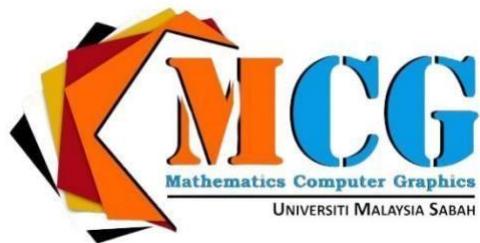




**UMS**  
UNIVERSITI MALAYSIA SABAH



**SEMESTER 1 SESSION 2025/2026**

**SV40303 SCIENTIFIC DATA VISUALIZATION**

**ASSIGNMENT 2**

**"DICOM: Advanced Medical Data Visualizer"**

**LECTURER NAME:**

PROF. DR ABDULLAH BIN BADE

<b>NAME</b>	<b>MATRIC NO.</b>
JEREMY JOHNNY	BS22110183
MOHAMAD ADLI ZILIKHRAM BIN SAHARUDIN	BS22110469

## **TABLE OF CONTENTS**

<b>CONTENTS</b>	<b>PAGE</b>
SYSTEM INTRODUCTION	3
SYSTEM ARCHITECTURE	4-5
SYSTEM FLOWCHART	6
ALGORITHMS	7-8
OUTPUTS	9

## **SYSTEM INTRODUCTION**

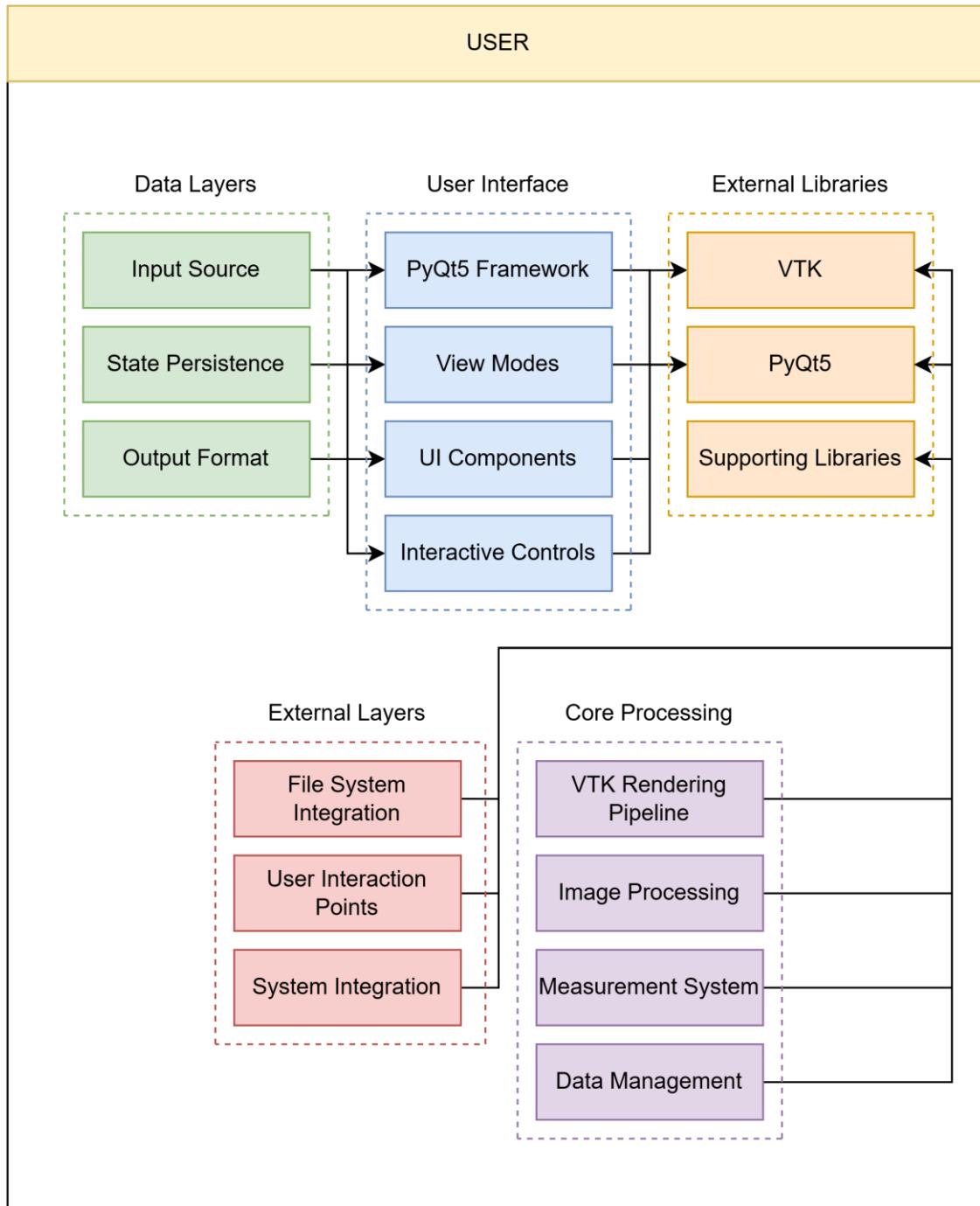
The DICOM Medical Visualizer is a powerful desktop application designed to transform standard medical imaging data into interactive 3D visualizations. Built for medical professionals, researchers, and students, this system takes DICOM scan files, commonly used in CT, MRI, and other medical imaging, and converts them into dynamic, manipulatable 3D models. Users can explore anatomical structures in real time, switch between 3D volume views and detailed 2D slice displays and perform precise measurements directly within the rendered anatomy.

