**COMSATS University Islamabad,   
Abbottabad Campus**

**SOFTWARE DESIGN DESCRIPTION   
(SDD DOCUMENT)**

**for**

**Liver Tumor Segmentation in CT Scan images Using Light Weight Deep Learning Model**  
Version 1.0

***By***

**Faizan CIIT/FA21-BSE-011/ATD**

**Fawad Iqbal CIIT/FA21-BSE-012/ATD**

***Supervisor*Dr. Mubashir Ahmad**

***Bachelor of Science in Computer Science (2024-2025)***

**Table of Contents**

**Revision History 3**

**1.** **Introduction 5**

**2.** **Design Methodology and software process model 5**

**3.** **System Overview 5**

3.1 Architectural Design 5

3.2 Process Flow/Representation 5

**4.** **Design Models [along with descriptions] 5**

**5.** **Data Design 6**

5.1 Data Dictionary 6

**6.** **Algorithm & Implementation 6**

**7.** **Software Requirements Traceability Matrix 6**

**8.** **Human Interface Design 7**

8.1 Screen Images 7

8.2 Screen Objects and Actions 7

**9.** **Appendix I 7**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

**Supervised by**

**<Supervisor’s Name>**

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

The liver tumor segmentation project automates tumor segmentation in 3D CT scans using nnFormer architecture. Due to computational limitations, the project processes 2D images extracted as patches from 3D CT scans. nnFormer integrates convolution layers and attention mechanisms, effectively combining local and global contexts to enhance segmentation accuracy and efficiency.

The scope of the project includes the following modules:

1. **Data Preprocessing**: Conversion of segmentation.nii and volume.nii files into 2D images of 512x512 resolution, excluding patches without relevant information.
2. **Segmentation Models**: Two distinct nnFormer models adapted for 2D inputs—one for liver extraction and another for tumor segmentation.
3. **Frontend Interface**: A professional yet simple UI for doctors, built with React and Material UI, enabling image upload, segmented result display, and analytical visualization.
4. **Analytical Insights**: Statistical analysis of tumor characteristics to aid in medical decision-making.

This project uses the LITS17 dataset and emphasizes improving segmentation outcomes compared to architectures like CNN-based UNet, TransUNet, and SwinUNet.

# Design methodology and software process model

Design Methodology  
Our project follows an Object-Oriented Programming (OOP) approach. The use of OOP is justified by the modular and reusable nature of its design principles, which align with the complexity of the system. Key components, such as data preprocessing, model training, and frontend development, are encapsulated into distinct classes and objects. This methodology supports better abstraction, enhances code readability, and simplifies the integration of different modules. The OOP approach also allows us to efficiently manage the interactions between the nnFormer-based segmentation models and the frontend interface.

Software Process Model

We are adopting the **Incremental Development Model** for our project. This choice is justified by the modular nature of our system and the iterative improvements required to handle resource constraints effectively. Each module—data preprocessing, model training, and UI development—is treated as a separate increment, allowing us to deliver functional components progressively.

1. **Flexibility**: Allows iterative refinement of nnFormer adaptations, especially for 2D input handling and performance optimization.
2. **Risk Management**: Early delivery of functional modules, such as preprocessing or frontend interfaces, helps identify and address risks early.
3. **Resource Optimization**: Dividing the project into increments helps manage limited computational resources effectively during model training and testing.
4. **Feedback Integration**: Each increment provides an opportunity to incorporate feedback from stakeholders or test outcomes.

# System overview

The liver tumor segmentation system is designed to automate the segmentation of liver and tumor regions in 3D CT scans. The core objective of the system is to enhance segmentation accuracy and efficiency by leveraging the nnFormer architecture, which combines convolutional layers with self-attention mechanisms. This hybrid approach captures both local and global context, improving the overall segmentation performance. Due to computational constraints, the system processes 2D image patches extracted from 3D CT scans rather than utilizing full 3D volumes.

## Functionality

The system operates through a two-phase process: preprocessing and segmentation. The preprocessing phase involves converting the 3D CT scans into 2D images (512x512). These patches are then input into the nnFormer model for segmentation, generating masks that highlight the liver and tumor regions. These segmented outputs are visualized and analyzed through a user-friendly frontend.

1. **Image Upload and Preprocessing**: Users can upload 3D CT scans, which are pre-processed into 2D patches to serve as input for the model.
2. **Segmentation**: The pre-processed 2D patches are fed into the nnFormer architecture, which segments the liver and tumor regions.
3. **Visualization**: The segmented output, including tumor statistics such as volume and area, is displayed to the user for clinical analysis.
4. **Analytics**: Tumor-related metrics such as size, volume, and location are extracted and presented to assist in the diagnosis process.

