Adrishya - AI Object Remover

- Effortlessly remove unwanted objects from images. Powered by AI for seamless results.

- Note : Improvements have been made to enhance error handling and data validation Based On Evaluation

Overview

Java-based Tool

Robust application designed to remove objects from images efficiently.

AI Powered

Uses advanced AI for flawless, seamless object removal with precision.

Tech Stack

- Java
- FlatLaf UI framework
- IntelliJ IDE
- MySQL database
- LAMA model with ONNX runtime

Core Feature Implementation

Object Removal Engine

Utilizes the LAMA AI model (ONNX) to perform high-quality, lossless object removal

Lasso Selection Tool

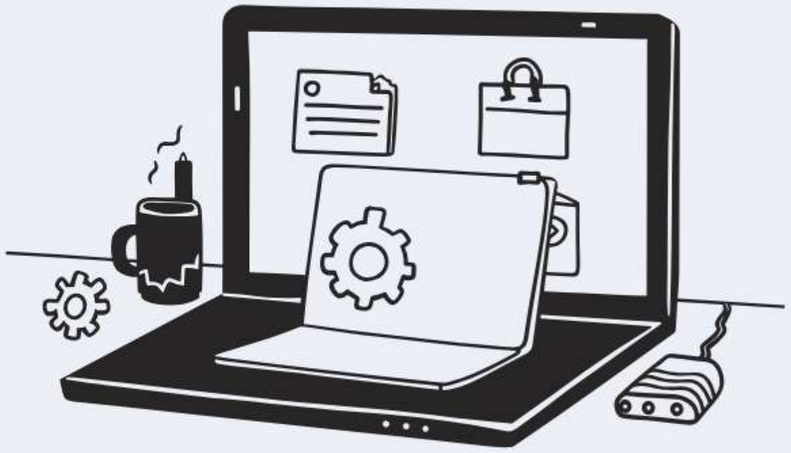
Allows users to intuitively select areas for removal using free-form lasso drawing

Java Backend Integration

Robust backend architecture built using object-oriented principles ensures maintainability and scalability.

Database Connectivity

MySQL database integration handles logging and result storage using DAO and model classes



Python Bridge

`inpaint.py` connects the Java application to the ONNX model, enabling AI-powered image processing.

FlatLaf UI

Clean, modern user interface implemented with FlatLaf for smooth and aesthetic user interaction.

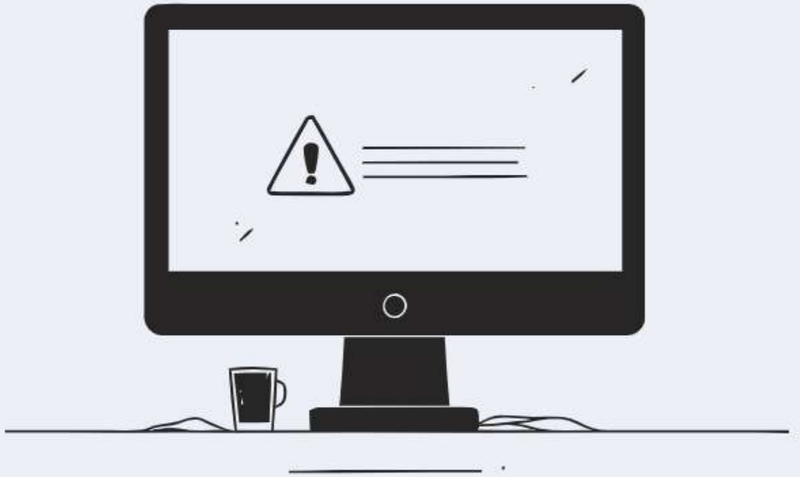
IntelliJ Ready

Entire project is fully runnable within IntelliJ IDEA, making it easy to run, test, and extend.

Error Handling & Robustness:

Try-Catch Blocks:

All critical operations, including file loading, image processing, database connections, and external script execution, are wrapped in try-catch blocks to prevent unexpected crashes.



User Feedback

Meaningful error messages are displayed through dialog boxes when an issue occurs (e.g., file not found, model missing, DB connection failed).

Missing Model Check

The application checks for the presence of `lama_fp32.onnx` and alerts the user if the model is missing or incorrectly placed.

