only view permissions

INSERT INTO role\_permissions (role\_id, permission, created\_at) VALUES

('3', 'view\_fee\_reports', NOW());

**Step 3: Create setup\_permissions\_ui() Method**

Add this method to your form class:

def setup\_permissions\_ui(self):

"""Set up UI elements based on user permissions"""

# Check permissions

can\_create = has\_permission(self.user\_session, "create\_fee")

can\_edit = has\_permission(self.user\_session, "edit\_fee")

can\_delete = has\_permission(self.user\_session, "delete\_fee")

can\_export = has\_permission(self.user\_session, "export\_fee\_data")

can\_process = has\_permission(self.user\_session, "process\_payments")

# Enable/disable buttons based on permissions

self.save\_btn.setEnabled(can\_create)

self.update\_btn.setEnabled(can\_edit)

self.delete\_btn.setEnabled(can\_delete)

self.export\_btn.setEnabled(can\_export)

self.process\_payment\_btn.setEnabled(can\_process)

# Set tooltips for disabled buttons

if not can\_create:

self.save\_btn.setToolTip("Permission required: create\_fee")

if not can\_edit:

self.update\_btn.setToolTip("Permission required: edit\_fee")

if not can\_delete:

self.delete\_btn.setToolTip("Permission required: delete\_fee")

if not can\_export:

self.export\_btn.setToolTip("Permission required: export\_fee\_data")

if not can\_process:

self.process\_payment\_btn.setToolTip("Permission required: process\_payments")

**Step 4: Call setup\_permissions\_ui() in init**

Add the call in your form's initialization:

def \_\_init\_\_(self, user\_session, db\_connection):

super().\_\_init\_\_()

self.user\_session = user\_session

self.db\_connection = db\_connection

# Your existing initialization code...

self.setup\_ui()

self.load\_data()

# Add this line

self.setup\_permissions\_ui() # ✅ Add permission setup

**Step 5: Add Permission Checks to Action Methods**

Add permission checks at the start of each action method:

def save\_fee(self):

"""Save new fee record"""

# ✅ Permission check first

if not has\_permission(self.user\_session, "create\_fee"):

QMessageBox.warning(self, "Permission Denied",

"You don't have permission to create fees.")

return

# Rest of your save logic...

def update\_fee(self):

"""Update existing fee record"""

# ✅ Permission check first

if not has\_permission(self.user\_session, "edit\_fee"):

QMessageBox.warning(self, "Permission Denied",

"You don't have permission to edit fees.")

return

# Rest of your update logic...

def delete\_fee(self):

"""Delete fee record"""

# ✅ Permission check first

if not has\_permission(self.user\_session, "delete\_fee"):

QMessageBox.warning(self, "Permission Denied",

"You don't have permission to delete fees.")

return

# Rest of your delete logic...

**Step 6: Optional - Add Menu/Tab Level Restrictions**

If you want to hide entire tabs or menu items:

def setup\_main\_menu\_permissions(self):

"""Hide/show menu items based on permissions"""

# Hide fees tab if user can't view fees

if not has\_permission(self.user\_session, "view\_fee\_reports"):

self.fees\_tab.setVisible(False)

# Disable menu items

if not has\_permission(self.user\_session, "create\_fee"):

self.action\_new\_fee.setEnabled(False)

**Summary Checklist for New Forms:**

1. ✅ **Identify permissions needed** (create, edit, delete, view, export, etc.)
2. ✅ **Add SQL INSERTs** for each role in role\_permissions table
3. ✅ **Create setup\_permissions\_ui() method** in your form
4. ✅ **Call setup\_permissions\_ui()** in \_\_init\_\_
5. ✅ **Add permission checks** to each action method (save, update, delete, etc.)
6. ✅ **Test with different user roles** to ensure permissions work

**Permission Naming Convention:**

* create\_[module] - for creating new records
* edit\_[module] - for updating existing records
* delete\_[module] - for deleting records
* view\_[module] - for viewing/reading data
* export\_[module] - for exporting data
* manage\_[specific\_feature] - for complex operations

This pattern ensures consistent permission handling across all your forms!