```python
#pip install pytesseract

import numpy as np
import cv2
import matplotlib.pyplot as plt
import re
import pytesseract
from pytesseract import Output
from skimage.filters import threshold_local
from PIL import Image
import pandas as pd
from datetime import datetime
import os
from google.colab.patches import cv2_imshow
import glob
import joblib
from sklearn.feature_extraction.text import TfidfVectorizer
import pickle

# Install required packages
!sudo apt install tesseract-ocr
!pip install pytesseract scikit-image openpyxl pandas joblib scikit-
learn

# Constants for file names
MASTER_EXCEL_FILE = "hotel_bills_database.xlsx"
MODEL_FILE = "hotel_bill_processor_model.pkl"

def load_and_preprocess_image(image_path):
    """Load and preprocess the image with enhanced preprocessing"""
    image = cv2.imread(image_path)
    if image is None:
        raise ValueError(f"Could not load image from {image_path}")

    # Convert to RGB
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Resize image if too large for better processing
    height, width = image.shape[:2]
    if height > 2000 or width > 2000:
        scale = min(2000/height, 2000/width)
        new_width = int(width * scale)
        new_height = int(height * scale)
        image = cv2.resize(image, (new_width, new_height),
interpolation=cv2.INTER_AREA)

    return image

def enhance_image_quality(image):
    """Enhanced image quality improvement for better OCR"""
    # Convert to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

    # Multiple enhancement techniques
    # 1. Denoising
    denoised = cv2.medianBlur(gray, 3)

    # 2. Contrast enhancement using CLAHE
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    contrast_enhanced = clahe.apply(denoised)

    # 3. Adaptive thresholding
    T = threshold_local(contrast_enhanced, 15, offset=12,
method="gaussian")
    binary = (contrast_enhanced > T).astype("uint8") * 255

    # 4. Morphological operations to clean up the image
    kernel = np.ones((2,2), np.uint8)
    binary = cv2.morphologyEx(binary, cv2.MORPH_CLOSE, kernel)
    binary = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)

    return binary, contrast_enhanced, gray

def extract_text_from_image(image):
    """Enhanced text extraction with multiple OCR configurations"""
    # Try different PSM modes for better results
    configs = [
        r'--oem 3 --psm 6',  # Uniform block of text
        r'--oem 3 --psm 4',  # Single column of text
        r'--oem 3 --psm 3',  # Fully automatic page segmentation
    ]

    best_text = ""
    best_config = ""

    for config in configs:
        try:
            text = pytesseract.image_to_string(image, config=config)
            if len(text) > len(best_text):
                best_text = text
                best_config = config
        except:
            continue

    # Get detailed data with best config
    detailed_data = pytesseract.image_to_data(image,
output_type=Output.DICT, config=best_config)

    return best_text, detailed_data
```

```
def extract_hotel_info(text):
    """Enhanced hotel name and address extraction"""
    lines = [line.strip() for line in text.split('\n') if
line.strip()]
    hotel_name = ""
    address = ""
    address_lines = []

    # Improved patterns for hotel name
    hotel_patterns = [
        r'^[A-Z][A-Za-z\s&\.\-]{3,}(?:Hotel|Resort|Inn|Suites|Lodge|
Motel|Plaza)$',
        r'^(?!.*(?:invoice|bill|receipt|date|total|tax|room|guest))[A-
Z][A-Za-z\s&\.\-]{3,}$'
    ]

    # Look for hotel name in first 10 lines
    for i, line in enumerate(lines[:10]):
        # Check if line matches hotel patterns
        for pattern in hotel_patterns:
            if re.match(pattern, line, re.IGNORECASE):
                if not hotel_name:
                    hotel_name = line
                    break

        # Address detection (usually follows hotel name)
        if hotel_name and i > lines.index(hotel_name) if hotel_name in
lines else i > 0:
            if re.search(r'\d+[\sA-Za-z]+,?[\sA-Za-z]+,?[\sA-Za-z]+',
line):
                address_lines.append(line)

    address = ' '.join(address_lines[:3])  # Take first 3 address
lines max

    return hotel_name, address

def extract_dates(text):
    """Enhanced date extraction with multiple formats"""
    date_patterns = [
        # MM/DD/YYYY or DD/MM/YYYY
        r'\b(0?[1-9]|1[0-2])[/-](0?[1-9]|[12][0-9]|3[01])[/-](\d{4}|\
d{2})\b',
        # DD-MM-YYYY or MM-DD-YYYY
        r'\b(0?[1-9]|[12][0-9]|3[01])[-/](0?[1-9]|1[0-2])[-/](\d{4}|\
d{2})\b',
        # Month name formats
        r'\b(0?[1-9]|1[0-2]|0?[1-9]|[12][0-9]|3[01])\s+(Jan|Feb|Mar|
Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[a-z]*\s+\d{4}\b',
        r'\b(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[a-z]*\
s+(0?[1-9]|[12][0-9]|3[01]),?\s+\d{4}\b',
    ]

    dates = []
    for pattern in date_patterns:
        found_dates = re.findall(pattern, text, re.IGNORECASE)
        # Convert tuples to strings
        for date_match in found_dates:
            if isinstance(date_match, tuple):
                date_str = ' '.join([str(part) for part in date_match
if part])
                dates.append(date_str)
            else:
                dates.append(date_match)

    # Remove duplicates while preserving order
    seen = set()
    unique_dates = []
    for date in dates:
        if date not in seen:
            seen.add(date)
            unique_dates.append(date)

    check_in = unique_dates[0] if len(unique_dates) > 0 else ""
    check_out = unique_dates[1] if len(unique_dates) > 1 else ""

    return check_in, check_out

def extract_guest_info(text):
    """Enhanced guest information extraction"""