At its core, the system integrates the advanced rendering capabilities of VTK with an intuitive PyQt5 interface, creating a seamless experience where complex medical data becomes accessible and interactive. Features like adjustable contrast, tissue-specific viewing presets, and multi-angle rotation allow for detailed examination of internal structures, while measurement tools provide quantitative insights that support diagnostic and educational purposes. The application supports multiple datasets simultaneously, preserves user sessions, and exports results, making it a practical tool for both clinical and research environments.

Designed with usability in mind, the DICOM Medical Visualizer bridges the gap between specialized medical software and everyday usability, offering professional-grade visualization without the steep learning curve or high cost of traditional medical imaging platforms. Its dual-view interface, responsive controls, and modular architecture make it an adaptable solution for visualizing human anatomy, planning procedures, or teaching complex medical concepts in an engaging, visual format.

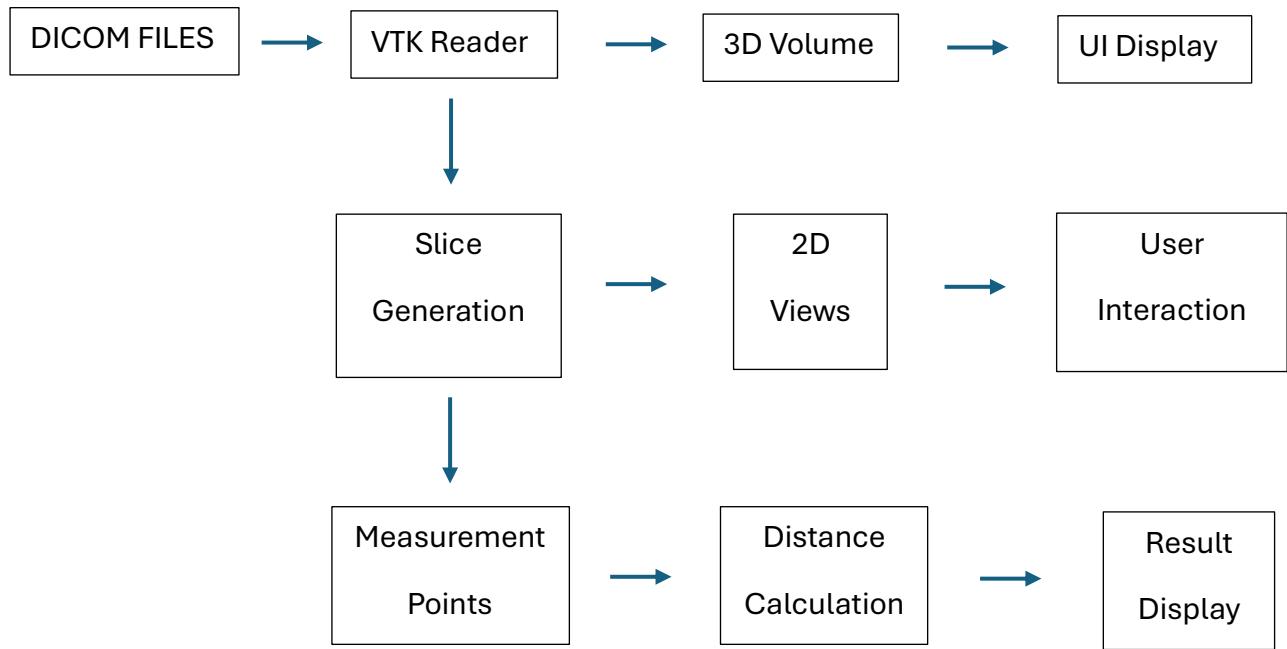


## SYSTEM ARCHITECTURE



The DICOM Medical Visualizer follows a layered architecture where the Core Processing Engine (VTK-based rendering and image processing) serves as the central nervous system, processing medical data from the Data Management Layer (DICOM files and saved states) and driving visual output. The User Interface Layer (PyQt5 framework) wraps this core, providing interactive controls and dual-view displays to users while communicating their inputs back to the processing engine. External Libraries (VTK, PyQt5, NumPy) form the foundation that powers both the interface and processing capabilities. Finally, the External/Interface Layer handles file system operations and user interactions, creating a complete feedback loop where user actions through the UI trigger processing in the core, which in turn updates the display while managing data persistence and system resources.

