System Design Document

Enabling 3D Printing of a Medical CT-Scan:

A Web App for Patients and Practitioners

Prepared by The Slice Is Right

COS 397 - Computer Science Capstone 1

November 10th, 2025

Version 1.0.0

Table of Contents

1 Introduction

This capstone project, completed as part of the requirements for the Bachelor of Science in Computer Science at the University of Maine, focuses on developing a web application called *"Enabling 3D Printing of a Medical CT-Scan: A Web App for Patients and Practitioners."* The goal of this project is to create a browser-based tool that gives users the ability to upload medical CT scans in the DICOM format, produce a viewable 3D model, and turn them into files that can be used for 3D printing. The application is designed to produce both polygonal (.stl) files, which can be used to visualize different anatomical structures like bone, skin, or muscle, and G-code files, which are needed for 3D printers to accurately reproduce tissue characteristics. One of the most important aspects of this project is that all processing is done locally on the user's computer, which helps protect sensitive medical data and makes the tool more accessible for both medical research and patient education. By working on this project, we are contributing to the University of Maine's ongoing efforts to advance medical imaging and 3D printing, especially through the Laboratory for Convergent Science.

1.1 Purpose of This Document

The main goal of this System Design Document (SDD) is to explain, in technical terms, how the system described in the Software Requirements Specification (SRS) will actually be built. In other words, this document takes the requirements and turns them into a clear plan for development, covering things like the system's architecture, how data will be organized, and how the different parts of the project will be broken down. This SDD is written for Dr. Terry Yoo, the University of Maine Computer Science faculty, and everyone on *The Slice Is Right* development team. It is meant to be a guide for how the web application, "Enabling 3D Printing of a Medical CT-Scan: A Web App for Patients and Practitioners," will be put together, including the main structure, how the different components will work together, how data will move through the system, and how files will be organized. By laying all of this out, the SDD helps make sure that everyone involved has the same understanding of how the design meets the requirements in the SRS, and that there is enough technical detail for building, testing, and maintaining the system in the future.

1.2 References

Primary Project Source

Yoo, Terry. *Enabling 3D Printing of a Medical CT-Scan: A Web App for Patients and Practitioners*. University of Maine, Laboratory for Convergent Science, 2025.

Web and Technical References

- "DICOM Standard." National Electrical Manufacturers Association (NEMA). Available at: https://www.dicomstandard.org/
- "Polygonizing a Scalar Field(Marching Cubes)." Paul Bourke. Available at: https://paulbourke.net/geometry/polygonise/
- "STL (File Format) Specification." 3D Systems. Available at: https://www.3dsystems.com/support
- "G-code Reference." RepRap Wiki. Available at: https://reprap.org/wiki/G-code websites).
- "JavaScript library for scientific visualization"(vtk.js). Kitware. Available at: https://kitware.github.io/vtk-js
- "Vtk-s Github Repository." Kitware. Available at: https://github.com/Kitware/vtk-js/tree/master
- "Selenium Automates Browsers." Selenium. Available at: https://www.selenium.dev/

2 System Architecture

This section defines how the system is organized and why. In 2.1, Architectural Design, we present the overall runtime approach and key quality attributes, along with the logical and technology views. In 2.2, Decomposition Description, we break the system into concrete parts with clear responsibilities and interfaces.

2.1 Architectural Design

The system uses a client-only, browser-resident architecture to keep CT data private and simplify deployment.

