The Edward S. Rogers Sr. Department of
Electrical and Computer Engineering
University of Toronto

ECE496Y Design Project Course

Group Final Report

Web-based Photodynamic Therapy Planning and Optimization

Team 2020181

Supervisor: Prof. Vaughn Betz
Co-Supervisor: Dr. Lothar Lilge
Administrator: Ms. Inci McGreal
Section: 5
Submission Date: March 30, 2021

Zihao (Samuel) Chen	zihao.chen@mail.utoronto.ca
Xiao Ying (Helen) Dai	helen.dai@mail.utoronto.ca
Shengxiang (Bob) Ji	shengxiang.ji@mail.utoronto.ca

Attribution Table

Section	Student Names		
	Shengxiang Ji	Xiaoying Dai	Zihao Chen
Group Highlights		RS, RD, ET	
Individual Contributions	RS, RD, MR, ET	RS, RD, MR, ET	RS, RD, MR, ET
Acknowledgements		RS, RD, ET	
Executive Summary			RS, RD, ET
Background and Motivation		ET	RS, RD, ET
Project Goal and Requirements	ET		RS, RD, ET
System-level overview	RS, RD, ET		ET
Module-level description and Designs	RS, RD, MR, ET	RS, RD, MR, ET	RS, RD, MR, ET
Assessment of final design	RS, RD, ET	MR, ET	
Testing and Verification	RS, RD, MR, ET	RS, RD, MR, ET	RS, RD, MR, ET
Summary and Conclusions		RS, RD, MR, ET	
Gantt Chart History	RS, RD, MR, ET		
Financial Plan		RS, RD, MR, ET	
All	FP, CM	FP, CM	FP, CM

Abbreviation Codes:
RS – responsible for research of information
RD – wrote the first draft

MR – responsible for major revision
ET – edited for grammar, spelling, and expression
FP – final read through of complete document for flow and consistency
CM – responsible for compiling the elements into the complete document

Signatures:
By signing below, we verify that we have read the attribution table and agree that it accurately reflects our contribution to this document.

Name	**Signature**	**Date**
Zihao (Samuel) Chen	Zihao Chen	Mar 30, 2021
Xiao Ying (Helen) Dai	Xiao Ying Dai	Mar 30, 2021
Shengxiang (Bob) Ji	Shengxiang Ji	Mar. 30, 2021

Voluntary Document Release Consent Form

To all ECE496 students:

To better help future students, we would like to provide examples that are drawn from excerpts of past student reports. The examples will be used to illustrate general communication principles as well as how the document guidelines can be applied to a variety of categories of design projects (e.g. electronics, computer, software, networking, research).

Any material chosen for the examples will be altered so that all names are removed. In addition, where possible, much of the technical details will also be removed so that the structure or presentation style are highlighted rather than the original technical content. These examples will be made available to students on the course website, and in general may be accessible by the public. The original reports will not be released but will be accessible only to the course instructors and administrative staff.

Participation is completely voluntary and students may refuse to participate or may withdraw their permission at any time. Reports will only be used with the signed consent of all team members. Participating will have no influence on the grading of your work and there is no penalty for not taking part.

If your group agrees to take part, please have all members sign the bottom of this form. The original completed and signed form should be included in the hardcopies of the final report.

Sincerely,
Khoman Phang
Phil Anderson
ECE496Y Course Coordinators

Consent Statement:
We verify that we have read the above letter and are giving permission for the ECE496 course coordinator to use our reports as outlined above.

Team #: 2020181
Project Title: Web-based Photodynamic Therapy Planning and Optimization
Supervisor: Prof. Vaughn Betz
Co-Supervisor: Dr. Lothar Lilge
Administrator: Ms. Inci McGreal

Name	Signature	Date
Zihao (Samuel) Chen	Zihao Chen	March 30, 2021
Xiao Ying (Helen) Dai	Xiao Ying Dai	March 30, 2021
Shengxiang (Bob) Ji	Shengxiang Ji	March 30, 2021

Group Highlights (Author: X. Dai)

As of March 30, 2021, all primary functional requirements and constraints of the web application have been met, providing users with an intuitive web interface for running FullMonte simulations, PDT-SPACE optimizations, and visualizations of results, on any popular browser.

The team has successfully achieved launching FullMonteWeb jobs on any user-specified AWS EC2 computing server. We have also implemented an interactive visualization interface, composed of a 3D mesh visualizer and a dose-volume histogram, to view and analyze simulation results. Furthermore, a history page has been implemented for users to download any important files from their simulation, optimization, and visualization tasks so that they can keep the files for future reference.

In addition to these key features, the team has provided a detailed tutorial page on the website to guide users through setting up an EC2 computing instance, which is a prerequisite for using any key features of the website. If the user attempts to launch simulation, optimization, or visualization before setting up an EC2 instance, they will be automatically redirected to the AWS setup page with an informational warning. If an EC2 instance is used for the first time, we have implemented scripts to automatically set it up with necessary libraries and software without any user interference required.

The most important components are the simulation, optimization, and visualization interfaces. We are providing users with a series of forms to enter their input parameters, laid out in a logical sequence to guide them through the process. We have also carefully chosen the wordings in the instructions for clarity, which was reinforced in several iterations based on feedback from our test users. In addition, the website provides links to additional documentation for tricky terminologies and obscure input parameters.

Individual Contributions

Shengxiang (Bob) Ji

I mainly contributed to the infrastructure development and maintenance of the website. In the early phase of the project, I designed and implemented the framework to connect to the remote AWS computing server by DNS name and permission files. This helped the website to avoid handling billing of the computational resources, and therefore users can choose the amount of computational power they wish to purchase. I also contributed to write scripts to automate environment configuration and software installation on AWS instances. In addition, I implemented simulation history logging, so users are able to view and download the raw output of their simulation runs, if they wish to process them further.

In the second reporting period, I implemented support for the GPU version of FullMonte, allowing users to utilize even more powerful computational resources compared to the multi-core CPU version. As the major functionalities of the website are being implemented, I tested the website on larger inputs to see its robustness. Later I fixed a memory bottleneck issue on the web hosting server when the mesh file was being transferred to the AWS remote compute server. Originally the entire mesh file was stored in the memory before copying over to AWS, and the memory consumption was controlled by loading and transferring the files piece by piece.

Zihao (Samuel) Chen

One major contribution I made to the project is designing the execution flow of PDT-SPACE optimization. The execution flow includes setting up the PDT-SPACE environment for users, acquiring input from users, generating parameters script, launching PDT-SPACE optimization, displaying running progress and output display and visualization. The environment setup happens when the user uploads the AWS instance credentials and our design will transfer all required scripts to the user's AWS instance. I also designed the web pages for users to enter or select input parameters. After specifying all input parameters, the user only needs to click a button to launch the PDT-SPACE optimization for the user. Moreover, after the optimization completes, I designed the web page to display outputs from optimization and the algorithm to generate the dose-volume histogram. By designing like this, the PDT-SPACE workflow abstract away the complexity of running the software that requires extensive computer knowledge, and the user only needs to interact with simple text boxes and plain text.

Another contribution is that I designed the algorithm for checking the progress of FullMonte simulation and PDT-SPACE optimization, and developed the web pages to display status and progress for users. When the optimization is running, our design will constantly read the log file on the user's AWS instance and determine the running status and progress. The web page will automatically refresh every five seconds to update the running information for the user. With this feature, the user is able to keep track of the running process if the runtime of simulation and optimization is very long.

Xiao Ying (Helen) Dai

I was mainly responsible for implementing the visualization functionalities of the website. To allow for 3D visualization of VTK files, I conducted research on several possible options and selected ParaView Visualizer, which is a third party application that best suits our requirements. I then augmented Shengxiang's AWS setup scripts to automatically install ParaView Visualizer on the user-specified EC2 instance if it is not already detected. FInally, I added code to run the setup script and launch ParaView Visualizer on the user's EC2 instance. For the dose-volume histogram feature, I wrote the algorithm to generate dose-volume data as well as the scripts to plot and display the data on an interactive graph.

Another one of my main contributions was testing different web hosting options. The previous Capstone team that initiated the FullMonteWeb project deployed the website on Heroku web hosting services. We wanted to migrate the website to a more powerful web hosting service from Amazon, and I discovered that LightSail and Elastic Beanstalk both fit our needs. I tested each of their work flows by deploying our website to the services and compared all options against each other. Our final decision was to continue using Heroku but document the other options for possible future improvements. All investigations are documented in Appendix O, and the test deployment sessions are checked into our project as a separate branch on GitHub, which can be found as a link in Appendix L.

My final contribution was re-designing the overall website layout for better aesthetics and easier navigation. I used the Bootstrap library for website designs, which provides design templates for website components such as headers, footers, animations, and table formats.