Python Runtime Validation

Ensures that Python and required packages are available before attempting to run `inpaint.py`, with clear prompts for missing dependencies.

Database Connectivity Checks

Gracefully handles connection errors and ensures fallback behavior without crashing the application.

Input Validation

Verifies that the user has selected a valid image and made a selection before attempting AI removal, preventing null or invalid operations.

Integration of Components

- **Calling Python Script from Java**.

```
public class ImageProcessor {

    public static void runInpainting(String imagePath, String maskPath) {
        try {
            ProcessBuilder pb = new ProcessBuilder("python", "src/inpaint.py", imagePath, maskPath);
            pb.redirectErrorStream(true);
            Process process = pb.start();

            // Output for logging
            BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }

            process.waitFor();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Inpainting failed: " + e.getMessage());
        }
    }
}
```

Saving Results to MySQL Database

```
public class InpaintDAO {
    public static void saveResult(String originalImage, String resultImage, String timestamp) {
        try (Connection conn = DBConnection.getConnection()) {
            String query = "INSERT INTO inpaint_results (original_image, result_image, timestamp)
VALUES (?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(query);
            stmt.setString(1, originalImage);
            stmt.setString(2, resultImage);
            stmt.setString(3, timestamp);
            stmt.executeUpdate();
        } catch (SQLException e) {
            System.err.println("Database Error: " + e.getMessage());
        }
    }
}
```

Trigger from GUI (MainApp.java)

```
removeBtn.addActionListener(e -> {
    String imagePath = currentImagePath;
    String maskPath = generatedMaskPath;

    // Run LAMA AI via Python
    ImageProcessor.runInpainting(imagePath, maskPath);

    // Save to DB
    String timestamp = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
    InpaintDAO.saveResult(imagePath, "output/inpainted.png", timestamp);

    JOptionPane.showMessageDialog(null, "Object removed and result saved.");
});
```

This demonstrates the smooth integration between:

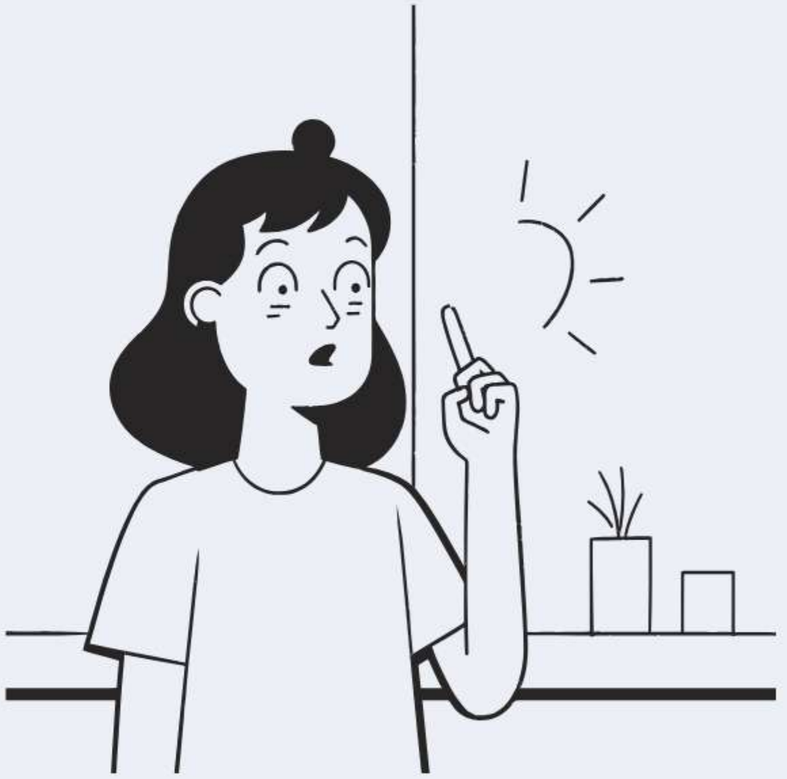
- Java UI interaction (`removeBtn`)
- Backend logic and external Python AI execution
- Database result logging via DAO pattern

Event Handling & Processing

Adrishya AI is designed with efficient and responsive event handling. User actions such as button clicks, lasso selection, and image loading are connected to lightweight, non-blocking event listeners. These listeners delegate tasks like image processing or database saving to appropriate modules using multithreading where needed, ensuring the UI remains responsive.