## Context

This system is intended to improve the efficiency and accuracy of medical image segmentation by automating the process of tumor identification in liver CT scans. By leveraging advanced techniques such as convolution layers and attention mechanisms, the system is able to provide accurate results even with the challenges of computational limitations. The system will be tested using the LITS17 dataset, a well-known benchmark in liver tumor segmentation research.

## Design

The system follows a modular design approach, ensuring separation of concerns between model training, data preprocessing, and frontend development. This modularity makes the system maintainable and scalable. The nnFormer architecture incorporates both convolutional layers and self-attention mechanisms to handle both local and global contexts effectively. The frontend is built using **React** and **Material UI**, designed to provide an intuitive, professional interface suitable for medical applications, following Nielsen’s usability principles.

## Background Information

* **nnFormer**: nnFormer is an advanced deep learning architecture specifically designed for medical image segmentation. It integrates convolutional layers with attention mechanisms, enabling the model to capture both local features and global context from the input images.
* **LITS17 Dataset**: The LITS17 dataset consists of 3D CT images annotated with liver and tumor masks, widely used for benchmarking medical segmentation models. It provides a standard for evaluating the performance of liver and tumor segmentation systems.

**Architectural design**

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high-level overview of how the system’s modules collaborate with each other in order to achieve the desired functionality.

Don’t go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together.

Provide a diagram showing the major subsystems and their connections. **Use a simple Line-Box-Diagram for simpler systems and detailed diagrams (MVC, Client-Server, Layered, Multi-tiered) for complex systems.**

**Process flow/Representation**

Provide a representation of the flow of **MAJOR processes** of your system in the form of an activity diagram. **DO NOT CREATE ACTIVITY DIAGRAMS FOR LOGIN OR SIGN-UP UNLESS THEY INVOLVE SIGNIFICANT COMPLEXITY**. Include only the major processes.

**Design models [along with descriptions]**

**The applicable models may include:**

* Class Diagram
* Sequence Diagram
* State Transition Diagram
* Data Flow Diagram
* Schematic diagram (Hardware projects only)
* Timing diagram (Hardware projects only)

Insert ***applicable*** system models here.

You should be clear about all the concepts used in your diagrams for example for class diagram you should know about aggregation, composition, and inheritance/generalization. Also ensure visibility of all diagrams.

Class diagram and associated models shall only be necessary for object-oriented approach. In case of procedural, create a DFD. Data flow diagram should be extended to 2-3 levels. It should clearly list all processes, their sources/sinks and data stores.

**Note: System design should be complete in all aspects. Create any/all diagrams if you need to. A DFD can also be supplemented by a State Transition Diagram depending on the nature of the project.**

**Hardware projects can include Schematic diagram, System block diagram, timing diagram, Flow charts as replacement of sequence diagram/ Data flow diagram AFTER CONSULTATION WITH THEIR SUPERVISORS. Choice of models must be properly justified.**

**Data design**

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized.

List any databases or data storage items.

**Data dictionary**

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

**Algorithm & Implementation**

In this section, we take a closer look at what each component does in a more systematic way. Provide a summary of your algorithm for each function listed in procedural description language (PDL) or pseudo code.

If you gave an OO description, summarize each object member function for all the objects listed in PDL or pseudo code. Describe any local data when necessary.

**Software requirements traceability matrix**

This section should contain a table that summarizes how each software requirement has been met in this document. The tabular format permits one-to-one and one-to-many relationships to be shown.

**Table 1 Requirements Traceability Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. Number** | **Ref. Item** | **Design Component** | **Component Items** |
| FR01 | Class Diagram | ClassName | FunctionName(s) |
| OR | | | |
| FR01 | DFD | DiagramNumber/Level | FunctionName(s) |

**Human interface design**

Describe the functionality of the system from the user’s perspective. Explain how the user will be able  to use  your system to complete  all the  expected  features and  the  feedback  information that will be displayed for the user.

**Screen images**

Display screenshots showing the interface from the user’s perspective. These can be hand-drawn, or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)

**8.2 Screen objects and actions**

A discussion of screen objects and actions associated with those objects

**Appendix I**

* How to design using UML (OOP): For guidance please follow the instructions mentioned in the link: http://agilemodeling.com/artifacts/
* How and when to design ER diagrams: For guidance please follow the instructions mentioned in the link:

<http://people.inf.elte.hu/nikovits/DB2/Ullman_The_Complete_Book.pdf>

* Data flow diagrams: For guidance please follow the instructions mentioned in the link and book:
  + http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm
  + Software Engineering –A Practitioner’s approach by Roger Pressman
* Architecture diagram: For guidance please follow the instructions mentioned in the link and book:
  + Ian Sommerville – Software Engineering 9th Edition– Chapter 6