Acknowledgements (Author: X. Dai)

The team would like to express our sincere gratitude to Professor Vaughn Betz (supervisor) and Professor Lothar Lilge (co-supervisor) for their extensive support and guidance throughout the course of this project. Professor Betz took time out of his busy schedule to host all our bi-weekly meetings, and his enthusiasm, patience, critiques, and practical advice helped us tremendously during the process. Professor Lilge also provided important insights on the usability aspect of the website, which were integral to the success of our project.

We would also like to express our special thanks of gratitude to Abdul-Amir Yassine (PhD student) and Fynn Schwiegelshohn (Postdoctoral researcher), who are the main contributors of the FullMonte and PDT-SPACE software. They were our point of contact for all technical questions regarding the software packages, and they always kindly provided us with detailed clarifications and helpful suggestions when we encountered problems beyond our expertise.

Last but not least, we would like to thank Professor Lilge's three students for testing the website functionalities. We received helpful feedback from them and were able to improve the usability of our web application in response.

Without any of these people, the project would not have been made possible.

Executive Summary (Author: Z. Chen)

PDT-SPACE optimizer and FullMonte simulator are software packages designed to generate optimized light probe placement plans for Interstitial photodynamic therapy (iPDT), which is a modern cancer treatment method that uses light-sensitive drugs to destroy tumours. The major problem with these two tools is that they require computer programming skills for setup and operations, making them inaccessible to most medical researchers in the field. To solve this problem, we introduce FullMonteWeb.

FullMonteWeb is a web-based application that serves to abstract away the complications and allows users to interact with the FullMonte and PDT-SPACE software using simple buttons and text field inputs. For the final design, FullMonteWeb uses AWS EC2 (Amazon Web Services Elastic Compute Cloud) as the remote computing resource for running optimization, simulation, and visualization tasks, and provides effortless environment setup on any EC2 instance owned by the users. The user is able to select, specify, and upload all the input parameters to the website, then launch simulation or optimization on the specified EC2 instance. When a simulation or optimization task is launched, FullMonteWeb will display the running status, progress and the elapsed time. When the simulation or optimization task is finished, the user can read the raw simulation output, view the dose-volume histogram, and visualize the output mesh in 3D on the website. Moreover, FullMonteWeb also provides simulation history logging and allows users to download the output files they previously generated.

All seven functional requirements and two constraints of the project were accomplished and verified by multiple tests. Two out of three optional functional requirements that represent nice-to-have features of the design have been met based on the results from verification and testing. One objective regarding the ability for all users to use the website without developer assistance has not yet been tested, and the objective related to the responsiveness of web pages failed as the redirect time of three web pages is slightly longer than the threshold. Overall, our design meets the goal of allowing users who do not have extensive computer engineering knowledge to use FullMonte and PDT-SPACE software.

Table of Contents

1. Introduction (Author: Z. Chen)

This report illustrates the final design of the "Web-based Photodynamic Therapy Planning and Optimization" project for the design project course, ECE496. Details regarding the background, motivation, and project requirements are provided in this section. The next few sections of this report will focus on the implementation, test and verification of the final design, and how the final design accomplishes the project requirements of our project.

Besides FullMonteWeb (our web application), FullMonte, and PDT-SPACE, Please also note the definitions to the following key terminologies that will appear throughout the rest of this report:

- Input Mesh: .vtk file generated from multiple layers of DICOM medical images to describe the complex tissue in 3D; an input file for FullMonte and PDT-SPACE. [3]
- Output Mesh: .vtk file output from FullMonte simulator that contains additional information from the simulation results, such as light fluence (optical energy delivered per unit area); an input file to the Visualizer module in our website. [3]
- EC2 instance: Amazon Web Services (AWS) Elastic Compute Cloud; a remote cloud computing server that can be acquired by paying some fees. "EC2 instance", "EC2 computing node", and "EC2 computing server" all refer to the same thing and are used interchangeably throughout this report.
- Fluence: Energy, in terms of joules or watts, absorbed by a given area of the mesh.

1.1 Background and Motivation

Interstitial photodynamic therapy (iPDT) is a modern cancer treatment method that eradicates tumors using light sensitive drugs activated by a set of light probes [1]. In order to only activate drugs in the targeted tumor while minimizing damage to the healthy organ tissues surrounding it, the positions of the light probes and their intensities are the key. This is where PDT-SPACE and FullMonte come in – PDT-SPACE intelligently searches for possible light probe setups, it then uses FullMonte to evaluate the various promising choices by simulating their light propagation and absorption behaviors, and it eventually outputs a set of information, including the optimized number of light probes needed, their powers, and their positions for the specified iPDT treatment plan [2]. Despite being promising tools for iPDT treatment planning, both the FullMonte and

PDT-SPACE software packages rely heavily on the command line interface and programming knowledge for setup and operations, making them impractical for medical practitioners to use. Therefore, there is a gap to build an intuitive and accessible user interface for PDT-SPACE and FullMonte in order to promote a more widespread adoption of the tools.

1.2 Project Goals and Requirements

The goal of this project is to design a web interface that allows users who do not have extensive computer engineering knowledge to run FullMonte and PDT-SPACE, visualize the simulation and optimization results, and download output files.

Table 1: Requirement Table

ID	**Project Requirement**	**Description**
F1	Input: source parameters, optical properties and mesh files	**Functional requirement**: The user must be able to specify all the light source parameters, optical properties, and upload or choose existing mesh files.
F2	Output: downloadable raw simulation results	**Functional requirement**: After running the simulation, the users must be able to download VTK files and generate Dose-Volume histograms.
F3	Interactive visualization	**Functional requirement**: The users are able to visualize output mesh interactively by 3D visualization.
F4	Amazon Web Service Backend	**Functional requirement**: The design must use Amazon Web Services as a backend for user authentication, simulation and optimization.
F5	Integrating Amazon compute cloud	**Functional requirement**: The design must use Amazon compute cloud as computing resource, so the users are able to pay for more computing power to shorten compute time.
F6	Add PDT-SPACE as the optimizer	**Functional requirement**: The design must support PDT-SPACE for optimization.

F7	The design must be able to keep track of the progress of the task and display status to users.	**Functional requirement**: Since optimization tasks take a long time to finish, the user must be able to check the status of the task and be reminded when the task is finished.
C1	The design must allow two or more users to operate simultaneously on different accounts.	**Constraint**: The design must allow two or more users to use it simultaneously without conflicts and degraded performance on file loading, page redirection, task completion and visualization rendering.
C2	Browser compatibility	**Constraint**: The visualizer must be able to work correctly in the stable releases of popular browsers, including Google Chrome, Mozilla Firefox, Internet Explorer, and Safari.
O1	Users are able to use the design without outside assistance.	**Objective**: Users should be able to understand how to use the design and make use of all features successfully by watching tutorial videos or reading user instructions.
O2	The design should be responsive; latency should not exceed 800 ms	**Objective**: Users should be able to interact with the design quickly and smoothly without obvious lag, i.e. latency should not exceed 800ms[3].
OF1	Visualize input mesh files and Interactive set light source	**Optional functional requirement**: The users must be able to visualize the input mesh file. Users must also have the ability to set the position of light sources interactively by dragging the cursor in the visualizer.
OF2	Visualization for PDT dose volume histogram	**Optional functional requirement**: After PDT-SPACE optimization, the design must be able to visualizePDT dose volume histogram for users.
OF3	GPU support	**Optional functional requirement**: The design must be able to run simulation using GPU.