Key optimizations include:

- Minimal logic inside action listeners; heavy operations are offloaded to separate threads or backend functions.
- Clear separation between UI events and business logic using delegation.
- SwingUtilities and background threads used for GUI updates and long-running tasks (e.g., AI inpainting)



Code

```
removeBtn.addActionListener(e -> {
    // Disable the button during processing
    removeBtn.setEnabled(false);

    new Thread(() -> {
        try {
            // Run the AI inpainting process
            ImageProcessor.runInpainting(currentImagePath, maskPath);

            // Save the result in the database
            String timestamp = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
            InpaintDAO.saveResult(currentImagePath, "output/inpainted.png", timestamp);
```

```
// Update UI on the main thread
            SwingUtilities.invokeLater(() -> {
                JOptionPane.showMessageDialog(null, "Object removed successfully.");
                removeBtn.setEnabled(true);
            });

        } catch (Exception ex) {
            SwingUtilities.invokeLater(() -> {
                JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());
                removeBtn.setEnabled(true);
            });
        }
    }).start();
});
```

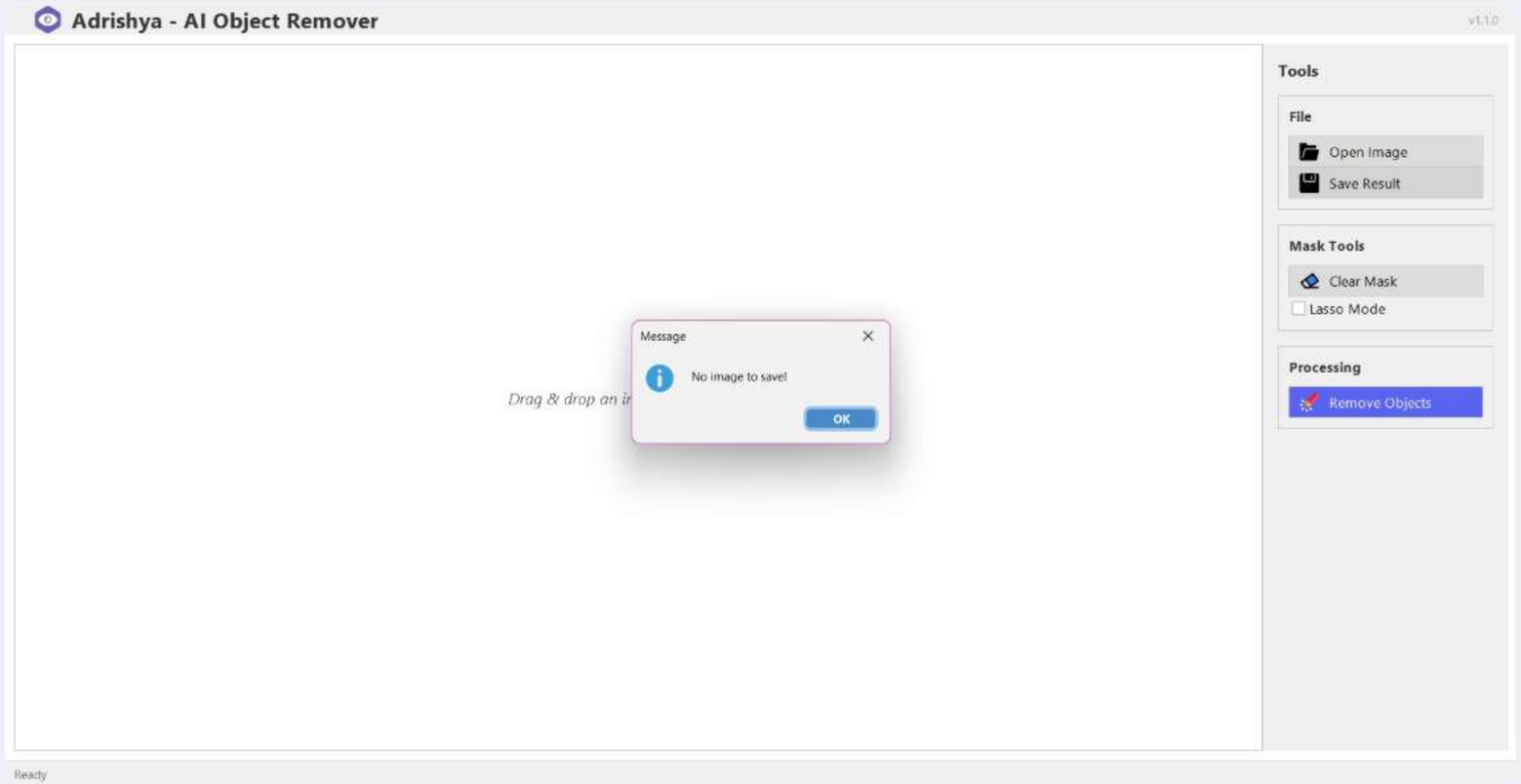
This approach avoids UI freezing by running time-consuming tasks on a background thread, while safely updating the interface using `SwingUtilities.invokeLater`.