    guest_name = ""
    room_number = ""

    # Improved guest name patterns
    guest_patterns = [
        r'(?:Guest|Name|Customer|Passenger)\s*:?\s*([A-Z][a-zA-Z]+(?:\
s+[A-Z][a-zA-Z\.]+)+)',
        r'(?:Guest Name|Customer Name)\s*:?\s*([A-Z][a-zA-Z]+\s+[A-Z]
[a-zA-Z]+)',
        r'Name\s+of\s+Guest\s*:?\s*([A-Z][a-zA-Z]+\s+[A-Z][a-zA-Z]+)',
    ]

    for pattern in guest_patterns:
        matches = re.findall(pattern, text, re.IGNORECASE)
        if matches:
            # Take the longest name (most complete)
            guest_name = max(matches, key=len)
            break

    # Enhanced room number patterns
    room_patterns = [
        r'(?:Room|Rm)\.?\s*(?:No|Number|#)?\.?\s*:?\s*([A-Z]?\d+[A-
Z]?)',
        r'Room\s+([A-Z]?\d+[A-Z]?)',
        r'Rm\s+([A-Z]?\d+[A-Z]?)',
        r'(?:Room|Rm)\s*:?\s*([A-Z]?\d+[A-Z]?)',
    ]

    for pattern in room_patterns:
        match = re.search(pattern, text, re.IGNORECASE)
        if match:
            room_number = match.group(1)
            break

    return guest_name, room_number

def extract_line_items(text):
    """Enhanced line item extraction with better filtering"""
    items = []
    lines = text.split('\n')

    # Improved patterns for line items
    item_patterns = [
        r'^([A-Za-z][A-Za-z\s\-&]+?)\s+([\$€£]?\s?\d{1,3}(?:[,.]\
d{3})*\.?\d{0,2})$',
        r'^([A-Za-z][A-Za-z\s\-&]+?)\s+([\$€£]?\s?\d+\.\d{2})$',
        r'(.+?)\s+(\$?\d+[,.]?\d*\.?\d{2})\s*$'
    ]

    # Keywords to exclude
    exclude_keywords = ['total', 'subtotal', 'tax', 'vat', 'gst',
'balance', 'amount due', 'grand total']

    for line in lines:
        line = line.strip()

        # Skip lines with exclude keywords
        if any(keyword in line.lower() for keyword in
exclude_keywords):
            continue

        # Skip very short lines or lines that are likely headers
        if len(line) < 4 or line.isupper():
            continue

        for pattern in item_patterns:
            match = re.search(pattern, line)
            if match:
                description = match.group(1).strip()
                amount_str = match.group(2).strip()
                # Clean amount
                amount_clean = re.sub(r'[^\d.]', '', amount_str)

                # Validate description
                if (len(description) > 2 and
                    description[0].isalpha() and
                    not any(word in description.lower() for word in
exclude_keywords)):

                    try:
                        amount_float = float(amount_clean)
                        if 0.01 <= amount_float <= 10000:  #
Reasonable amount range
                            items.append({
                                'description': description,
                                'amount': amount_float
                            })
                            break  # Stop checking other patterns for
this line
                    except ValueError:
                        continue

    return items

def extract_totals(text):
    """Enhanced financial totals extraction"""
    subtotal = 0.0
    tax = 0.0
    total = 0.0

    # Improved patterns for financial amounts
    patterns = {
        'subtotal': [
            r'(?:Sub\s*Total|Subtotal)\s*:?\s*[\$€£]?\s*(\d{1,3}(?:
[,.]\d{3})*\.?\d{0,2})',
            r'(?:Sub\s*Total|Subtotal)\s*[\$€£]?\s*(\d{1,3}(?:[,.]\
d{3})*\.?\d{0,2})'
        ],
        'tax': [
            r'(?:Tax|GST|VAT)\s*:?\s*[\$€£]?\s*(\d{1,3}(?:[,.]\
d{3})*\.?\d{0,2})',
            r'(?:Tax|GST|VAT)\s*[\$€£]?\s*(\d{1,3}(?:[,.]\d{3})*\.?\
d{0,2})'
        ],
        'total': [
            r'(?:Total|Grand\s*Total|Amount\s*Due|Balance\s*Due)\s*:?\
s*[\$€£]?\s*(\d{1,3}(?:[,.]\d{3})*\.?\d{0,2})',
            r'(?:Total|Grand\s*Total|Amount\s*Due|Balance)\s*[\$€£]?\
s*(\d{1,3}(?:[,.]\d{3})*\.?\d{0,2})'
```

```
        ]
    }

    for key, pattern_list in patterns.items():
        for pattern in pattern_list:
            matches = re.findall(pattern, text, re.IGNORECASE)
            if matches:
                # Take the last occurrence (usually the final amount)
                value_str = matches[-1].replace(',', '').replace(' ',
'')
                try:
                    value_float = float(value_str)
                    if key == 'subtotal':
                        subtotal = value_float
                    elif key == 'tax':
                        tax = value_float
                    elif key == 'total':
                        total = value_float
                except ValueError:
                    continue

    return subtotal, tax, total

def extract_invoice_number(text):
    """Enhanced invoice number extraction"""
    patterns = [
        r'(?:Invoice|Bill|Receipt)\s*(?:No|Number|#)?\.?\s*:?\s*([A-
Z0-9\-/#]+)',
        r'(?:Folio|Confirmation)\s*(?:No|Number|#)?\.?\s*:?\s*([A-Z0-
9\-/#]+)',
        r'(?:Invoice|Bill)\s*#?\s*:?\s*([A-Z0-9\-/#]+)',
        r'[A-Z]{2,}\d{4,}',  # Pattern like INVOICE1234
    ]

    for pattern in patterns:
        matches = re.findall(pattern, text, re.IGNORECASE)
        if matches:
            return matches[0]

    return ""

class HotelBillProcessor:
    """Model class to handle hotel bill processing and saving"""

    def __init__(self):
        self.vectorizer = TfidfVectorizer(max_features=1000)
        self.processed_data = []
        self.model_version = "1.0"
        self.is_fitted = False
    def load_existing_model(self):
        """Load existing model if available"""
        if os.path.exists(MODEL_FILE):
            try:
                with open(MODEL_FILE, 'rb') as f:
                    model_data = pickle.load(f)
                self.vectorizer = model_data['vectorizer']