2. Final Design

2.1 System-level Overview (Author: S. Ji)

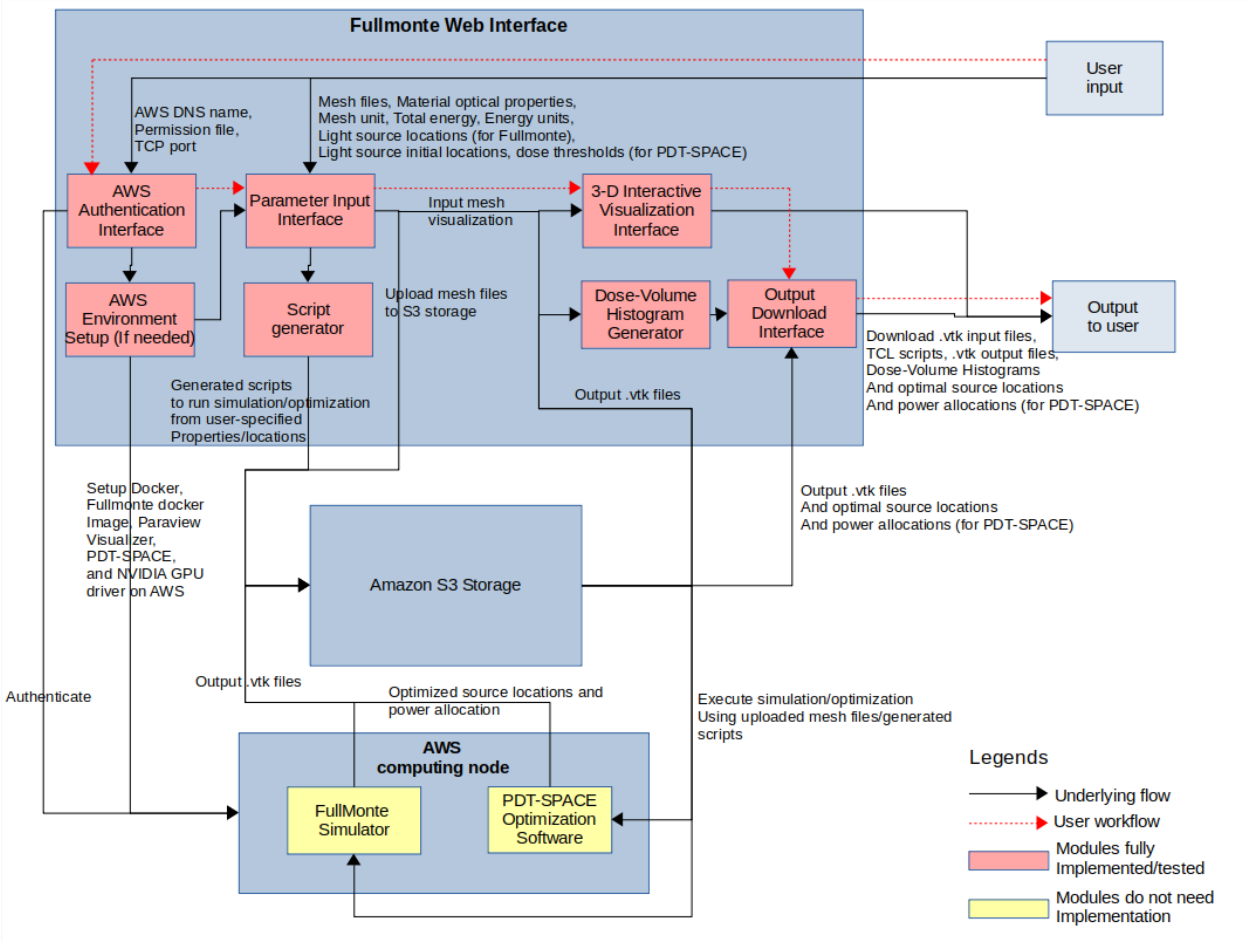


Figure 1. System block diagram for the FullMonteWeb web application.

At a high level, the FullMonte web interface first generates scripts using optical properties, lightsource locations, and dose thresholds from user inputs. It then sends the generated scripts and the user-uploaded meshes to an AWS computing node, and executes the simulation or optimization software. When the execution is finished, the web interface saves the output files from the AWS computing node for visualization and makes it available for users to download. From the user's perspective, the web interface abstracts away the underlying software and convoluted interconnections, and they only need to interact with intuitive buttons and text fields.

2.2 Module-level Description and designs (Author: X. Dai)

The FullMonteWeb application is built on the Django web framework, which is an open source Python-based framework designed for developing web applications. It was chosen by the previous Capstone team that initiated this project, and switching to a different one would mean refactoring the codebase for almost all existing parts of the website. Upon research and careful considerations, we decided to continue using the Django framework because it is very well designed for developing secure and scalable web applications while also providing functionalities readily for use, such as the user authentication system, so that web developers do not have to reinvent the wheels. Therefore, there is no reason for us to select an alternative. The application is hosted on Heroku Web Hosting Services, and a link to the website is located in Appendix L. Heroku was chosen by the previous Capstone team for its stability and Django compatibility, and we have tried several other web hosting services for comparison, such as Amazon Lightsail and Amazon Elastic Beanstalk. Heroku offers the easiest and cleanest deployment process, but we also found that it is not very suitable for larger scale applications, so although it suits our current requirements, it is possible to migrate the website to Amazon's web hosting services in the future. A detailed investigation of website hosting options can be found in Appendix M. The following subsections describe our application modules in detail.

2.2.1 AWS Authentication Interface Module (Author: S. Ji)

Inputs: ● AWS computing node DNS (Domain Name System) name ● User-uploaded permission file to access the AWS computing node ● AWS computing node TCP port for visualizer connection ● Whether the AWS computing node contains a GPU
Output: If the uploaded DNS name and permission file cannot be used to access an AWS computing node, then an error is reported to the user; otherwise users will be redirected to run the simulation or visualization.
Function:

This module reads the user-uploaded DNS name and permission file and tries to use them to perform a remote SSH (Secure Shell) operation to connect to an AWS computing node. If the connection is successful, it will detect whether the computing node is set up with the required environment and software (Docker, FullMonte, Paraview Visualizer, PDT-SPACE, and NVIDIA GPU driver if the server contains a GPU) to perform the simulation or optimization. If not, it will proceed to the AWS Environment Setup Module.

2.2.2 AWS Environment Setup Module (Author: S. Ji)

Input: Connection to an AWS computing node
Output: Redirect users to simulation or visualization module
Function: This module executes five bash scripts on the connected AWS computing node, which installs Docker, FullMonte, Paraview Visualizer, PDT-SPACE, NVIDIA GPU driver respectively, as well as their dependencies. After the scripts finished execution, users will be automatically redirected to run the simulation for visualization.

2.2.3 Parameter Input Interface Module (Author: Z. Chen)

Input: Connection to an AWS computing node
Output: Redirect user to wait for simulation/optimization results
Function: FullMonte Simulation: This module spans across multiple web pages, and redirects users in between. As the first step, it asks users to upload mesh files, or to choose one from the list of mesh files that they used in previous simulations. It also asks users to specify the unit of the mesh file (centimeters or millimeters), the types of energies to record, number of photons used, total energy used and the unit of the energy (Watts or Joules). The second step is to specify the optical material properties, and the unit of preset optical materials will be updated to correspond to the unit of

the user-specified mesh. The third step is to specify the locations of the light sources. As of the submission of this report, the supported light source types are Point, Pencil Beam, Volume, Ball, Cylinder and Surface Source Builder. Validators are used on all input data to prevent the user input data format that is not supported by the simulator. The website will then invoke the Script Generator Module with the user specified parameters. As the final step, the website will list the user-specified parameters for the user to confirm before submitting for execution, as well as allow the user to download the generated script and modify it as needed. After the user confirms the parameters, the web interface sends the script, along with the user uploaded mesh, to Amazon S3 storage and the connected AWS computing node, and launches the simulation/optimization, and redirects the user to a page to wait for the results.

PDT-SPACE optimization:

The first step is to select an available input mesh and specify the relevant parameters like optical properties. The user then needs to select the source type, placement type and upload a placement file that contains the locations of light placement. Then all input parameters will be saved into the database. The Script Generator Module will use these input parameters to generate a parameter script that contains all input parameter names and values for PDT-SPACE. The parameter script then will be transferred to the user's AWS instance for launching PDT-SPACE optimization. To launch the optimization, the first step is to connect to the user's AWS account with the user's credentials so we can run any commands on the user's AWS account remotely. To launch the PDT-SPACE job, our design will initiate a new process using Python multi-process library and then execute two scripts that are created during the AWS setup module. The first script is to download and open the docker image which contains the PDT-SPACE library on the user's AWS account. The second script is to set the running environment and execute the PDT-SPACE command with the parameter file generated in the previous module.

FullMonte Simulation and PDT-SPACE optimization:

After the simulation/optimization launched, our design will check whether the process is finished and display the running progress to the user. When the process is running, all output

from the terminal will be saved into a log file, so our design is able to acquire the status and progress by reading the log file on AWS instance and then display that information on the web page. The web page will be automatically refreshed every five seconds to update the latest running status and progress for the user.

2.2.4 Script Generator Module (Author: S. Ji)

Input:

- Name of the user-uploaded mesh
- Optical properties of the mesh
- Locations of the lightsource (for Fullmonte)
- Initial locations of the lightsource and does threshold of the target (for PDT-SPACE)

Output: A TCL/OP script with instructions to run the simulation/optimization

Function:

This Module reads user-specified parameters stored in the session dictionaries in a Django request, and generates a script with the required format for the FullMonte simulator (TCL script) and the PDT-SPACE optimizer (OP script). The generated script will be stored in Amazon S3 storage for the AWS computing node to use.

2.2.5 3D Interactive Visualizer Module (Author: X. Dai)

Input: Output VTK files from the simulation/optimization

Output: Interactive visualization to user

Function:

This module reads a user-specified VTK file (mesh file), renders the 3D geometry of the body tissue regions described in the file, and displays the region as a 3D model. The interface allows users to interact with the displayed model by zooming in and out, moving and rotating, and adding rendering filters such as the slicing plane to view the mesh's internal contents. Users can launch the module in several situations:

1. Before launching FullMonte simulation, users can click on the "Open 3D Interactive

Visualizer" button (Figure 16 on pg.40) on the light source parameters input page to automatically view their input VTK file. This serves to help users choose desired light source positions for simulation.

