Oracle Forms Webcam Integration for Image Capture and Storage

# 1. Project Overview

This project aims to develop an Oracle Forms application that integrates with a webcam to capture images and store them in an Oracle database. The primary goal of the form is to facilitate image capture and dynamic storage, allowing users to save images as BLOBs in any Oracle database table. This solution uses Oracle Forms’ built-in features such as the **HOST command** to invoke external programs, alongside **Java-based scripts** for image capture via a webcam using **OpenCV**.

The form is designed to be modular and database-agnostic, thus being able to be plugged into any Oracle database setup with minor adjustments. The application will save the captured images with dynamically generated filenames based on the record's ID and timestamp, ensuring that each image is uniquely identified and stored, both in a BLOB and an **optional archive folder** on the shared network.

## Purpose

* Capture webcam images and associate them with records in an Oracle database.
* Store images as BLOB data in the database.
* Optionally store images in an archive folder for local or network-based storage.
* Retrieve and export images to a local or network file system for further processing.
* Enable dynamic configuration for database table names, columns, and other fields, making the form reusable across different database setups.

# 2. System Requirements:

### 2.1 Software Requirements:

* **Oracle Forms Builder**: Version 12.2.x.xxx
* **Java Development Kit (JDK)**: Version 1.8.
* **OpenCV**: Version 4.1.1 (or earlier) for webcam capture.
* **Operating System**: The application should be hosted on a shared network or server environment that supports Oracle Forms and has access to the required Java and OpenCV libraries.
* **Oracle Database**: The form should connect to an Oracle database with the capability to store images as BLOBs in the specified table.

### **2.2 Paths for Execution**:

The form relies on specific global paths for Java execution and the library used for capturing images:

**Java Executable Path** (assuming Java is installed on a shared network):

| **:global.java\_exec** := *'\\sharednetworkpath\Java\jdk1.8.0\_281\bin\javaw.exe'*; |
| --- |

**Java Library Path for OpenCV**:

| **:global.java\_lib\_path** := *'\\sharednetworkpath\FormsScripts\JavaCapture'*; |
| --- |

**Note**: The Java installation and OpenCV libraries may be hosted on a shared network rather than installed on individual machines. Shared network access is required for the form to invoke Java, OpenCV, and other necessary components.

### 

### **2.3 Permissions**:

* **Host Access**: The form must have access to the **HOST command** (cmd) on any machine that will be using the application. This is essential for invoking external Java programs to handle webcam capture.
* **BLOB Creation and Storage**: The form should have permissions to create and store **BLOBs** in the Oracle database. This is optional for cases where the archive folder is being used to store images.
* **Shared Network Access**: The form should be able to access a shared network location for invoking both Java, OpenCV, and other necessary components, as well as storing image files in the archive folder, if enabled.

# 3. Technical Design

### 3.1 System Architecture

* **Client-Side**: Oracle Forms, running on a shared network or server, provides the user interface. The form uses a "Capture" button to trigger a Java-based image capture program via the HOST command.
* **Server-Side**: The form communicates with an Oracle Database, saving images as BLOBs in the specified table. Optionally, images are stored in an archive folder on a shared network for backup purposes.

### 3.2 Key Components

1. **Oracle Forms**:
   * **Form Layout**: The form includes fields for client information and an image item to display the captured image.
   * **Capture Button**: Invokes the image capture process.
   * **Dynamic Configuration**: Global variables are used to dynamically set Java paths, library paths, block names, and image item names.
2. **Java and OpenCV**:
   * **Java Program**: A Java program utilizing OpenCV is executed via Oracle Forms using the HOST command to capture images from a connected webcam.
   * **Image Processing**: The captured image is saved in a temporary directoryl.
3. **Database and Image Storage**:
   * **BLOB Storage**: Images are stored as BLOBs in the Oracle database. A table with a BLOB column is used to store images along with other client details.
   * **Dynamic Database Handling**: The table name and column names are dynamically assigned via global variables, making the system flexible and reusable across different databases.
4. **Archive Folder**:
   * **Optional Backup**: Images can optionally be saved in an archive folder on a shared network for backup purposes.