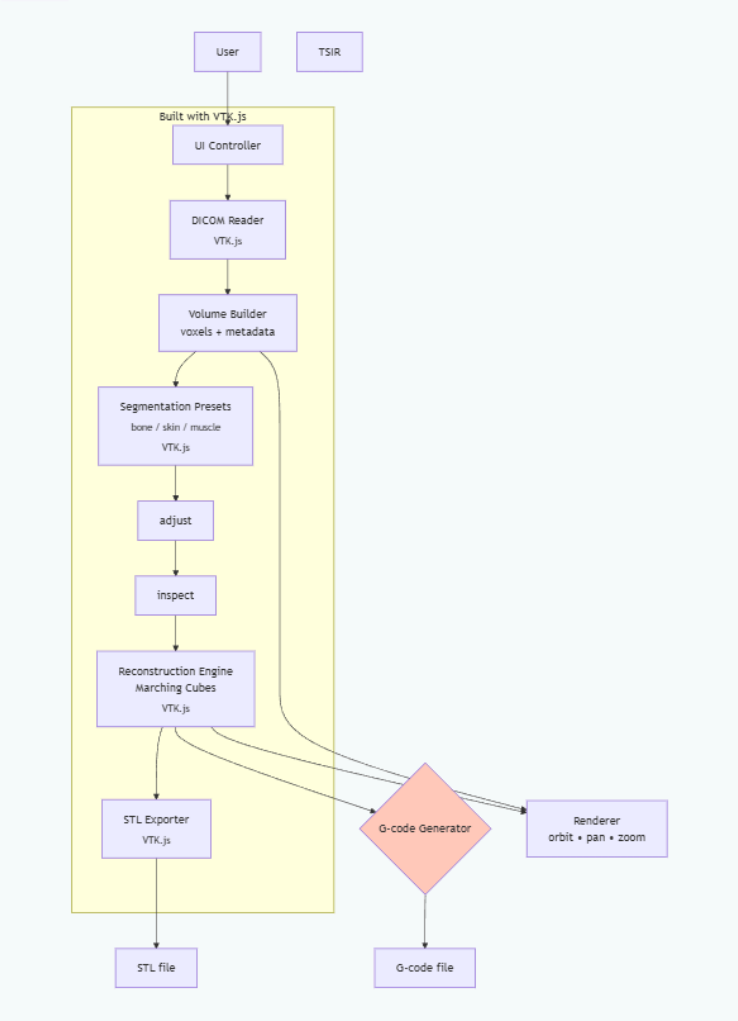

Figure 2.1 - Logical architecture diagram

As shown in the diagram, the logical flow of The Slice Is Right runs entirely on the user's device, with its core components built upon the open-source VTK.js library. The process begins with the VTK.js DICOM Reader, which ingests and validates data to produce a voxel volume.

VTK.js Segmentation Presets then apply Hounsfield Unit (HU) thresholds to this volume to create a tissue mask. The VTK.js Reconstruction Engine, utilizing the Marching Cubes algorithm, extracts a triangle mesh for visual preview within the Renderer. From this point, the user can choose to export the model. The VTK.js STL Exporter saves the mesh as an STL file directly. Alternatively, the generated mesh is passed to an external G-code Generator, a component over which The Slice Is Right has no control. The UI mediates each step, allowing users to adjust thresholds, re-run reconstructions, and visually confirm results before saving. No runtime network traffic occurs, and the only artifacts written are the output files the user explicitly chooses to save.