2. After FullMonte simulation completes, users can click on a button to automatically view the generated mesh file that describes simulated distribution of fluence in addition to the 3D geometry of the region itself. This is important because it allows users to have a more thorough understanding of the simulation results for treatment planning.
3. After PDT-SPACE optimization, users can click on a button to automatically view the generated mesh file that describes simulated results from the best light source placement and power allocation determined by PDT-SPACE.
4. Users can also launch the visualizer any time as a standalone feature. We have provided an option for users to upload any VTK file of their choice, which could be output mesh files downloaded from a past simulation or optimization.

To achieve this functionality, the team researched and attempted several possible solutions. The final design that best aligns with our requirements makes use of a third party 3D visualization software called ParaView Visualizer, and how we are interfacing with this software is a computer networking process best described in pictorial form. As shown in Figure 2 below, when the visualizer module is launched, our website will run a script to start the ParaView desktop application installed on the user's remote EC2 server instance, ParaView will then locally render the specified VTK file located on the EC2 instance and send the processed images back to the user's browser through the internet using a communication endpoint. All transfer of files are done using the Paramiko library, which is an implementation of the Secure Shell (SSH) cryptographic network protocol for operating network services securely. The version of ParaView Visualizer that is compatible with this process does not currently support some advanced features offered by the desktop version, so as an immediate next step, we will be creating a page that informs users what can or cannot be supported by the website, and suggests them to use the desktop version instead if they would like more advanced options.

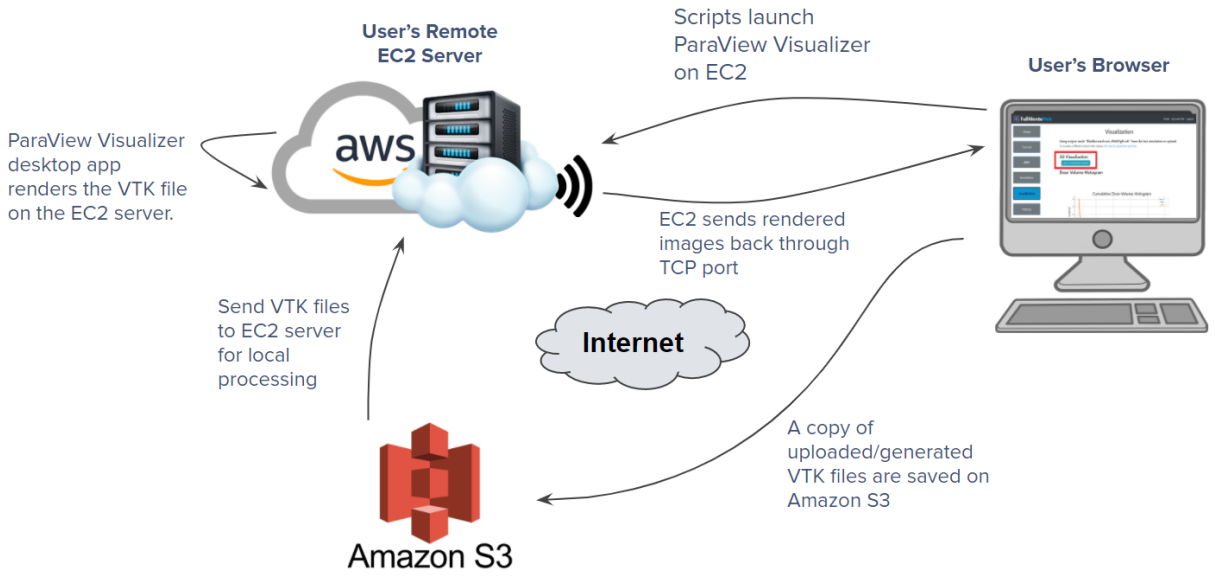


Figure 2. 3D visualization module networking architecture.

2.2.6 Dose-Volume Histogram Module (Author: X. Dai)

Input: Input VTK files/Output VTK files from the simulation/optimization
Output: A Dose-Volume histogram of the simulation results
Function: This module reads a specified VTK file (mesh file) output from FullMonte simulation or PDT-SPACE optimization, generates a dose-volume histogram, and displays the histogram as a 2D plot on the user's browser. A dose-volume histogram serves to show the distribution of fluence (energy intensity) throughout each body tissue region described in the VTK file. An example of the histogram can be found in Figure 4 on pg.18. The x-axis represents the percentage of fluence with respect to the maximum fluence absorbed by any volume of tissue in the entire mesh, and the y-axis represents the percentage of a region that is exposed to at least a given dose of energy. The histogram is calculated separately for each tissue region (material type) present in the mesh, and the results from each region are stacked on a multiple line graph. The module is further divided into two submodules, one that plots the histogram for FullMonte results and another that plots the histogram for PDT-SPACE results. This is because PDT-Space already generates dose-volume histogram data in a convenient file format, so we

can extract the data and plot the histogram without additional computation. FullMonte also generates histogram data, but the format is a bit tricky to work with, so we decided to implement our own algorithm to generate dose-volume histogram data from raw results present in the output VTK files. The calculations are performed by a Python script that runs on a separate process when the visualization page is selected, and a flowchart describing the algorithm is shown in Appendix N. Our script then uses Matplotlib, a Python-based graphing library, to plot the graph, and leverages the Mpld3 library to add interactive components and convert the plot into an HTML string for display on the browser. This process takes a few minutes to complete, so users are redirected to an informational waiting page in the meantime. Once the graph is displayed, users can interact with the plot by zooming in and out, toggling region data on and off, and choosing whether or not to show markers for data points. When markers are turned on, the user can hover the mouse over them to view their x and y values. Finally, the generated Dose-Volume histogram (both the data in a CSV file and the plot in a PNG file) will be stored into the Amazon S3 storage and be made available for the user to download.

2.2.7 Output Download Interface Module (Author: S. Ji)

Input: The current logged in user
Output: Send files to user's browser
Function: This module locates the user in the database to see their simulation histories. The histories will be shown in tabular form, showing the simulation type, input mesh file, TCL script used to run the simulation, output VTK file, output TXT file, output DVH data as a spreadsheet, output DVH data as a line chart, and the finishing time of the simulation. All the above mentioned files are hyperlinked to their stored locations in the Amazon S3 storage, and users can click to download them to their local devices.

2.3 Assessment of Final Design (Author: S. Ji)

The final design splits the process of running the simulation or optimization software and visualization of the results into multiple stages. Users are directed through the flow with helpful prompts and internal links as guidance, as illustrated in Figure 3 below. In the input stage, validators are placed on all inputs to prevent users from entering invalid parameters. In addition, users also have the ability to manually modify the generated scripts, giving advanced users the flexibility to control fine-grained details of their experiment runs. While FullMonte or PDT-SPACE is running on the remote EC2 instance, the webpage will display the elapsed time as well as the progress of the task as a percentage for the users to keep track of. The visualization module supports dose-volume histogram representation of the simulated energy absorption distributions and 3D visualization of the output mesh. All files are stored in the Amazon S3 storage, and users can choose to download them should they wish to do further processing on their local machines. All the computationally expensive operations are done remotely on AWS EC2 instances, which users will be able to pay more for more powerful computation resources. Hence, the users' physical devices will not limit the throughput of the experiments. The final design simplifies the complexity of setting up and running the command-line version of the underlying software, and in the meantime provides all the available functionalities of those software.

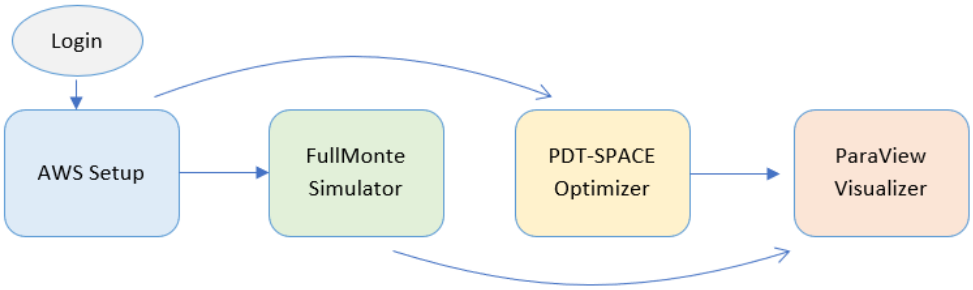


Figure 3. User workflow guided by the website.

The FullMonteWeb web interface is still under active development as of the submission of this report. The website is currently being used and tested by users outside of the team (see section 3.B4 below in Testing and Verifications), and the team is getting feedback from their use cases and experiences. The team is regularly improving the usability of the website by adding more

help pages and buttons, as well as making the website more robust. In addition, the tutorial videos will be filmed at a later date, to give a more comprehensive guide of the entire website.