### 3.3 Capture Button Trigger Logic

When the user clicks the "Capture" button, the following actions are executed:

1. **Invoke Java Program**:
   * The form uses the HOST command to invoke the Java-based webcam capture program.

| HOST(**:global.java\_exec** || ' -Djava.library.path="' || **:global.java\_lib\_path** || '" -cp "' || **:global.java\_lib\_path** || '\opencv-411.jar;' || **:global.java\_lib\_path** || '" WebcamCapture'); |
| --- |

1. **Retrieve Image Path**:
   * The Java program captures the image and writes the image to a temporary directory.
   * The Oracle Forms application reads the image path from this file using UTL\_FILE.
2. **Display Image in the Form**:
   * The image is dynamically loaded into the form’s image item using the file path.

**Example Code:**

| READ\_IMAGE\_FILE(**v\_file\_path**, 'JPEG', **:global**.block\_name || '.' || **:global**.image\_item); |
| --- |

1. **Store Image in Database**:
   * The captured image is saved as a BLOB in the Oracle database.
   * Optional: The image is also saved in an archive folder for backup.

### 

### 3.4 Global Variables and Configuration

The following global variables are set when a new form instance is created:

**Java Executable Path**: :global**.java\_exec**

**:global.java\_exec** := 'C:\Program Files\Java\jdk1.8.0\_281\bin\javaw.exe';

**Java Library Path**: :global.java\_lib\_path

| **:global.java\_lib\_path** := *'C:\FormsScripts\JavaCapture'*; |
| --- |

**Temporary Directory**: :global.temp\_dir

| **:global.**temp\_dir := *'C:\Temp'*; |
| --- |

**Image Filename:** :global.image\_file

| **:global.**image\_file := *'webcam\_capture.jpg'*; -- Direct reference to the image file |
| --- |

**Database Column for Image**: :global.image\_column

| **:global.**image\_column := *'pl\_image'*; |
| --- |

**Dynamic Block and Image Item Names**:

| **:global.**block\_name := *'BC\_IRRADIATION\_DTL'*; |
| --- |

| **:global**.image\_item := *'PL\_IMAGE'*; |
| --- |

### 3.5 Error Handling

* **File Read Errors**: If there is an issue reading the image path file, the system will display an error message and halt the image-loading process.
* **Capture Errors**: Errors encountered during the Java-based image capture process will be logged, and the user will be notified of any issues.

# 4. Implementation Steps

This section provides a step-by-step guide to implementing the Oracle Forms Webcam Integration project. It covers the setup of Oracle Forms, Java, OpenCV, and the database configuration for storing captured images.

### 4.1 Setup of Oracle Forms

**Step 1:** Edit the Existing Form Layout

1. **Open the existing Oracle Forms application** in Oracle Forms Builder (12.2.x.xxx).
2. **Add the following components** to the form:
   * **Image Field**: A form item to display the captured image.
   * **Capture Button**: A button that triggers the image capture process when clicked.

### 4.2 Java and OpenCV Setup

**Step 2:** Set Up Java and OpenCV on the Shared Network

1. **Install JDK 1.8 on a shared network accessible by all users.**
2. **Install OpenCV (version 4.1.1 or earlier) and place the necessary Java bindings and OpenCV .dll/.so files in the shared network folder.**
3. **Compile the Java Program (using javac) and place the .class file in the shared network folder if not already present.**

### 4.3 Oracle Forms and Java Integration

**Step 3:** Capture Button Logic

In the existing form, set up the **WHEN-BUTTON-PRESSED trigger** for the "**Capture**" button. This trigger will invoke the Java program to capture the image:

| **DECLARE  v\_file\_path VARCHAR2(1000); BEGIN  -- Execute the Java program using global variables for Java paths and library paths  HOST(:global.java\_exec || ' -Djava.library.path="' || :global.java\_lib\_path || '" -cp "' || :global.java\_lib\_path || '\opencv-411.jar;' || :global.java\_lib\_path || '" WebcamCapture');   -- Construct the full file path for the image  v\_file\_path := :global.temp\_dir || '\' || :global.image\_file;    -- Load the image into the dynamically named block and item  IF v\_file\_path IS NOT NULL THEN  -- Use the constructed file path to load the image  READ\_IMAGE\_FILE(v\_file\_path, 'JPEG', :global.block\_name || '.' || :global.image\_item);    -- Log successful loading  MESSAGE('Image loaded successfully, please save to push changes....');  ELSE  MESSAGE('No image selected.');  END IF; END;** |
| --- |