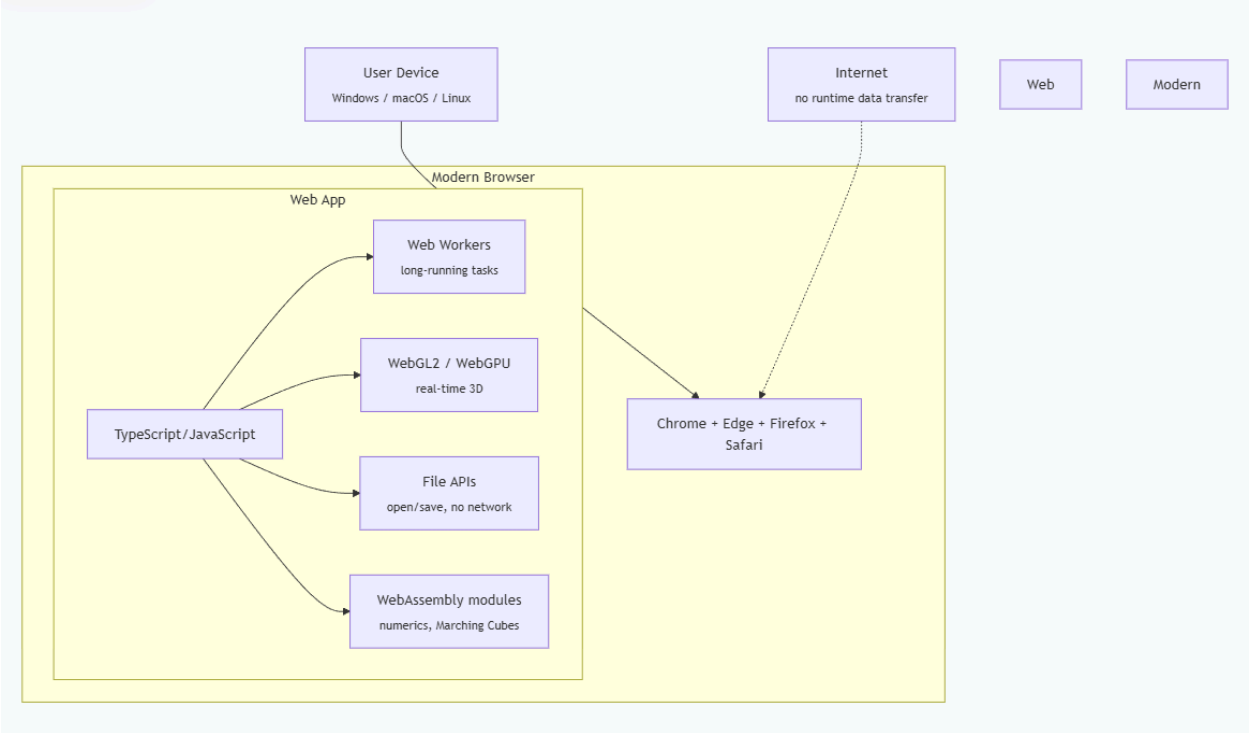

Figure 2.2 - Technology architecture diagram

Figure 2.2 Illustrates the technology stack inside a modern browser. Application logic is written in TypeScript, while long-running computations are executed in Web Workers to maintain a responsive interface. Vtk.js handles performance-critical tasks such as surface extraction via the marching cubes algorithm and 3D rendering, leveraging WebGL2 for GPU acceleration and optionally using WebGPU when available. Browser file APIs manage local file open and save to enable offline operation. This design targets the latest versions of Chrome, Edge, Firefox, and Safari on Windows, macOS, and Linux, balancing privacy and portability while providing sufficient performance for typical study data sizes.

Pending decisions: default WebGL2 vs. opportunistic WebGPU; whether G-code profiles are fixed templates or user-saved.

2.2 Decomposition Description

This section breaks down the system into its main parts. The entire application runs in a web browser on the user's own computer. This design keeps data private and allows for offline use. The main parts work together to manage the steps from loading a medical scan to exporting a 3D model. These include the UI Controller that manages the different steps, the processing units that handle data and complex calculations, and the visual display that shows the 3D model. The following sections explain what each of these parts does and how they interact.