3. Testing and Verification

3.1 Verification Table (Author: All)

Table 2: Verification Table

ID	Requirement	Verification Method	Verification Result and Proof
F1	Input: source parameters, optical properties and mesh files	**Test**: Try to upload each mesh file from the FullMonte repository (see Appendix F) with corresponding property parameters through the web interface. **Success Criteria**: The simulation is able to run with all the uploaded valid meshes and properties.	Pass. See Appendix C below.
F2	Output: downloadable raw simulation results	**Test**: For each test in F1, try to download the simulation output mesh and raw simulation results through the web interface after the simulation is finished. **Success Criteria**: The user is able to download all the simulation output mesh and raw results.	Pass. See Appendix D below.
F3	Interactive visualization	**Test**: Use the visualizer to visualize each output VTK file from F1 and change the view perspective by dragging the cursor. **Success Criteria**: The visualizer is able to show the 3D geometry and dose-volume histogram of all the mesh files correctly, and the user is able to change the visualization perspective interactively.	Pass with concerns. See Section 3.2 below.
F4	Amazon Web Service Backend	**Review of Design**: Check each stage of the design to see whether it is properly using Amazon Web Services for user authentication, storage, and task running.	Pass. See Section 3.3 below.

		Success Criteria: Amazon Web Service must be used for above activities.	
F5	Integrating Amazon compute cloud	**Test**: Try to launch simulation and optimization with multiple CPU cores (4 or more) using Amazon compute cloud. **Success Criteria**: The simulation and optimization must run successfully. Users are able to reduce the running time if the user pays for more computing resources.	Pass. See Section 3.4 below.
F6	Add PDT-SPACE as the optimizer	**Test**: Try to launch optimization using PDT-SPACE through the web interface. **Success Criteria**: The optimization is able to run correctly and the user is able to download optimization output.	Pass. See Appendix E below.
F7	The design must be able to keep track of the progress of the task and display status to users.	**Test**: While running simulation or optimization, check the progress bar on the webpage. **Success Criteria**: The progress bar must be consistent with the real progress of simulation and optimization.	Pass. See Appendix F below.
C1	The design must allow two or more users to operate simultaneously on different accounts.	**Test**: Two testers, each with a different account, try to proceed through all functional verification tests (F1 - F7) on the web interface at the same time. **Success Criteria**: Both testers are able to complete all tests without conflict and degraded performance.	Pass. See Appendix G below.

C2	Browser compatibility	**Test**: Launch verification test of F3 (see above) using different browsers (Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari). **Success Criteria**: The design must pass the F3 verification test for all aforementioned browsers.	Pass. See Appendix H below.
O1	Users are able to use the design without outside assistance.	**Statistical Measures**: Let 10 users, who have knowledge of photodynamic therapy but have no experience with the web simulator, proceed through all functional verification tests (F1 - F7) as instructed by tutorial videos and instructions on the website. **Success Criteria**: All users must successfully utilize all the provided features by following instructions and watching tutorial videos.	Untested. See Section 3.5 below.
O2	The design should be responsive; latency should not exceed 800 ms	**Test**: Directly measure the time of each web page redirection. **Success Criteria**: Each web page redirection should not exceed 800ms.	Failed. See Appendix I below.
OF1	Visualize input mesh files and Interactive set light source	**Test**: Try to visualize the input mesh and set light source as the user is dragging the cursor. **Success Criteria**: The visualizer shows the 3D geometry of the input meshes correctly and the position of the light source is captured accurately in real time.	Failed. See Appendix J below.
OF2	Visualization for PDT dose	**Test**: Use the visualizer feature to visualize the PDT dose volume histogram. **Success Criteria**: The visualizer must successfully	Pass. See Section 3.6 below.

	volume histogram	visualize the PDT dose volume histogram.	
OF3	GPU support	**Test**: Launch simulation using GPU through the web interface. **Success Criteria**: The simulation must run successfully in GPU mode.	Pass. See Appendix K below.

3.2 Verification for F3 (Author: X. Dai)

After the FullMonte simulation completes, users should be able to see the resulting dose-volume histogram and visualize the output mesh in 3D. Several VTK mesh files were tested for 3D visualization. The mesh files were sent as input to FullMonte for simulation, then the 3D visualization was launched to view each mesh. Base criteria for the 3D visualizer to achieve our requirements include being able to visualize the mesh in 3D, rotate and change perspectives, add a threshold filter, add clip filter, and add slice filter with user-specified position and orientations. We have also implemented additional features for the dose volume histogram so that users can toggle regions on and off, and hover their mouse over the data points for x and y values. The tested mesh files and results are shown below.

Table 3: Testing for 3-D visualization

File Name	Visualize in 3D	Rotate & Zoom	Threshold Filter	Slice & Clip Filter	View in Log Scale	Result
Bladder.mesh	✓	✓	✓	✓	×	Pass
HeadNeck.mesh	✓	✓	✓	✓	×	Pass
HeadNeck_Tumor.mesh	✓	✓	✓	✓	×	Pass
Colin27	✓	✓	✓	✓	×	Pass
Colin27_tagged_tumor_1	✓	✓	✓	✓	×	Pass
Colin27_tagged_tumor_2	✓	✓	✓	✓	×	Pass

Colin27_tagged_tumor_3	✓	✓	✓	✓	×	Pass
Colin27_tagged_tumor_4	✓	✓	✓	✓	×	Pass
test-pancreas-final.mesh	×	×	×	×	×	Fail

Table 4: Testing for DVH visualization

Visualize DVH plot	View Dose Volume Histogram	Toggle DVH markers on/off	Hover mouse over for value	Toggle DVH regions on/off	Result
Bladder.mesh	✓	✓	✓	✓	Pass
HeadNeck.mesh	✓	✓	✓	✓	Pass
HeadNeck_Tumor.mesh	✓	✓	✓	✓	Pass
Colin27	✓	✓	✓	✓	Pass
Colin27_tagged_tumor_1	✓	✓	✓	✓	Pass
Colin27_tagged_tumor_2	✓	✓	✓	✓	Pass
Colin27_tagged_tumor_3	✓	✓	✓	✓	Pass
Colin27_tagged_tumor_4	✓	✓	✓	✓	Pass
test-pancreas-final.mesh	×	×	×	×	Fail

All test cases passed except for the pancreas mesh, which is over 250MB in size. The failure occurred because memory bottlenecked on the Heroku server when processing the mesh for dose-volume histogram generation. We are currently using Heroku's free tier service, which only grants us 512MB of memory, so an easy solution would be to upgrade the plan for more quota on memory. We are currently also trying possible solutions for this problem without having to upgrade our Heroku service, and these are described in detail in Appendix X. The visualizer also does not support the logarithmic scaling feature, which is a nice-to-have feature that is not part of our requirements. The basic functionalities of slice, contour, and clip filters suit the need for most situations, but if the user needs more advanced options, our website suggests they download the output mesh and visualize it on a desktop application instead.

The following figures show the visualization of the simulation output for 'Colin27_tagged_tumor_1'.

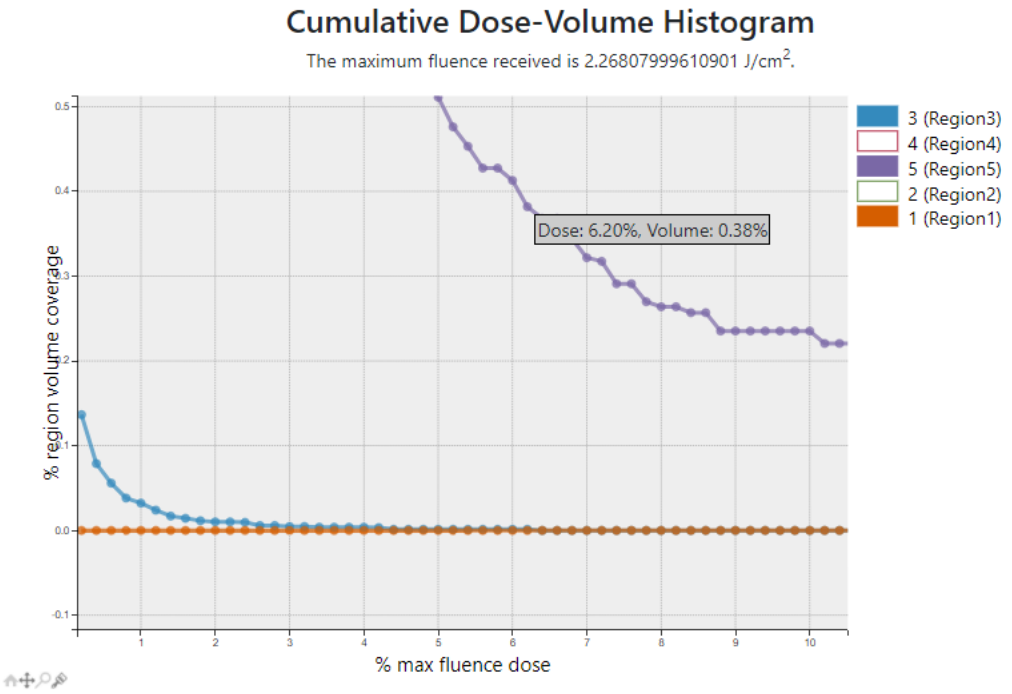
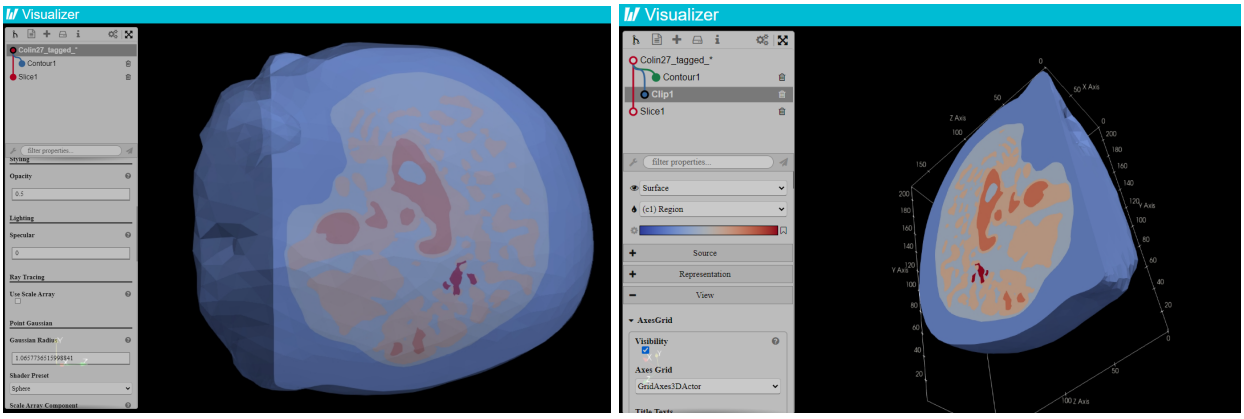