### 4.4 Global Variables Setup

**Step 4:** Initialize Global Variables

**In the Form-Level Trigger (e.g., WHEN-NEW-FORM-INSTANCE), define the following global variables to manage Java paths, library paths, and image processing:**

| **BEGIN**  **-- Set the Java executable and library path**  **:global.java\_exec := 'C:\Program Files\Java\jdk1.8.0\_281\bin\javaw.exe';**  **:global.java\_lib\_path := 'C:\FormsScripts\JavaCapture';**    **-- Set the static image file path**  **:global.temp\_dir := 'C:\Temp';**  **:global.image\_file := 'webcam\_capture.jpg';**  **-- Set the block and image item dynamically for flexibility**  **:global.block\_name := 'BC\_IRRADIATION\_DTL';**  **:global.image\_item := 'PL\_IMAGE';**      **-- Ensure this matches the column name in the SQL Table**  **:global.image\_column := 'pl\_image';**  **END;** |
| --- |

### 4.5 Adding BLOB Image Save-to-File Functionality

This section explains how to add the functionality that saves captured images (stored as BLOBs) to a file on the shared drive, using the save\_blob\_to\_file procedure. This requires configuring the **UTL\_FILE** package and invoking the save logic after a record is committed to the database.

**Step 1**: Configure UTL\_FILE for the Database

1. Create the Oracle Directory: The first step is to create a directory object in the database, where the files will be stored.

| **CREATE OR REPLACE DIRECTORY ARCHIVE\_DIR AS '\\sharednetworkpath\ImageArchive';** |
| --- |

1. **Grant Permissions: Grant the necessary read/write privileges on this directory to the schema.**

| **GRANT READ, WRITE ON DIRECTORY ARCHIVE\_DIR TO your\_schema;** |
| --- |

**Step 2: Add the WHEN-DATABASE-COMMIT Trigger**

In the form, set up the WHEN-DATABASE-COMMIT trigger to invoke the save\_blob\_to\_file procedure. This trigger ensures that the BLOB data is saved to a file only after the form data is successfully committed to the database.

| **DECLARE  v\_file\_name VARCHAR2(256); BEGIN  -- Call the stored procedure to save the BLOB to a file on the shared network  save\_blob\_to\_file(  p\_id => :bc\_irradiation\_dtl.id, -- ID of the current form record  p\_blob\_column => :global.image\_column, -- Dynamic BLOB column (configured via global variables)  p\_table\_name => :global.block\_name, -- Dynamic table name (configured via global variables)  p\_file\_name => v\_file\_name -- Output: Generated file name for saving  );  EXCEPTION  WHEN OTHERS THEN  MESSAGE('Error: ' || SQLERRM);  PAUSE; END;** |
| --- |

**Important:** The field names (e.g., :bc\_irradiation\_dtl.id) are placeholders and should match the actual field names in the form.

**Step 3: Save BLOB to File Procedure**

The full **save\_blob\_to\_file** procedure, which saves the image from the database to the shared drive. Can be found in the package dir, and may require changes depending on your schema (e.g., column names, BLOB storage specifics).

#### Emphasizing Customization:

* The **column names** (p\_blob\_column, p\_table\_name, etc.) are dynamic, but they must match your specific schema. Be sure to adjust these values based on your actual form configuration.
* The **ID** and **table structure** are highly dependent on how your Oracle Forms application is set up. Ensure that the correct values are passed to the procedure.

# 5. Testing and Final Adjustments

Testing ensures that all components (Oracle Forms, Java program, OpenCV, and database operations) work together as expected. This section outlines the testing approach, key areas to focus on, and potential adjustments that might be needed based on test outcomes.

### 5.1 Initial Testing

1. **Webcam Capture Test**:
   * Test the **Capture Button** to verify that the Java program (executed via the HOST command) captures the image using the webcam.
   * Ensure that the **captured image** is stored in the temporary folder and correctly loaded into the Oracle Form's image field.
   * **Check that the image is loaded** correctly into the form and displays without issues.
2. **Test Cases**:
   * Ensure that clicking the capture button executes the Java program and captures an image.
   * Check if the image is displayed correctly within the form.
   * Validate error messages if the image cannot be captured or loaded.
3. **BLOB Storage Test (Optional)**:
   * For those implementing the BLOB storage, check if the image is saved as a BLOB in the database.
   * Ensure the **BLOB column** is updated with the image for the corresponding record.
   * Validate that no image is saved when there is no valid capture.
4. **Test Cases**:
   * Confirm that the image is stored in the BLOB column after being captured.
   * Ensure that only valid captures are saved.