Data Validation

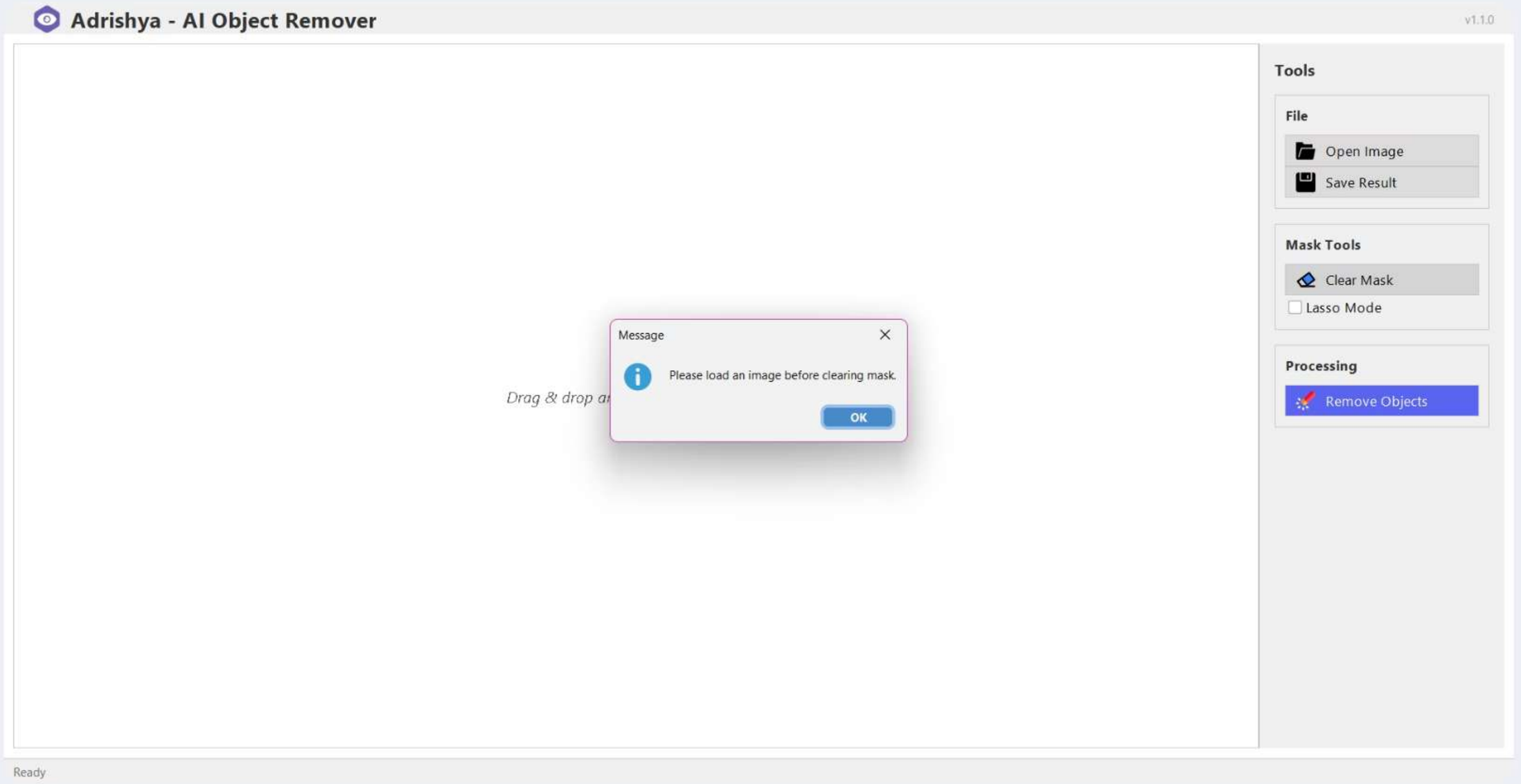
✅ Improvements Added:

- Checks before allowing mask operations (e.g., `clearMask`, `lasso mode`) to ensure an image is loaded.
- Extra validation for `removeBtn` click to ensure both image and mask are present.
- Tooltip hints for buttons and more informative dialogs.
- Defensive coding on possible `null` cases.

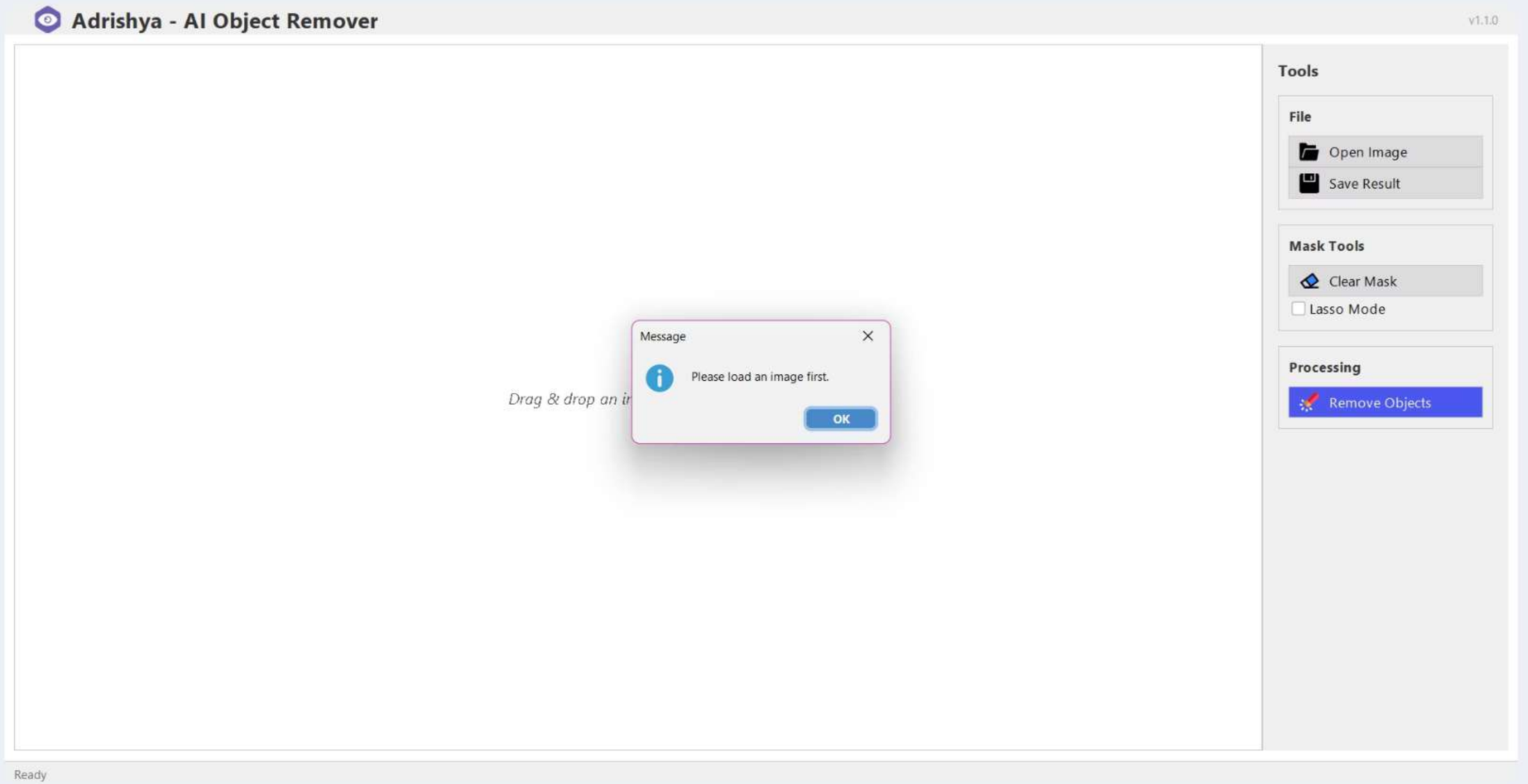
Screenshot - " Saving Image Before Loading Image "