                self.processed_data = model_data['processed_data']
                self.model_version = model_data['model_version']
                self.is_fitted = model_data.get('is_fitted', False)
                print(f"□ Loaded existing model with
{len(self.processed_data)} records")
                return True
            except Exception as e:
                print(f"⚠ Could not load existing model: {str(e)}")
                return False
        return False

    def fit(self, texts):
        """Fit the vectorizer on text data"""
        if texts:
            try:
                self.vectorizer.fit(texts)
                self.is_fitted = True
            except Exception as e:
                print(f"⚠ Error fitting vectorizer: {str(e)}")

    def add_data(self, hotel_name, raw_text, image_path):
        """Add new data to the processor"""
        self.processed_data.append({
            'hotel_name': hotel_name,
            'raw_text': raw_text,
            'image_path': image_path,
            'timestamp': datetime.now()
        })

    def save_model(self, filename=MODEL_FILE):
        """Save the trained model"""
        model_data = {
            'vectorizer': self.vectorizer,
            'processed_data': self.processed_data,
            'model_version': self.model_version,
            'is_fitted': self.is_fitted,
            'timestamp': datetime.now()
        }

        with open(filename, 'wb') as f:
            pickle.dump(model_data, f)

        print(f"□ Model saved as {filename} with
{len(self.processed_data)} total records")
        return filename

def process_hotel_bill(image_path, processor=None):
    """Main function to process hotel bill and extract structured
data"""
    print("Loading image...")
    original_image = load_and_preprocess_image(image_path)

    print("Enhancing image quality...")
    binary_image, contrast_enhanced, gray_image =
enhance_image_quality(original_image)

    print("Extracting text from enhanced images...")
    text_binary, _ = extract_text_from_image(binary_image)
    text_contrast, _ = extract_text_from_image(contrast_enhanced)
    text_gray, _ = extract_text_from_image(gray_image)

    # Use the best result
    texts = [text_binary, text_contrast, text_gray]
    final_text = max(texts, key=len)

    print("Extracting structured data...")

    # Extract all information
    hotel_name, address = extract_hotel_info(final_text)
    check_in, check_out = extract_dates(final_text)
    guest_name, room_number = extract_guest_info(final_text)
    invoice_number = extract_invoice_number(final_text)
    line_items = extract_line_items(final_text)
    subtotal, tax, total = extract_totals(final_text)

    # Store in processor if provided
    if processor:
        processor.add_data(hotel_name, final_text, image_path)

    # Display extracted text
    print("\n=== EXTRACTED TEXT ===")
    print("-" * 50)
    print(final_text)
    print("-" * 50)

    # Display images
    fig, axes = plt.subplots(2, 2, figsize=(15, 10))

    axes[0,0].imshow(original_image)
    axes[0,0].set_title('Original Image')
    axes[0,0].axis('off')

    axes[0,1].imshow(gray_image, cmap='gray')
    axes[0,1].set_title('Grayscale Image')
    axes[0,1].axis('off')

    axes[1,0].imshow(contrast_enhanced, cmap='gray')
    axes[1,0].set_title('Contrast Enhanced')
    axes[1,0].axis('off')

    axes[1,1].imshow(binary_image, cmap='gray')
    axes[1,1].set_title('Binary Image')
    axes[1,1].axis('off')

    plt.tight_layout()
    plt.show()

    return {
        'hotel_name': hotel_name,
        'address': address,
        'invoice_number': invoice_number,
        'guest_name': guest_name,
        'room_number': room_number,
        'check_in_date': check_in,
        'check_out_date': check_out,
        'line_items': line_items,
        'subtotal': subtotal,
        'tax': tax,
        'total': total,
        'raw_text': final_text,
        'processing_date': datetime.now().strftime("%Y-%m-%d %H:%M:
%S"),
        'image_filename': os.path.basename(image_path)
    }

def append_to_master_excel(result):
    """Append new bill data to the master Excel file"""

    # Prepare summary record
    summary_record = {
        'Processing Date': result['processing_date'],
        'Hotel Name': result['hotel_name'],
        'Address': result['address'],
        'Guest Name': result['guest_name'],
        'Room Number': result['room_number'],
        'Invoice Number': result['invoice_number'],
        'Check-in Date': result['check_in_date'],
        'Check-out Date': result['check_out_date'],
        'Subtotal': result['subtotal'],
        'Tax': result['tax'],
        'Total Amount': result['total'],
        'Image Filename': result['image_filename']
    }
    # Prepare line items
    line_items_records = []
    for item in result['line_items']:
        line_items_records.append({
            'Processing Date': result['processing_date'],
            'Hotel Name': result['hotel_name'],
            'Guest Name': result['guest_name'],
            'Invoice Number': result['invoice_number'],
            'Description': item['description'],
            'Amount': item['amount']
        })

    # Check if file exists
    if os.path.exists(MASTER_EXCEL_FILE):
        print(f"□ Appending to existing file: {MASTER_EXCEL_FILE}")

        # Read existing data
        try:
            existing_summary = pd.read_excel(MASTER_EXCEL_FILE,
sheet_name='All Bills Summary')
            existing_items = pd.read_excel(MASTER_EXCEL_FILE,
sheet_name='All Line Items')
        except Exception as e:
            print(f"⚠ Error reading existing file: {str(e)}")
            existing_summary = pd.DataFrame()