## SYSTEM FLOWCHART



## ALGORITHMS

### Main System

```
ALGORITHM DICOM_Medical_Visualizer_Main_System BEGIN // Initialization Phase CALL
APPLICATION_STARTUP()

// Main Event Loop
WHILE application_is_running DO
    CALL PROCESS_USER_INTERACTION()

    // View Mode Management
    IF current_view_mode == MAIN_VIEW THEN
        CALL RENDER_MAIN_VIEW()
    ELSE IF current_view_mode == SLICE_VIEW THEN
        CALL RENDER_SLICE_VIEW()
    ENDIF

    // Measurement Mode Check
    IF measurement_mode_active == TRUE THEN
        CALL MONITOR_MEASUREMENT_INPUT()
    ENDIF

    // Handle UI Updates
    CALL UPDATE_UI_COMPONENTS()
    CALL RENDER_ALL_VIEWS()
ENDWHILE

// Cleanup Phase
CALL SYSTEM_CLEANUP()

END ALGORITHM

PROCEDURE APPLICATION_STARTUP() BEGIN // Initialize UI Framework CREATE PyQt5_Application()
CREATE MainWindow_Instance()

// Apply Visual Theme
SET_THEME_TO_DARK()

// Initialize VTK Engine
CREATE myVTK_Instance()
CONNECT_VTK_TO_UI()

// Setup Dual View System
CREATE_STACKED_WIDGETS()
CREATE_CONTROL_PANEL()
CREATE_MODEL_LIST_PANEL()

// Show Application
DISPLAY_FULL_SCREEN()
SET_VIEW_MODE(MAIN_VIEW)

END PROCEDURE

PROCEDURE PROCESS_USER_INTERACTION() BEGIN // Process Menu Actions IF menu_action ==
LOAD_DICOM THEN CALL LOAD_DICOM_MODEL() ELSE IF menu_action == SAVE_STATE THEN CALL
```

```
SAVE_MODEL_STATE() ELSE IF menu_action == EXPORT_IMAGE THEN CALL EXPORT_VIEW_AS_IMAGE()
ENDIF

// Process Button Actions
IF button_clicked == MEASUREMENT_TOGGLE THEN
    CALL TOGGLE_MEASUREMENT_MODE()
ELSE IF button_clicked == VIEW_SWITCH THEN
    CALL SWITCH_VIEW_MODE()
ENDIF

// Process Slider Changes
FOR EACH slider IN [axial, coronal, sagittal, window, level, opacity] DO
    IF slider_value_changed THEN
        CALL UPDATE_VISUALIZATION_PARAMETER(slider)
    ENDIF
ENDFOR

END PROCEDURE
```

## SEVERAL SIGNIFICANT OUTPUT