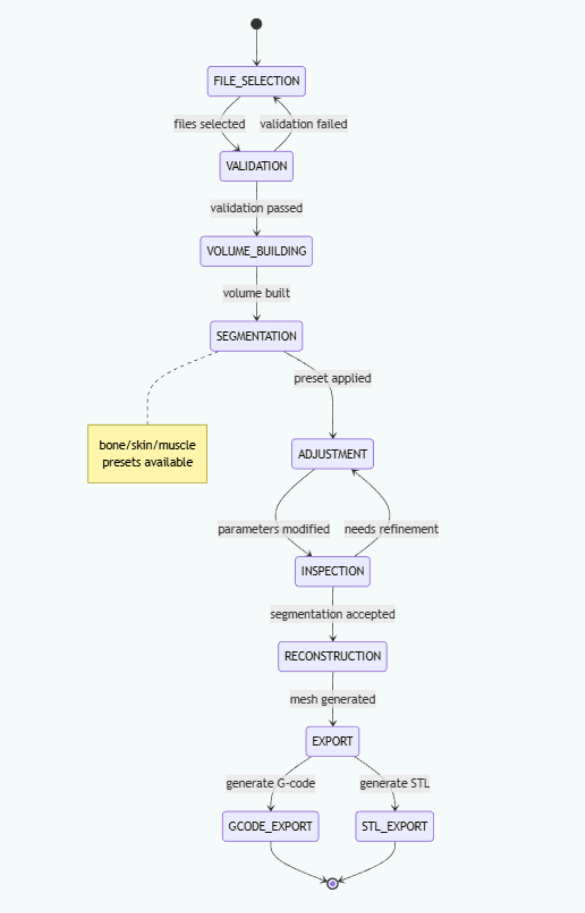

Figure 2.3

The UI Controller manages application flow through distinct states that correspond to different processing stages. It begins in FILE_SELECTION where users provide input data. Upon file selection, it transitions to VALIDATION, ensuring DICOM compatibility before proceeding to VOLUME_BUILDING where 3D voxel data is constructed. The controller then enters the SEGMENTATION phase, offering preset tissue types (bone/skin/muscle) for initial processing. This leads to an adjustable loop between ADJUSTMENT and INSPECTION states, allowing

users to refine parameters and preview results iteratively until satisfied. Once segmentation is approved, the RECONSTRUCTION state generates the 3D mesh using the Marching Cubes algorithm. Finally, the controller transitions to EXPORT options where users can choose between GCODE_EXPORT for 3D printing or STL_EXPORT for model sharing. Each state maintains clear boundaries with defined entry/exit conditions, ensuring smooth workflow progression while allowing backward transitions for refinement when needed.