            existing_items = pd.DataFrame()

        # Append new data
        new_summary_df = pd.DataFrame([summary_record])
        updated_summary = pd.concat([existing_summary,
new_summary_df], ignore_index=True)

        if line_items_records:
            new_items_df = pd.DataFrame(line_items_records)
            updated_items = pd.concat([existing_items, new_items_df],
ignore_index=True)
        else:
            updated_items = existing_items
    else:
        print(f"□ Creating new file: {MASTER_EXCEL_FILE}")
        updated_summary = pd.DataFrame([summary_record])
        updated_items = pd.DataFrame(line_items_records) if
line_items_records else pd.DataFrame()

    # Save to Excel
    with pd.ExcelWriter(MASTER_EXCEL_FILE, engine='openpyxl') as
writer:
        updated_summary.to_excel(writer, sheet_name='All Bills
Summary', index=False)
```

```
        if not updated_items.empty:
            updated_items.to_excel(writer, sheet_name='All Line
Items', index=False)
        else:
            pd.DataFrame({'Processing Date': [], 'Hotel Name': [],
'Guest Name': [],
                         'Invoice Number': [], 'Description': [],
'Amount': []}).to_excel(
                writer, sheet_name='All Line Items', index=False)

    print(f"□ Data appended to {MASTER_EXCEL_FILE}")

    # Get total count
    total_bills = len(updated_summary)
    print(f"□ Total bills in database: {total_bills}")

    return MASTER_EXCEL_FILE

def display_extracted_summary(results):
    """Display a clean summary of extracted data"""
    print("\n" + "="*50)
    print("□ EXTRACTED DATA SUMMARY")
    print("="*50)
    print(f"□ Hotel Name: {results['hotel_name'] or 'Not found'}")
    print(f"□ Address: {results['address'] or 'Not found'}")
    print(f"□ Guest Name: {results['guest_name'] or 'Not found'}")
    print(f"□ Room Number: {results['room_number'] or 'Not found'}")
    print(f"□ Invoice Number: {results['invoice_number'] or 'Not
found'}")
    print(f"□ Check-in: {results['check_in_date'] or 'Not found'}")
    print(f"□ Check-out: {results['check_out_date'] or 'Not found'}")
    print(f"□ Financial Summary:")
    print(f"   • Subtotal: ${results['subtotal']:.2f}" if
results['subtotal'] else "   • Subtotal: Not found")
    print(f"   • Tax: ${results['tax']:.2f}" if results['tax'] else "
• Tax: Not found")
    print(f"   • Total: ${results['total']:.2f}" if results['total']
else "   • Total: Not found")
    print(f"□ Line Items Found: {len(results['line_items'])}")
    if results['line_items']:
        for item in results['line_items'][:5]:  # Show first 5 items
            print(f"     - {item['description']}: $
{item['amount']:.2f}")
    print(f"□ Processed: {results['processing_date']}")
    print("="*50)

# Main execution for Colab
if __name__ == "__main__":
    from google.colab import files
```

```
import time

print("□ HOTEL BILL PROCESSING SYSTEM (PERSISTENT STORAGE)")
print("="*50)

# Initialize the processor model
processor = HotelBillProcessor()

# Load existing model if available
processor.load_existing_model()

print("\n□ Please upload your hotel bill image(s)...")
uploaded = files.upload()

image_files = list(uploaded.keys())
all_results = []

if image_files:
    print(f"\n□ Found {len(image_files)} image(s) to process:")
    for i, image_file in enumerate(image_files, 1):
        print(f"  {i}. {image_file}")

    print("\n" + "="*50)

    for image_path in image_files:
        print(f"\n□ Processing: {image_path}")

        try:
            # Process the image and extract data
            results = process_hotel_bill(image_path, processor)
            all_results.append(results)

            # Display summary
            display_extracted_summary(results)

            # Append to master Excel file
            append_to_master_excel(results)

            # Small delay between processing
            time.sleep(1)

        except Exception as e:
            print(f"□ Error processing {image_path}: {str(e)}")
            import traceback
            traceback.print_exc()
            continue

    # Fit/Update the processor model with all extracted texts
    if all_results:
        all_texts = [result['raw_text'] for result in all_results]
```

```
            # Combine with existing texts if model was loaded
            if processor.processed_data:
                existing_texts = [data['raw_text'] for data in
processor.processed_data]
                all_texts = existing_texts + all_texts

            processor.fit(all_texts)

            # Save the updated model
            model_filename = processor.save_model()

            print(f"\n□ Downloading files...")

            # Download the master Excel file
            if os.path.exists(MASTER_EXCEL_FILE):
                files.download(MASTER_EXCEL_FILE)
                print(f"□ Downloaded: {MASTER_EXCEL_FILE}")

            # Download the model file
            if os.path.exists(MODEL_FILE):
                files.download(MODEL_FILE)
                print(f"□ Downloaded: {MODEL_FILE}")

        print(f"\n{'='*50}")
        print(f"□ PROCESSING COMPLETED!")
        print(f"{'='*50}")
        print(f"□ Bills processed in this session:
{len(all_results)}")
        print(f"🤖 Model file: {MODEL_FILE}")
        print(f"□ Database file: {MASTER_EXCEL_FILE}")
        print(f"\n□ TIP: Keep both files. Next time you upload new
bills,")
        print(f"   upload these files first to append new data!")
        print(f"{'='*50}")

    else:
        print("□ No files uploaded.")
```

```
🏨 HOTEL BILL PROCESSING SYSTEM (PERSISTENT STORAGE)
==================================================

📤 Please upload your hotel bill image(s)...
```
Choose Files Image.JPG