Figure 4. Zoomed-in view of the generated dose-volume histogram with Regions 2 and 4 toggled off, markers turned on, and the mouse hovering over a datapoint.



(a) (b)

Figure 5. 3D representation of the mesh with (a) surface opacity set to 50% and a slice filter applied, and (b) with contour and clip filters applied.

3.3 Verification for F4 (Author: S. Ji)

As the FullMonte simulation is running, the “top” command was used to test if the simulation was using all the computation resources available in the user’s computing node. As shown in the figures below, the simulator is able to correctly determine the number of CPU cores available and automatically run on different CPU cores.

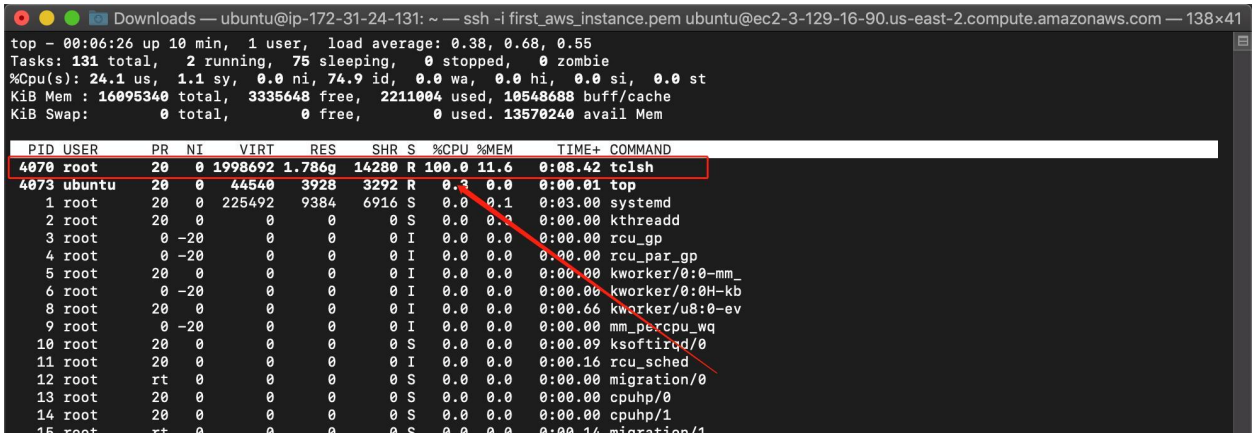


Figure 6. CPU usage on 1-core machine.

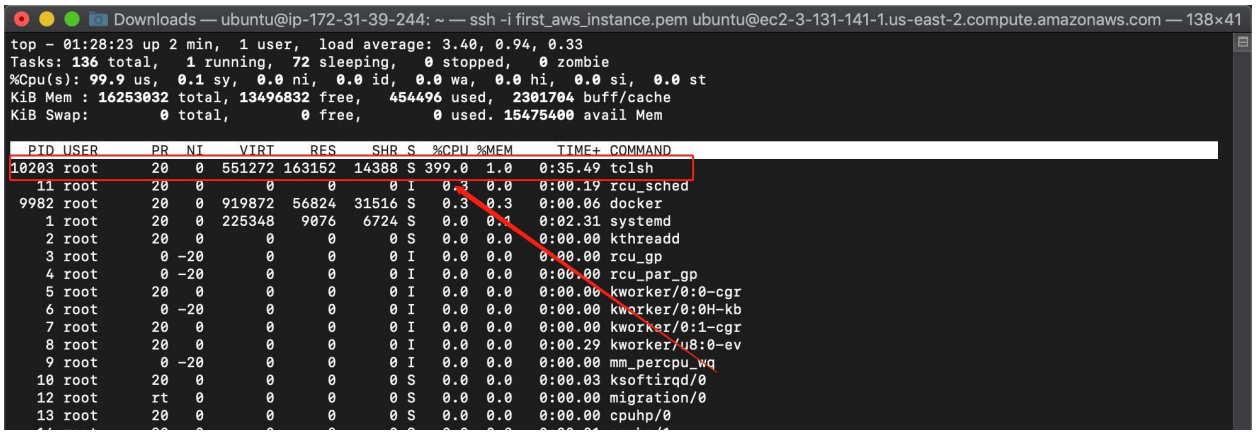


Figure 7. CPU usage on 4-core machine.

3.4 Verification for F5 (Author: S. Ji)

The performance of the FullMonte simulation was tested on different instance types on the AWS computing cloud, described in the table below. Two meshes were tested on different instance types. The first one was the Bladder mesh, which is about 9MB in size, contains about 300 thousand tetrahedrons, and was simulated using 1 million photons. The second one was the Pancreas mesh, which is about 230MB in size, contains about 8 million tetrahedrons, and was

simulated using 100 million photons. The tests showed that users are able to reduce the runtime of the simulation by purchasing more powerful compute instance types.

Table 5: Runtime performance of different instance types

Instance Type	Number of CPUs	Contains GPU?	Price	Runtime Performance on Bladder mesh	Runtime Performance on Pancreas mesh
t2.micro	1	No	$0.0116 USD/hour	1 minute 40 seconds	N/A
t3a.xlarge	4	No	$0.1504 USD/hour	30 seconds	1 hour
g4dn.xlarge	4	Yes	$0.526 USD/hour	<5 seconds	3 minutes

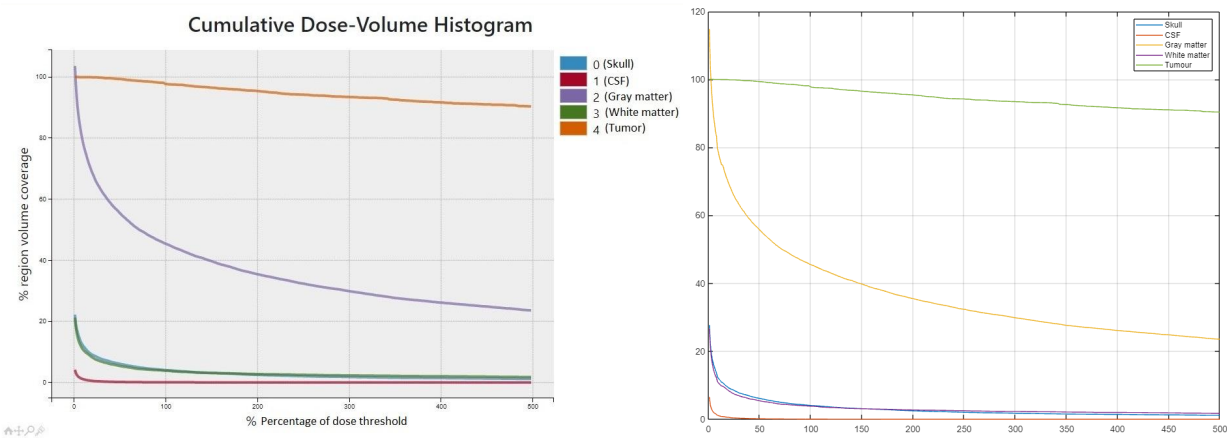
3.5 Verification for O1 (Author: S. Ji)

As of the submission of this report, the website is still being developed, and the tutorial video of the website has not been filmed. Therefore, testing for the intuitiveness of the website has been delayed. However, the website is already being used and tested by one Ph.D. student in medical biophysics and two high school students, with little assistance from the team. The feedback from the users helped the team to improve the usability of the website, and also provided insights into what should be included in the tutorial videos, which will be filmed at a later date. The intuitiveness test will be conducted on more users once the tutorial videos are completed.