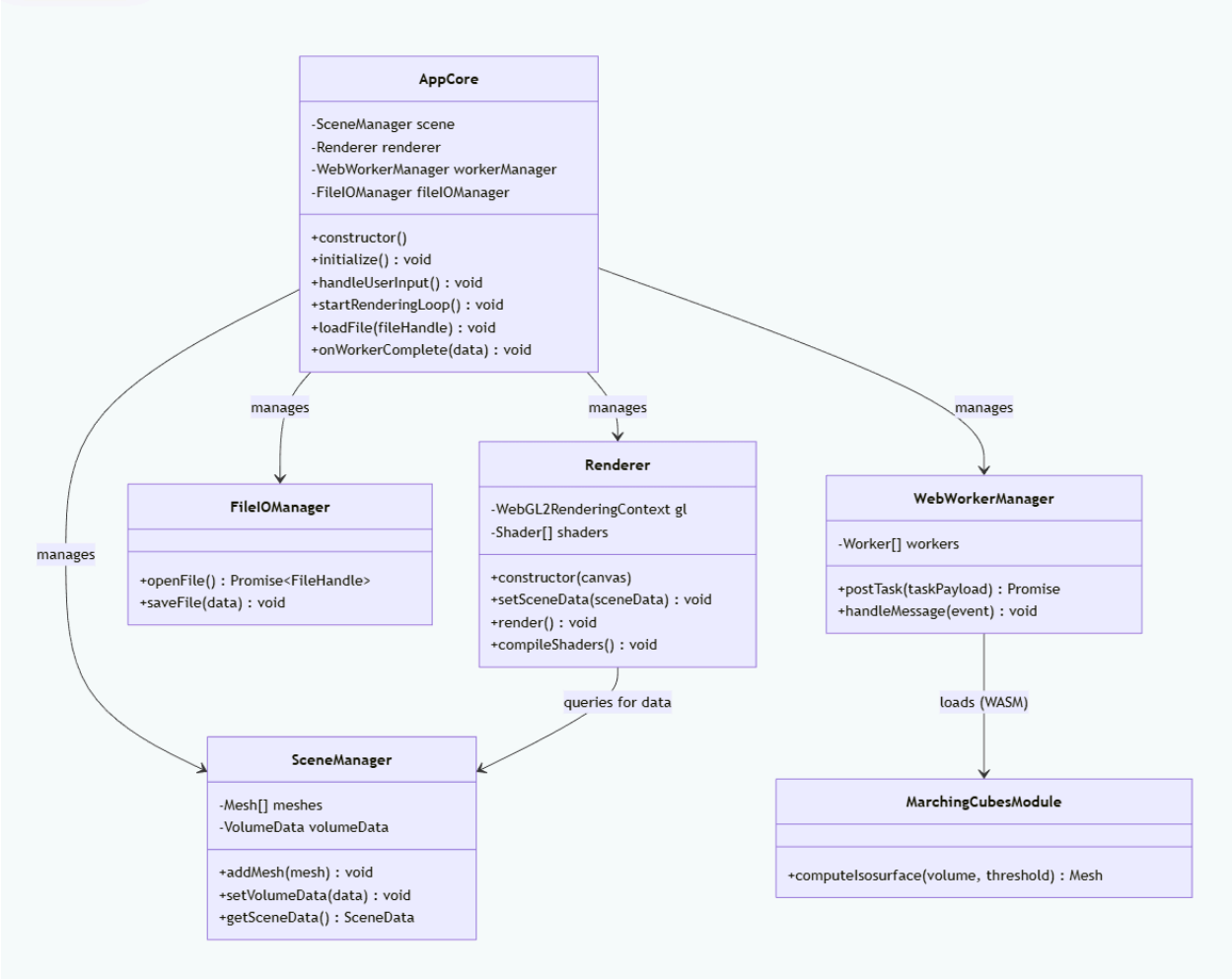

Figure 2.4

Based on the technology architecture described in the Architectural Design, the class diagram elaborates on the implementation within the Main App and Renderer. The AppCore class acts as the central Controller, initializing the system and managing the flow of data. It depends on the SceneManager (the Model) to maintain the state of the 3D scene and volume data. The Renderer (the View) is responsible for taking the scene graph from the SceneManager and drawing it to the screen. The WebWorkerManager abstracts the complexity of dealing with concurrent threads and is responsible for loading and executing the WebAssembly module for the performance-critical MarchingCubesModule. The FileIOManager encapsulates all interactions with the browser's file

system. This hybrid structure effectively balances performance, organization, and the constraints of the modern web environment.