**Image.JPG**(image/jpeg) - 55257 bytes, last modified: 23/11/2025 - 100% done

```
Saving Image.JPG to Image.JPG

📁 Found 1 image(s) to process:
  1. Image.JPG

==================================================

🔄 Processing: Image.JPG
Loading image...
Enhancing image quality...
Extracting text from enhanced images...
Extracting structured data...

 === EXTRACTED TEXT ===
 --------------------------------------------------
 TOM GREEN HANDYMAN
 5 Any Street, Any City, That Area Code
 Telephone: 0800 XXX XXX

 Date: 6/5/2016 Invoice No : 0003521
 Tax RegisteredNo 123456

 Mr and Mrs Fielding

 This Address

 This City

 This Area Code

 TAX INVOICE
 ‘Quan Description Unit Price Cost
 Upgrade to Bathroom


 23.75 Labour 40.00 950.00
 50 Nails and screws 080 40.00
 1 Paint and Plywood 1000.00 1000.00
 40 Imported wall tiles 1400 560.00
 1 Freight 150.00 150.00
 1 Sub-contractor : Tile-tt 228.00

 Subtotal 2928.00

 Tax 439.20



 Total Due[ $3,367.20
```

Payment due by the 10th of the month following the date of invoice.
Please make payment into Bank Account No. 12 3456 789112 012