Screenshot - " Clearing Mask Before Loading Image "



Screenshot - " Removing Objecr Before Loading Image"



Adrishya AI implements both **client-side** (within the Java GUI) and **server-side** (backend and database) validation to ensure reliable and secure data processing.

Client-Side Validation

- Ensures an image is selected before processing.
- Verifies that a lasso region is drawn before triggering object removal.
- Confirms that required files (e.g., `lama_fp32.onnx`) exist in the expected directory.

```
if (imagePath == null || imagePath.isEmpty()) {
    JOptionPane.showMessageDialog(null, "Please load an image before proceeding.");
    return;
}

if (!new File("src/lama_fp32.onnx").exists()) {
    JOptionPane.showMessageDialog(null, "LAMA model not found. Please place it in the src directory.");
    return;
}
```

Server-Side Validation (Database)

- Sanitizes input before inserting into MySQL to avoid injection or corruption.
- Validates timestamp and file paths before database operations.

```
public static void saveResult(String originalImage, String resultImage, String timestamp) {
    if (originalImage == null || resultImage == null || timestamp == null) return;

    try (Connection conn = DBConnection.getConnection()) {
        String query = "INSERT INTO inpaint_results (original_image, result_image, timestamp) VALUES (?, ?,
?)";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setString(1, originalImage);
        stmt.setString(2, resultImage);
        stmt.setString(3, timestamp);
        stmt.executeUpdate();
    } catch (SQLException e) {
        System.err.println("Database Error: " + e.getMessage());
    }
}
```

