# Server GUI Development: Java Swing Implementation & Integration Testing - Femi

## 1. Java Swing GUI Implementation

This document provides the implementation details of the Server GUI development using Java Swing, including server status display, client list management, log monitoring, and configuration settings.

### Java Swing Code

The following Java code sets up the initial GUI components as per the requirements. It includes panels for server status, client list, logs, and configuration settings.

import javax.swing.\*;  
import java.awt.\*;  
  
// Main Class for GUI Application  
public class ServerGUI extends JFrame {  
  
 public ServerGUI() {  
 super("Server GUI Development");  
 setLayout(new BorderLayout());  
 setSize(800, 600);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
  
 initComponents();  
 setVisible(true);  
 }  
  
 private void initComponents() {  
 JPanel serverStatusPanel = new JPanel();  
 serverStatusPanel.setBorder(BorderFactory.createTitledBorder("Server Status"));  
 serverStatusPanel.add(new JLabel("Server is running..."));  
   
 JPanel clientListPanel = new JPanel();  
 clientListPanel.setBorder(BorderFactory.createTitledBorder("Client List"));  
 clientListPanel.setLayout(new BoxLayout(clientListPanel, BoxLayout.Y\_AXIS));  
 clientListPanel.add(new JLabel("Client 1"));  
 clientListPanel.add(new JLabel("Client 2"));  
  
 JPanel logPanel = new JPanel();  
 logPanel.setBorder(BorderFactory.createTitledBorder("Logs"));  
 JTextArea logArea = new JTextArea(10, 30);  
 logArea.setText("Log messages will appear here...");  
 logPanel.add(new JScrollPane(logArea));  
  
 JPanel configPanel = new JPanel();  
 configPanel.setBorder(BorderFactory.createTitledBorder("Configuration"));  
 configPanel.setLayout(new GridLayout(2, 2));  
 configPanel.add(new JLabel("Server IP:"));  
 configPanel.add(new JTextField("127.0.0.1"));  
 configPanel.add(new JLabel("Port:"));  
 configPanel.add(new JTextField("8080"));  
  
 add(serverStatusPanel, BorderLayout.NORTH);  
 add(clientListPanel, BorderLayout.WEST);  
 add(logPanel, BorderLayout.CENTER);  
 add(configPanel, BorderLayout.SOUTH);  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.invokeLater(ServerGUI::new);  
 }  
}

## 2. Integration Testing

Integration testing ensures that the components of the GUI interact correctly, including server communication, admin controls, and monitoring systems.

### Testing Tools and Setup

To perform integration testing, we use JUnit and Mockito. JUnit is used for writing and running tests, while Mockito is used to mock server responses. Ensure the following dependencies are added to the project:

<dependencies>  
 <!-- JUnit -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 <!-- Mockito -->  
 <dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-core</artifactId>  
 <version>4.11.0</version>  
 <scope>test</scope>  
 </dependency>  
</dependencies>

### Integration Test Code

The following Java code provides integration tests for the GUI components using JUnit and Mockito.

import org.junit.Before;  
import org.junit.Test;  
import org.mockito.Mockito;  
import javax.swing.\*;  
import static org.junit.Assert.\*;  
import static org.mockito.Mockito.\*;  
  
public class ServerGUITest {  
 private ServerGUI gui;  
 private ServerMock serverMock;  
  
 @Before  
 public void setUp() {  
 gui = new ServerGUI();  
 serverMock = mock(ServerMock.class);  
  
 when(serverMock.getServerStatus()).thenReturn("Server is running...");  
 when(serverMock.getClientList()).thenReturn(new String[] {"Client 1", "Client 2"});  
 when(serverMock.getLogs()).thenReturn("Server started...  
Client 1 connected.");  
 }  
  
 @Test  
 public void testServerStatusPanelIntegration() throws Exception {  
 SwingUtilities.invokeAndWait(() -> {  
 JPanel serverStatusPanel = (JPanel) gui.getContentPane().getComponent(0);  
 JLabel statusLabel = (JLabel) serverStatusPanel.getComponent(0);  
 statusLabel.setText(serverMock.getServerStatus());  
 assertEquals("Server is running...", statusLabel.getText());  
 });  
 }  
  
 @Test  
 public void testClientListPanelIntegration() throws Exception {  
 SwingUtilities.invokeAndWait(() -> {  
 JPanel clientListPanel = (JPanel) gui.getContentPane().getComponent(1);  
 clientListPanel.removeAll();  
 for (String client : serverMock.getClientList()) {  
 clientListPanel.add(new JLabel(client));  
 }  
 assertEquals(2, clientListPanel.getComponentCount());  
 });  
 }  
}

### Documentation Updates

This section covers how to run integration tests and interpret results, focusing on panel behavior and UI updates.