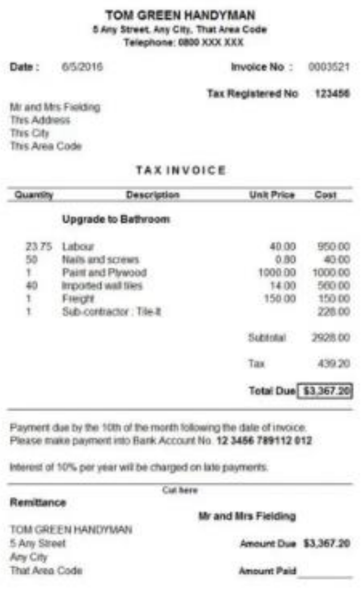
Interest of 10% per year will be charged on late payments,

Cath
Remittance

Mr and Mrs Fielding
TOM GREEN HANDYMAN
5 Any Street Amount Due $3,367.20
Any City

‘That Area Code Amount Paid

Original Image



**TOM GREEN HANDYMAN**
**5 Any Street, Any City, That Area Code**
**Telephone: 0800 XXX XXX**

**Date :** 6/5/2016 **Invoice No :** 0003521

**Tax Registered No** **123456**

Mr and Mrs Fielding
This Address
This City
This Area Code

**TAX INVOICE**

| Quantity | Description | Unit Price | Cost |
|---|---|---|---|
| | **Upgrade to Bathroom** | | |
| 23.75 | Labour | 40.00 | 950.00 |
| 50 | Nails and screws | 0.80 | 40.00 |
| 1 | Paint and Plywood | 1000.00 | 1000.00 |
| 40 | Imported wall tiles | 14.00 | 560.00 |
| 1 | Freight | 150.00 | 150.00 |
| 1 | Sub-contractor : Tile-It | | 228.00 |
| | | Subtotal | 2928.00 |
| | | Tax | 439.20 |
| | | **Total Due** | **$3,367.20** |

Payment due by the 10th of the month following the date of invoice.
Please make payment into Bank Account No. **12 3456 789112 012**

Interest of 10% per year will be charged on late payments.