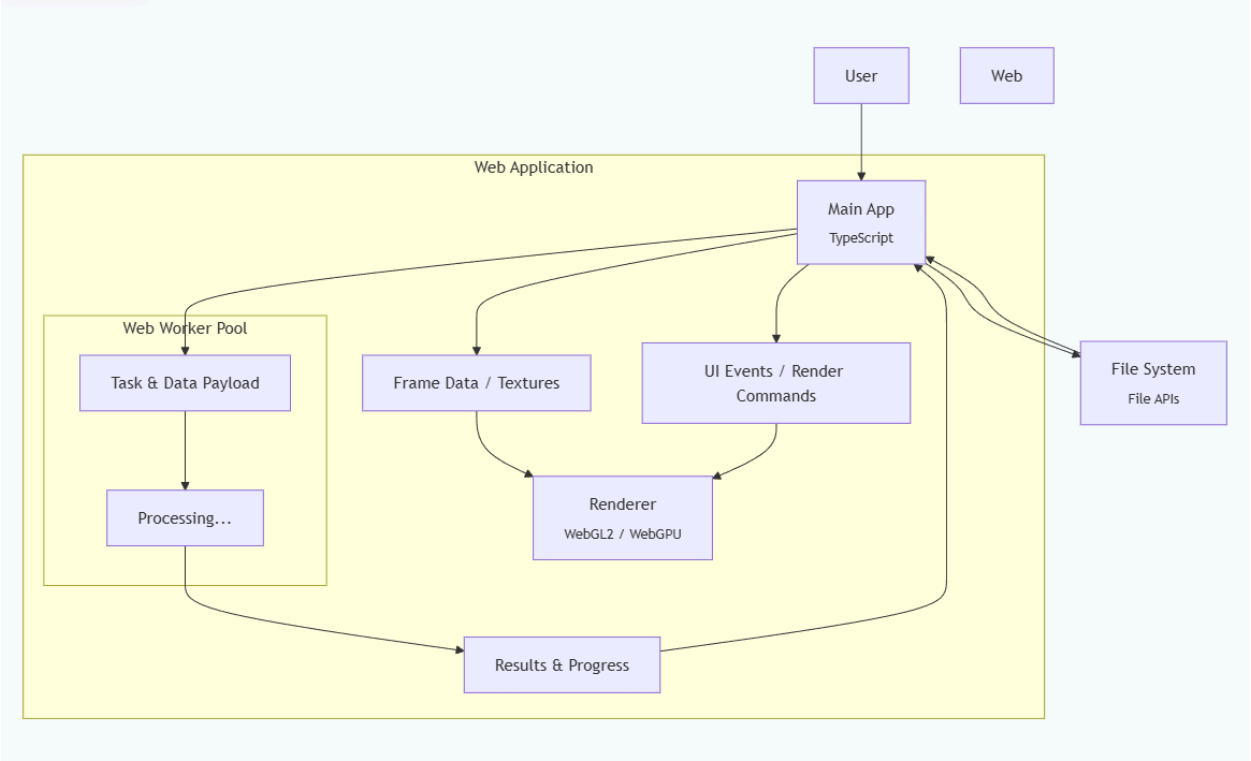

Figure 2.5

Based on the technology architecture described in the Architectural Design, the structural diagram communicates the system's runtime architecture, highlighting concurrent execution and data flow. The Main App handles the user interface and central coordination. It dispatches computationally intensive tasks, like running the Marching Cubes algorithm, to the Web Worker Pool to maintain UI responsiveness. The worker sends results back to the main thread, which forwards scene data to the Renderer. The Renderer, utilizing WebGL2 or WebGPU, processes this data to generate real-time 3D visuals. All data persistence is handled locally via the File System APIs, completing the offline operation capability.

3 Persistent Data Design

3.1 Client-Side History Storage

To support session persistence and user history without requiring server-side storage, the application leverages browser cookies. Cookies are used to store metadata such as recently loaded DICOM files, user-selected anatomical regions and settings, or application preferences. This allows the system to restore the user's state between sessions while maintaining privacy, since no data is sent over the network.

Key points:

- **Scope**: Cookies are limited to storing small amounts of non-sensitive information (e.g., file names, timestamps, user preferences).
- **Implementation**: The application reads and writes cookies via standard browser APIs when files are loaded, saved, or modified.
- **Retention**: Cookies are configured with an appropriate expiration period to maintain history while avoiding indefinite storage.

3.2 File Descriptions

The system processes and generates three primary file types: **DICOM**, **STL**, and **G-code**. Each file type is handled entirely on the client side using browser APIs, with no network or database storage. The following subsections describe each file's structure, purpose, and data fields relevant to implementation.

DICOM Input Files

File Type: .dcm
Source: User-supplied medical CT scan (DICOM standard compliant)
Purpose: Represents the raw input data from which anatomical reconstructions are generated.

Each DICOM file contains one image “slice” with embedded metadata that defines pixel intensity values (Hounsfield Units), orientation, spacing, and acquisition information. When multiple slices are provided, the system reconstructs them into a volumetric dataset (3D voxel array).

Structure Overview

Field Name	Data Type	Description
PatientID	string	Unique identifier for the study (removed before processing to ensure privacy).
SeriesInstanceUID	string	Unique series identifier used to group slices into a 3D volume.
PixelData	uint16[][]	2D array of voxel intensity values representing one CT slice.
Rows, Columns	int	Dimensions of the pixel grid for each slice.
SliceThickness	float	Distance (mm) between consecutive slices.
PixelSpacing	float[2]	Physical spacing (mm) between adjacent pixels within a slice.
RescaleIntercept, RescaleSlope	float	Parameters for converting raw pixel data to Hounsfield Units.
ImageOrientationPatient, ImagePositionPatient	float[6], float[3]	Defines spatial orientation and position for reconstructing the 3D volume.

STL Output Files

File Type: .stl (ASCII or Binary)

Purpose: Encodes the 3D polygonal surface mesh derived from segmented CT data.

The STL file describes a 3D geometry using a collection of triangular facets, each defined by three vertex coordinates and a surface normal vector. Binary STL is used by default for compactness.