3.6 Verification for OF2 (Author: Z. Chen)

After the PDT-SPACE optimization run completes, one of its outputs is a Matlab file that contains data describing the percentage of volume which absorbed at least a corresponding percentage of threshold dose (fluence) for each tissue region. This data is called the dose-volume histogram (DVH). To test whether our design can plot the DVH correctly, we compared the DVH plot from our web interface with the DVH plot generated in Matlab by loading and plotting the file in Matlab manually. Our design passes this test if two graphs are identical.

Figure 8(a) on the left is the DVH from our web page, and Figure 8(b) on the right is the DVH generated by Matlab. We can see that these two graphs are identical with the same DVH data, so our design successfully meets this optional project requirement.



(a) *(b)*

Figure 8. Dose-volume histogram from (a) our design and (b) Matlab.

4. Summary and Conclusions (Author: X. Dai)

FullMonteWeb, our web-based iPDT planning and optimization application, runs on top of three major software packages – FullMonte light simulator, PDT-SPACE treatment optimizer, and ParaView Visualizer. Our team's goal is to put these three pieces of software together into an easy, one-stop service for our website users, who are most likely medical professionals in the field of iPDT research.

As demonstrated through our testing and validation results from Section 3, our project goal is met. In particular, all our functional requirements have been achieved and the website abides all constraints. However, two of our objectives were not achieved. The website did not pass the responsiveness test because two out of the eleven web pages loaded with latency greater than 800ms. To mitigate the effect of these latencies, we can set up alert popups to inform users of the wait, or upgrade to a website hosting plan with more power. We also failed to provide the optional functionality for users to interactively set light source positions for FullMonte input due to infrastructure limitations, but instead, we implemented a 3D visualization of the input mesh that allow users to turn on the Axis Grid to view the xyz coordinates.

Testing and validation were done on extensive, realistic test cases that are likely to be encountered during normal operations, but they were not exhaustive. Namely, we only had six people accessing the website at a time to test simultaneous usage, and no structured procedure to check whether users could use the website without additional assistance. Therefore, if we were to extend the project beyond this course, we would have to invite more people to test the application.

Overall, the project is fully functional and sets a good base for future extensions. Our supervisors and their research teams have good faith in the outcomes of this project, and they are very excited to launch the website for real. Once perfected, our web application could encourage more medical practitioners worldwide to exploit the benefits of this tool for future medical advancements in the field and cancer research.

5. References

[1] G. Shafirstein, D. Bellnier, E. Oakley, S. Hamilton, M. Potasek, K. Beeson, and E. Parilov, "Interstitial Photodynamic Therapy—A Focused Review," Cancers, vol. 9, no. 12, pp. 12, 2017.

[2] A. Yassine, W. Kingsford, Y. Xu, J. Cassidy, L. Lilge, and V. Betz, "Automatic interstitial photodynamic therapy planning via convex optimization," Biomedical Optics Express, vol. 9, no. 2, pp. 898-920, 2018.

[3] Özgür Bal, "Some interesting bits about latency," citycloud.com, Aug.21, 2012 [Online]. Available: https://citycloud.com/city-cloud/some-interesting-bits-about-latency/. [Accessed March. 30, 2021].

Appendix A: Gantt Chart History (Author: S. Ji)

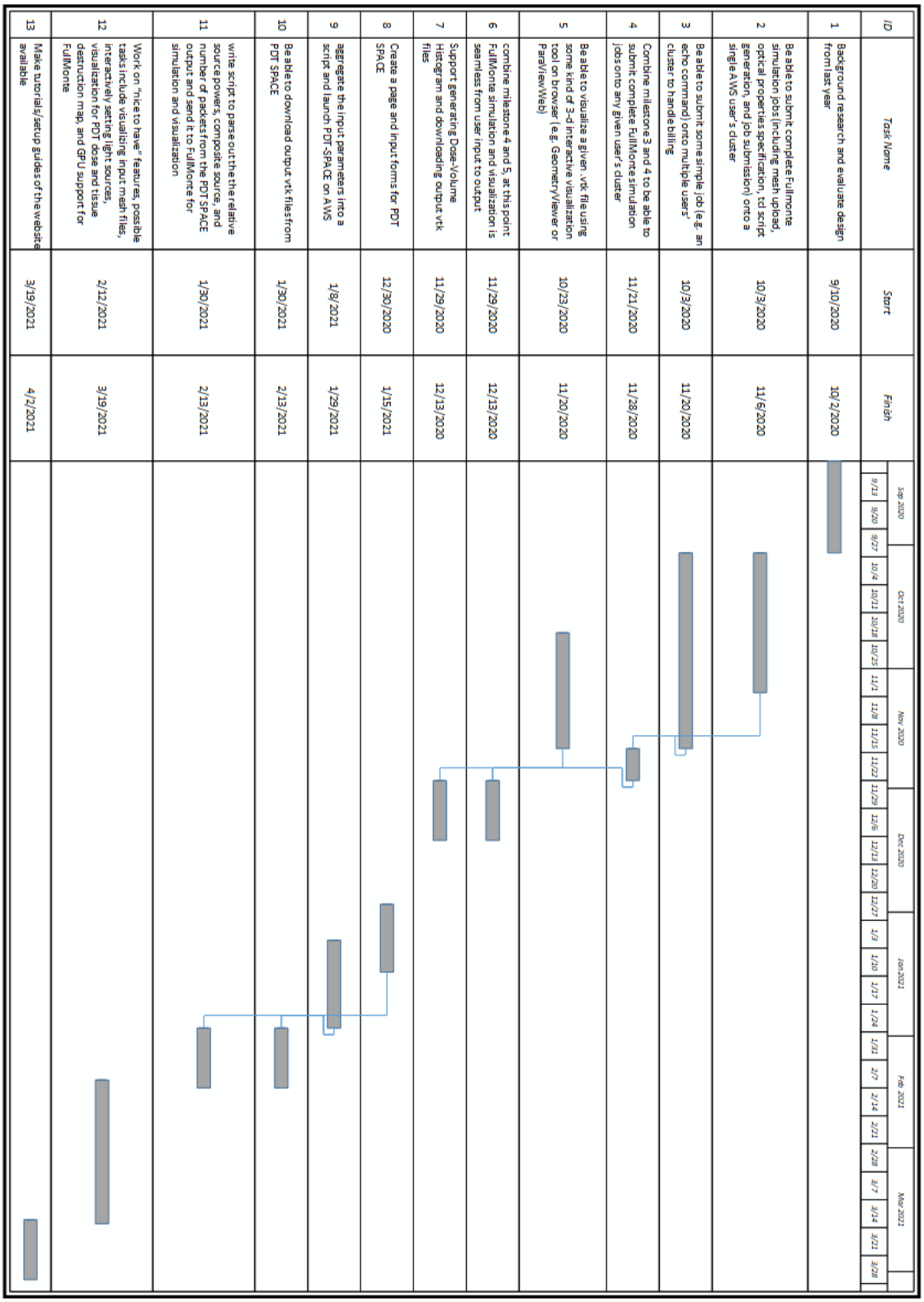


Figure 9. Gantt Chart in Project Proposal.

ID	Task Name	Start	Finish
1	Background research on AWS computing options including EC2 and lambda (Shengxiang)	9/10/2020	10/2/2020
2	Background research on Website pricing information including Heroku and AWS to make decision on where to host the website (Xiaoying)	9/10/2020	10/2/2020
3	Background research on user management and billing options including Amazon cognito and devpay (Zihao)	9/10/2020	10/2/2020
4	Write verification program to setup user authentication using Amazon Cognito to potentially handle billing (Zihao)	10/3/2020	10/23/2020
5	Background research on Website hosting options including Heroku and AWS (Xiaoying)	10/3/2020	10/23/2020
6	Create page to allow users to upload AWS computing node DNS name and permission file (Shengxiang)	10/3/2020	10/16/2020
7	Setup EC2 instance with Docker and Fullmonte for testing (Shengxiang)	10/17/2020	10/30/2020
8	Remotely connect to an AWS EC2 instance with provided DNS name and permission file, and execute some simple command (e.g. and echo command) (Shengxiang)	10/24/2020	11/13/2020
9	Migrate website hosting to Amazon Lightsail and Elastic Beanstalk (Xiaoying)	10/24/2020	11/13/2020
10	Be able to submit complete Fullmonte simulation jobs (including mesh upload, optical properties specification, tcl script generation, and job submission) onto any given AWS computing node (Shengxiang)	10/31/2020	11/20/2020
11	**Milestone – be able to submit Fullmonte simulation jobs onto any given AWS instance to handle billing**	11/21/2020	11/21/2020
12	Display the simulation progress to the user when the simulation is running asynchronously, and redirect the page automatically when finished. (Zihao)	11/14/2020	12/4/2020
13	Visualize .vtk files using Paraview Visualizer on a single AWS computing server for testing (Xiaoying)	11/14/2020	12/4/2020
14	Automatically detect if the provided EC2 instance is a fresh instance; if yes, execute setup script to install Docker and Fullmonte on it (Shengxiang)	12/5/2020	12/18/2020
15	combine task 10 and 13, at this point FullMonte simulation and visualization is seamless from user input to output (Xiaoying)	12/5/2020	12/18/2020
16	**Milestone – Fullmonte simulation is seamless from user input (AWS info, mesh upload, optical properties, source locations) to 3-D interactive visualization and showing Dose-Volume histogram**	12/19/2020	12/19/2020