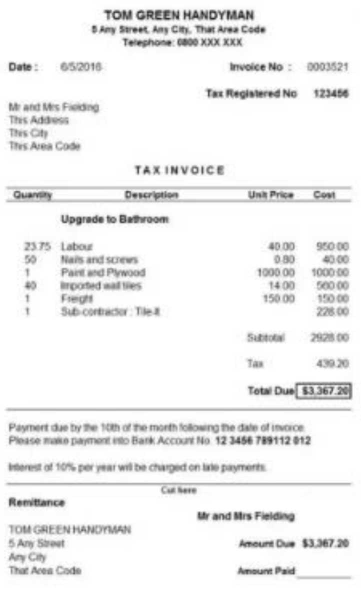
Cut here

**Remittance**

**Mr and Mrs Fielding**

TOM GREEN HANDYMAN
5 Any Street **Amount Due $3,367.20**
Any City
That Area Code **Amount Paid** ______

Grayscale Image



**TOM GREEN HANDYMAN**
**5 Any Street, Any City, That Area Code**
**Telephone: 0800 XXX XXX**

**Date :** 6/5/2016 **Invoice No :** 0003521

**Tax Registered No** **123456**

Mr and Mrs Fielding
This Address
This City
This Area Code

**TAX INVOICE**

| Quantity | Description | Unit Price | Cost |
|---|---|---|---|
| | **Upgrade to Bathroom** | | |
| 23.75 | Labour | 40.00 | 950.00 |
| 50 | Nails and screws | 0.80 | 40.00 |
| 1 | Paint and Plywood | 1000.00 | 1000.00 |
| 40 | Imported wall tiles | 14.00 | 560.00 |
| 1 | Freight | 150.00 | 150.00 |
| 1 | Sub-contractor : Tile-It | | 228.00 |
| | | Subtotal | 2928.00 |
| | | Tax | 439.20 |
| | | **Total Due** | **$3,367.20** |

Payment due by the 10th of the month following the date of invoice.
Please make payment into Bank Account No. **12 3456 789112 012**

Interest of 10% per year will be charged on late payments.

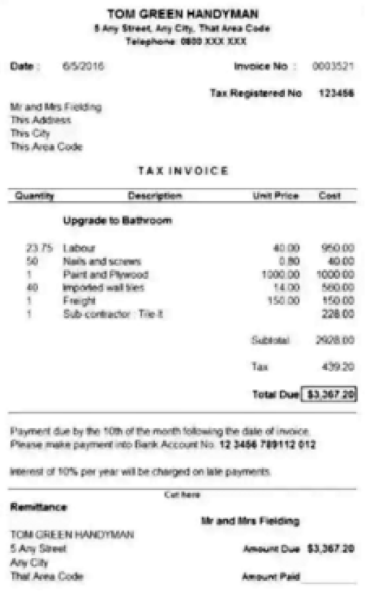
Cut here

**Remittance**

**Mr and Mrs Fielding**

TOM GREEN HANDYMAN
5 Any Street **Amount Due $3,367.20**
Any City
That Area Code **Amount Paid** ______

Contrast Enhanced



**TOM GREEN HANDYMAN**
**5 Any Street, Any City, That Area Code**
**Telephone: 0800 XXX XXX**

**Date :** 6/5/2016 **Invoice No :** 0003521

**Tax Registered No** **123456**

Mr and Mrs Fielding
This Address
This City
This Area Code

**TAX INVOICE**

| Quantity | Description | Unit Price | Cost |
|---|---|---|---|
| | **Upgrade to Bathroom** | | |
| 23.75 | Labour | 40.00 | 950.00 |
| 50 | Nails and screws | 0.80 | 40.00 |
| 1 | Paint and Plywood | 1000.00 | 1000.00 |
| 40 | Imported wall tiles | 14.00 | 560.00 |
| 1 | Freight | 150.00 | 150.00 |
| 1 | Sub-contractor : Tile-It | | 228.00 |
| | | Subtotal | 2928.00 |
| | | Tax | 439.20 |
| | | **Total Due** | **$3,367.20** |

Payment due by the 10th of the month following the date of invoice.
Please make payment into Bank Account No. **12 3456 789112 012**

Interest of 10% per year will be charged on late payments.