Structure Overview (Binary STL)

Field Name	Data Type	Description
Header	char[80]	Optional text header describing export parameters.
NumTriangles	uint32	Number of triangular facets in the mesh.
NormalVector	float[3]	Unit normal vector for each triangle.
Vertex1, Vertex2, Vertex3	float[3] each	Coordinates of the triangle's vertices in model space.
AttributeByteCount	uint16	Optional per-triangle attribute (set to zero).
PixelSpacing	float[2]	Physical spacing (mm) between adjacent pixels within a slice.
RescaleIntercept, RescaleSlope	float	Parameters for converting raw pixel data to Hounsfield Units.
ImageOrientationPatient, ImagePositionPatient	float[6], float[3]	Defines spatial orientation and position for reconstructing the 3D volume.

G-code Output Files

File Type: .gcode

Purpose: Encodes printer instructions for fabricating physical models that replicate the geometry and X-ray attenuation properties of tissue.

Each G-code file contains a sequence of printer movement and extrusion commands derived from the segmented mesh or voxel data. These files can be opened directly in common slicers or sent to standard 3D printers.

Structure Overview

Field Name	Data Type	Description
Header	string	Printer configuration and initialization commands.
LayerCommands	list	Sequence of movement and extrusion commands for each print layer.
Footer	string	Shutdown commands to finalize the print.
Comments	string	Optional notes describing density mapping or export parameters.
AttributeByteCount	uint16	Optional per-triangle attribute (set to zero).
PixelSpacing	float[2]	Physical spacing (mm) between adjacent pixels within a slice.
RescaleIntercept, RescaleSlope	float	Parameters for converting raw pixel data to Hounsfield Units.
ImageOrientationPatient, ImagePositionPatient	float[6], float[3]	Defines spatial orientation and position for reconstructing the 3D volume.

4 Requirements Matrix

The requirements matrix provides a structured overview of the system's functional and non-functional requirements, linking each requirement to the modules and components responsible for its implementation. This matrix serves as a reference to ensure that all specified behaviors and constraints are addressed in the design and development process.

Each entry in the matrix includes:

- **Requirement ID**: A unique identifier for traceability
- **Requirement Name**: A unique name
- **Primary Implementation Modules**: The system components responsible for fulfilling the requirement

To provide both high-level and detailed perspectives, the requirements matrix is presented in two formats:

- Concise Requirements Matrix
    - This table offers a high level-mapping of each functional requirement to the primary modules responsible for its implementation. It is intended for quick reference and traceability.
- Detailed Requirements Matrix
    - This table expands on the concise version by breaking down each requirement into specific steps, associated UI elements, and the responsibilities of each module during those steps. It provides a stepwise view of system behavior for design, implementation, and testing purposes.

Concise Requirements Matrix:

UC ID	**Functional Requirement Name**	**Primary Implementation Modules**
UC-01	Import DICOM File	DICOM Parser Module
UC-02	Select Anatomical Region	Segmentation Module, DICOM Parser Module, Rendering Engine
UC-03	Visualize and Inspect 3D Anatomy	Segmentation Module, Rendering Engine, DICOM Parser Module
UC-04	Create Polygonal Mesh	Marching Cubes Module, Segmentation Module,

		DICOM Parser Module
UC-05	Generate 3D Print Instructions	G-Code Generator Module, Segmentation Module, DICOM Parser Module

Detailed Requirements Matrix:

Use Case/ Functional Requirement	UI/Frontend	Dicom Parser Module	Segmentation Module	Rendering Engine	Marching Cubes Module	G-Code Generator Module
UC-1: Import Dicom File						
Allow user to select and upload a DICOM file or folder	File upload button, selection and drag-and-drop interface					
Validate DICOM format before processing	Display progress and errors	Validates DICOM files				
Parse DICOM data to extract voxel intensity and metadata		Reads slices, extracts voxel and metadata				
Notify user of success or failure	Pop-up status messages	Returns validation result				
UC-2: Select Anatomical Region						