### 5.2 File Export Test (Optional Archive Folder)

* For those implementing the file export, verify that the images are saved correctly in the archive folder as specified in the **save\_blob\_to\_file** procedure.
* Check if the filename is generated based on the **ID** and stored in the correct format (ID\_image\_capture.jpg).
* Ensure the file is accessible from the archive folder.

**Test Cases**:

* Confirm that the image is saved with the correct filename format (ID\_image\_capture.jpg).
* Validate that the images stored in the archive folder match the images stored in the database.

### 5.3 Error Handling and Logs

**Webcam Handling**:

* By default, the system will attempt to use the **primary webcam** connected to the machine.
* If the system has multiple webcams, ensure that the **desired webcam** is set as the primary device. If the wrong webcam is being used, you may need to **disable** the primary webcam or **reconfigure** the default settings to make another device the primary one.

**Test Cases**:

1. **Webcam Not Connected**:
   * Disconnect or disable the webcam and attempt to capture an image. Confirm that the system logs an error and provides an appropriate error message in Oracle Forms.
2. **Incorrect File Permissions**:
   * Test for incorrect or missing permissions on the archive folder. Ensure the system handles it gracefully, displaying an appropriate error message if the image cannot be saved to the shared directory.

### 5.4 Database Commit Testing

* Ensure that after the **database commit**, the images are saved as BLOBs and, exported to the archive folder.

**Test Cases**:

* Confirm that the **WHEN-DATABASE-COMMIT** trigger correctly invokes the save\_blob\_to\_file procedure after committing the record.

# 

# 6. Deployment

This section outlines the steps necessary to deploy the Oracle Forms Webcam Integration system in a production environment. It covers key configurations, network setups, and testing before making the system live.

### 6.1 Network and System Setup

1. **Shared Network Configuration**:
   * Ensure that all resources (Java JDK, OpenCV libraries, temp directories, archive folder) are available on the **shared network**.
   * Verify that each client machine has access to the shared network path where the **Java libraries/Scripts** and **OpenCV** files are stored.
   * Confirm the **shared network paths** are properly referenced in the global variables within the Oracle Form (e.g., :global.java\_exec, :global.java\_lib\_path, :global.temp\_dir).
2. **Oracle Database Setup**:
   * Ensure that the **database** is configured to store images as BLOBs and that the necessary tables (e.g., BC\_IRRADIATION\_DTL) exist.
   * Set up the **Oracle Directory** object for saving images to the shared network archive, if applicable:

| CREATE OR REPLACE DIRECTORY ARCHIVE\_DIR AS '\\sharednetworkpath\ImageArchive'; |
| --- |

Grant appropriate permissions for read and write access:

| GRANT READ, WRITE ON DIRECTORY ARCHIVE\_DIR TO your\_schema; |
| --- |

1. **Host Command Access**:
   * Confirm that **HOST command** access is enabled on all machines using the Oracle Forms application. This is critical for invoking external programs such as the **Java-based webcam capture**.

### 6.2 Form Configuration

1. **Form Customization**:
   * Modify the form's global variables to reference the correct paths based on the production environment setup:

| **:global.java\_exec := '\\sharednetworkpath\Java\jdk1.8.0\_281\bin\javaw.exe'; :global.java\_lib\_path := '\\sharednetworkpath\FormsScripts\JavaCapture'; :global.temp\_dir := '\\sharednetworkpath\Temp';** |
| --- |

**6.3 Testing Before Going Live**

1. **Pre-Deployment Testing**:
   * **Webcam Functionality**: Perform a thorough test of the webcam capture functionality on several machines to ensure the system uses the correct webcam and the image is properly displayed in the form.
   * **Network Path Testing**: Verify that all shared network paths are accessible from the client machines and that Java and OpenCV programs execute without issues.
   * **Database Testing**: Confirm that images are stored in the BLOB columns and, optionally, saved to the shared archive folder.
   * **Error Handling**: Test all error handling scenarios (e.g., webcam disconnected, permission issues) to ensure the system reacts appropriately.