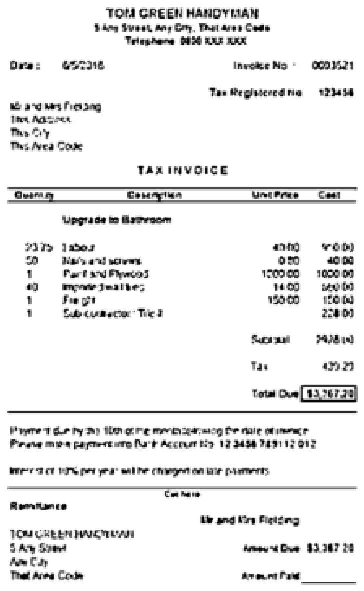
Cut here

**Remittance**

**Mr and Mrs Fielding**

TOM GREEN HANDYMAN
5 Any Street **Amount Due $3,367.20**
Any City
That Area Code **Amount Paid** ______

Binary Image



**TOM GREEN HANDYMAN**
**5 Any Street, Any City, That Area Code**
**Telephone: 0800 XXX XXX**

**Date :** 6/5/2016 **Invoice No :** 0003521

**Tax Registered No** **123456**

Mr and Mrs Fielding
This Address
This City
This Area Code

**TAX INVOICE**

| Quantity | Description | Unit Price | Cost |
|---|---|---|---|
| | **Upgrade to Bathroom** | | |
| 23.75 | Labour | 40.00 | 950.00 |
| 50 | Nails and screws | 0.80 | 40.00 |
| 1 | Paint and Plywood | 1000.00 | 1000.00 |
| 40 | Imported wall tiles | 14.00 | 560.00 |
| 1 | Freight | 150.00 | 150.00 |
| 1 | Sub-contractor : Tile-It | | 228.00 |
| | | Subtotal | 2928.00 |
| | | Tax | 439.20 |
| | | **Total Due** | **$3,367.20** |

Payment due by the 10th of the month following the date of invoice.
Please make payment into Bank Account No. **12 3456 789112 012**

Interest of 10% per year will be charged on late payments.

Cut here

**Remittance**

**Mr and Mrs Fielding**

TOM GREEN HANDYMAN
5 Any Street **Amount Due $3,367.20**
Any City
That Area Code **Amount Paid** ______

```
==================================================
📋 EXTRACTED DATA SUMMARY
==================================================
🏨 Hotel Name: TOM GREEN HANDYMAN
📍 Address: 5 Any Street, Any City, That Area Code Telephone: 0800 XXX XXX Date: 6/5/2016 Invoice No : 0003521
👤 Guest Name: Not found
🚪 Room Number: 23
📄 Invoice Number: 0003521
📅 Check-in: 6 5 2016
📅 Check-out: Not found
💰 Financial Summary:
   • Subtotal: $2928.00
   • Tax: $439.20
   • Total: $3367.20
📦 Line Items Found: 2
     - Date: 6/5/2016 Invoice No :: $3521.00
     - Please make payment into Bank Account No. 12 3456 789112: $12.00
🕒 Processed: 2025-11-24 07:59:36
==================================================
📄 Creating new file: hotel_bills_database.xlsx
✅ Data appended to hotel_bills_database.xlsx
📊 Total bills in database: 1
✅ Model saved as hotel_bill_processor_model.pkl with 1 total records


📥 Downloading files...
✅ Downloaded: hotel_bills_database.xlsx
✅ Downloaded: hotel_bill_processor_model.pkl

==================================================
✅ PROCESSING COMPLETED!
==================================================
📊 Bills processed in this session: 1
🤖 Model file: hotel_bill_processor_model.pkl
🗄️ Database file: hotel_bills_database.xlsx

💡 TIP: Keep both files. Next time you upload new bills,
   upload these files first to append new data!
==================================================
```