Allow user to select tissue type (bone, skin, muscle)	Dropdown for tissue selection	Provides metadata for thresholds	Applies segmentation thresholds	Updates 3D preview dynamically		
Filter CT data based on selected tissue		Supplies voxel intensity data	Threshold filtering logic	Visualizes filtered data		
Update visualization automatically when region changes	Handles selection from user		Applies new mask	Rerenders visualization		
UC-3: Visualize 3D Anatomy						
Generate and display 3D preview	“Preview” button and controls	Provides volume data	Provides segmentation data	Renders model		
Allow user to rotate, zoom, and inspect model interactively	Camera and interactions controls			Real-time rendering		
Update visualization dynamically based on user input	Threshold sliders	Provides data	Recomputes segmentation	Updates rendered output		
UC-4: Generate STL File						
Allow user to select STL (.stl) output type	Dropdown file type selection					
Apply Marching Cubes to extract polygonal surface		Supplies Voxel Grid	Provides segmented mask		Surface Extraction	
Generate and export 3D mesh (.STL)	“Export STL” Button		Supplies final segmentation		Encodes STL Mesh	
Notify user when file generation is complete or failed	Pop-up status message				Reports Success / Error	

UC-5: Generate G-Code File						
Allow user to select G-Code (.gcode) output type	Dropdown file type selection					Initializes Module
Generate G-Code reproducing tissue density/opacity		Supplies Voxel/ Mesh data	Provides tissue density mapping			Converts to printer-ready G-code
Validate and save G-Code to user location	“Save G-Code” button					Performs Syntax Validation
Notify user when file generation is complete or failed	Pop-up status message					Performs validation

Appendix A – Agreement Between Customer and Contractor

This System Design Document constitutes a formal agreement between the Customer, Terry Yoo, and the Contractor, The Slice Is Right, regarding the architectural and detailed design for the Enabling 3D Printing of a Medical CT-Scan system. By signing below, both parties acknowledge that this document accurately translates the requirements specified in the Software Requirements Specification into a technical blueprint for system construction. The Customer agrees that this design provides a sufficient and appropriate basis for the system's implementation, and the Contractor agrees to develop a system that faithfully adheres to the design described herein.

Any future changes, additions, or modifications to the design specified in this document must be managed through a formal change control process. A request for change must be submitted in writing by either party and will be evaluated for its impact on the system's architecture, project scope, schedule, and feasibility. An amended version of this SDD, or a formal change order referencing this document, must be mutually agreed upon and signed by authorized representatives of both the Customer and the Contractor before any design changes are implemented in the project.

Name	Signature	Date	Comments
Cooper Stepankiw			
Bryan Sturdivant			
Israk Arafat			
Greg Michaud			
Ethan Wyman			
Terry Yoo			

Appendix B – Team Review Sign-off

This document confirms that all undersigned members of The Slice Is Right project team have thoroughly reviewed the entirety of this System Design Document for the Enabling 3D Printing of a Medical CT-Scan system. By signing below, each team member acknowledges their understanding of the proposed system architecture and detailed design and formally agrees that the content, scope, and technical direction outlined in this document are accurate, feasible, and provide a suitable foundation for the implementation phase of the project.

The comment section provided for each team member is to be used for noting any minor suggestions, editorial feedback, or non-substantive points of clarification. It is recognized that for this sign-off to be granted, there are no major points of contention regarding the architectural or detailed design decisions outlined within this SDD.

Name	Signature	Date	Comments
Cooper Stepankiw			
Bryan Sturdivant			
Israk Arafat			
Greg Michaud			
Ethan Wyman			
Terry Yoo			

Appendix C – Document Contributions

This appendix details the contributions of each team member to the creation of this System Design Document. All members participated in the collaborative design process, including architectural planning, component specification, diagramming, writing, and review to ensure a comprehensive and technically sound document. The percentage contributions are estimates that reflect the primary authorship and development effort for the various sections and diagrams.

Member Name	Primary Responsibilities	Estimated Percentage
Israk Arafat	Architectural Design	20
Gregory Michaud	Introduction Sections	20
Cooper Stepankiw	Section 2.2 Appendix A,B,C	20
Bryan Sturdivant	Requirements Matrix	20
Ethan Wyman	Requirements Matrix, Formatting	20
Total		100%