MySQL Table

```
mysql> select * from inpaint_results;
+----+-----------------------------------------------------------------------------------------------+---------------+------------------+---------------------+
| id | input_image_path                                                                              | mask_image    | output_image     | created_at          |
+----+-----------------------------------------------------------------------------------------------+---------------+------------------+---------------------+
|  1 | input_image.png                                                                               | input_mask.png | output_image.png | 2025-05-22 01:12:17 |
|  2 | C:\Users\91783\Documents\Images\Normal_Image_3.jpg                                            | input_mask.png | output_image.png | 2025-05-22 01:25:26 |
|  3 | C:\Users\91783\Documents\Images\Normal_Image_2.jpg                                            | input_mask.png | output_image.png | 2025-05-22 02:47:59 |
|  4 | C:\Users\91783\Documents\1697962028648.png                                                    | input_mask.png | output_image.png | 2025-05-22 02:48:43 |
|  5 | C:\Users\91783\Documents\TestOn.py                                                            | input_mask.png | output_image.png | 2025-05-22 02:49:37 |
|  6 | C:\Users\91783\Documents\McLaren-Sabre-BC-03-3-1024x341 (1).jpg                                | input_mask.png | output_image.png | 2025-05-22 02:49:42 |
|  7 | C:\Users\91783\Documents\Images\Normal_Image_4.jpg                                            | input_mask.png | output_image.png | 2025-05-22 02:59:20 |
|  8 | C:\Users\91783\Documents\Images\Normal_Image_4.jpg                                            | input_mask.png | output_image.png | 2025-05-22 03:07:29 |
|  9 | C:\Users\91783\Documents\Images\Upscaled_Image_3.jpg                                          | input_mask.png | output_image.png | 2025-05-22 03:07:34 |
| 10 | C:\Users\91783\Documents\Images\Normal_Image_3.jpg                                            | input_mask.png | output_image.png | 2025-05-22 03:07:42 |
| 11 | C:\Users\91783\Documents\Images\Upscaled_Image_2.jpg                                          | input_mask.png | output_image.png | 2025-05-22 03:12:41 |
| 12 | C:\Users\91783\Downloads\Test_Images\WhatsApp Image 2025-05-21 at 5.42.48 PM.jpg              | input_mask.png | output_image.png | 2025-05-22 03:13:04 |
| 13 | C:\Users\91783\Documents\1689620933481.png                                                    | input_mask.png | output_image.png | 2025-05-22 03:17:39 |
| 14 | C:\Users\91783\Documents\Images\Normal_Image_3.jpg                                            | input_mask.png | output_image.png | 2025-05-22 20:23:42 |
| 15 | C:\Users\91783\Documents\Images\Normal_Image_4.jpg                                            | input_mask.png | output_image.png | 2025-05-22 20:23:47 |
| 16 | C:\Users\91783\Documents\Images\Normal_Image_3.jpg                                            | input_mask.png | output_image.png | 2025-05-22 22:24:36 |
| 17 | C:\Users\91783\Documents\Images\Normal_Image_3.jpg                                            | input_mask.png | output_image.png | 2025-05-22 22:26:30 |
| 18 | C:\Users\91783\Documents\Images\Normal_Image_3.jpg                                            | input_mask.png | output_image.png | 2025-05-22 22:28:00 |
| 19 | C:\Users\91783\Documents\Images\Normal_Image_3.jpg                                            | input_mask.png | output_image.png | 2025-05-22 23:00:33 |
| 20 | C:\Users\91783\Documents\Images\Normal_Image_2.jpg                                            | input_mask.png | output_image.png | 2025-05-22 23:12:18 |
| 21 | C:\Users\91783\Documents\Images\Normal_Image_4.jpg                                            | input_mask.png | output_image.png | 2025-05-22 23:12:22 |
| 22 | C:\Users\91783\Documents\Images\Upscaled_Image_1.jpg                                          | input_mask.png | output_image.png | 2025-05-22 23:14:08 |
| 23 | C:\Users\91783\Documents\Images\Upscaled_Image_4.jpg                                          | input_mask.png | output_image.png | 2025-05-22 23:14:13 |
| 24 | C:\Users\91783\Documents\Images\Upscaled_Image_4.jpg                                          | input_mask.png | output_image.png | 2025-05-22 23:20:08 |
| 25 | C:\Users\91783\Documents\Images\Upscaled_Image_2.jpg                                          | input_mask.png | output_image.png | 2025-05-22 23:20:12 |
| 26 | C:\Users\91783\Documents\Images\Normal_Image_2.jpg                                            | input_mask.png | output_image.png | 2025-05-22 23:20:21 |
| 27 | C:\Users\91783\Documents\Images\Upscaled_Image_2.jpg                                          | input_mask.png | output_image.png | 2025-05-22 23:21:53 |
```

Code Quality

- **Object-Oriented Design**: Proper use of encapsulation, abstraction, and separation of concerns across `dao`, `model`, and GUI layers.
- **Consistent Structure**: Follows standard Java naming conventions and package organization.
- **Modular Codebase**: Each class and method has a single responsibility, enabling easier testing and debugging.
- **Clear Documentation**: Inline comments and method descriptions are provided for readability and future development.

Innovation

Adrishya AI introduces innovative features that set it apart from standard image editors:

- **AI-Powered Object Removal**: Uses the LAMA inpainting model via ONNX Runtime, seamlessly integrated into a desktop Java app.
- **Lasso-Based Selection**: Interactive lasso tool allows free-form object marking—mimicking Photoshop-like selection in a lightweight setup.
- **Cross-Language AI Execution**: Bridges Java with Python using `ProcessBuilder` to access high-performance AI features.
- **Non-Blocking Execution**: Heavy operations like image inpainting run on background threads, ensuring the UI stays responsive.

Project Documentation

The complete project documentation is provided as an HTML file named `Doc.html`, located inside the `documentation` folder of the GitHub repository.
Simply navigate to the folder and open `Doc.html` in any web browser to explore the full guide, setup instructions, feature usage, code examples, and more.

Structure

```
/Adrishya
│
├── src/
├── documentation/
│   └── Doc.html
└── README.md
```

Screenshot Of Documentation