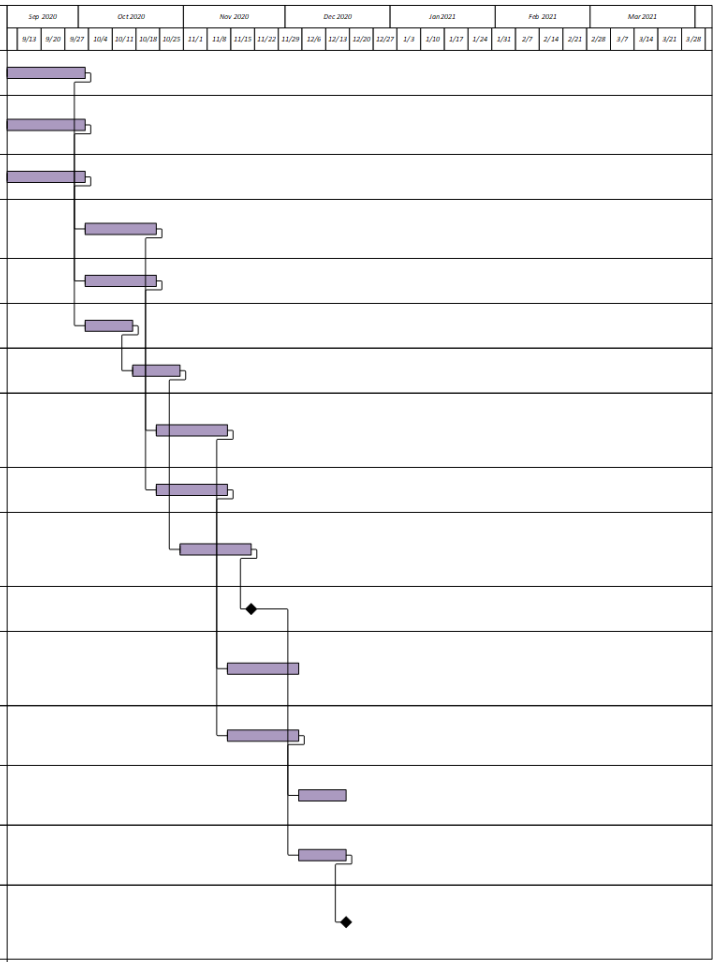


Figure 10. Gantt chart in Progress Report, first half.

ID	Task Name	Start	Finish
17	Setup an EC2 instance with PDT-SPACE for testing (Zihao)	12/30/2020	1/4/2021
18	Link Fullmonte output to the visualizer module and automatically run Paraview Visualizer on any given AWS computing server (Xiaoying)	1/4/2021	1/9/2021
19	Record history of simulation and optimization runs (with user and timestamps), and enable downloading them from Amazon S3 storage (Shengxiang)	1/4/2021	1/15/2021
20	Create a page and input forms for PDT SPACE for users to upload meshes, specify initial lightsource locations and target dose thresholds. (Zihao)	1/5/2021	1/13/2021
21	Allow users to upload their locally-saved .vtk file for viewing (Xlaoying)	1/10/2021	1/11/2021
22	Generate Dose-Volume histogram (DVH) for the simulation output for user-uploaded .vtk files (Xiaoying)	1/12/2021	1/14/2021
23	Generate parameters file (.op) from user-specified input mesh, tissue optical properties, target dose thresholds and initial placement. (Zihao)	1/14/2021	1/14/2021
24	Re-design all web pages using the Bootstrap library with HTML, CSS, and JavaScript (Xiaoying)	1/15/2021	1/16/2021
25	Allow users to view the DVH for any specific material type (Xiaoying)	1/17/2021	1/22/2021
26	Modify database and S3 storage hierarchy, so per-user history (user, timestamp, type of runs, output files) download can be supported (Shengxiang)	1/16/2021	1/29/2021
27	Combine milestone 18 and 19, at this point PDT-SPACE is seamless from user input to output (Zihao)	1/15/2021	2/4/2021
28	**Milestone – PDT-SPACE optimization is seamless from user input (AWS info, mesh upload, optical properties, initial lightsource locations, target dose thresholds) to output (optimized lightsource locations, power allocations)**	2/5/2021	2/5/2021
29	Make Dose-Volume histogram (DVH) spreadsheet available for user to download (Shengxiang)	1/30/2021	2/9/2021
30	**Milestone – Allow each user to download their own simulation and optimization history output files, including .vtk files, Dose-volume histograms and optimized lightsource locations and power allocations for PDT-SPACE**	2/10/2021	2/10/2021
31	write script to parse out the the relative source powers, composite source, and number of packets from the PDT SPACE output and send it to FullMonte for simulation and visualization (Xiaoying)	1/20/2021	2/9/2021
32	Work on "nice to have" features - interactively visualizing input mesh files and setting light source (Xiaoying)	2/12/2021	3/19/2021
33	Work on "nice to have" features - Visualize PDT dose and tissue destruction map (Shengxiang)	2/12/2021	3/19/2021
34	Work on "nice to have" features - GPU support for FullMonte (Zihao)	2/6/2021	3/21/2021
35	Record tutorial video for AWS setup, mesh upload, and launching simulation (Shengxiang)	3/20/2021	4/3/2021
36	Record tutorial video for visualization and download output files (Xiaoying)	3/20/2021	4/3/2021
37	Make tutorials/setup guides of the website available (Zihao)	3/22/2021	4/5/2021

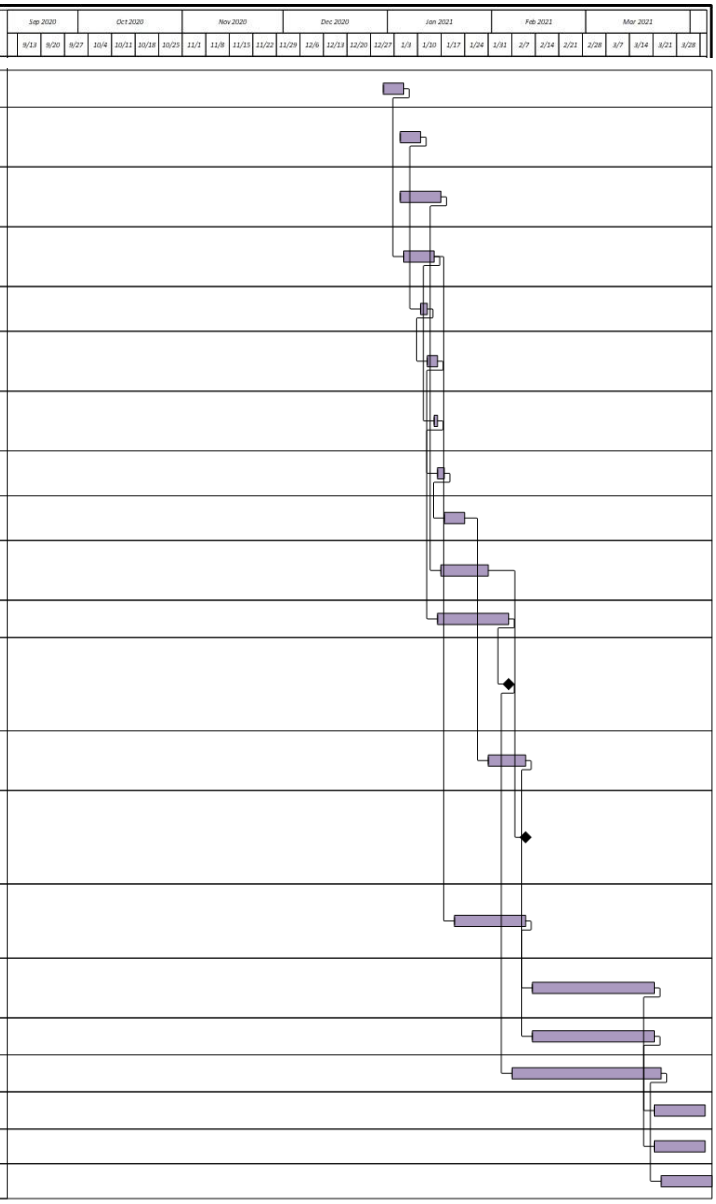
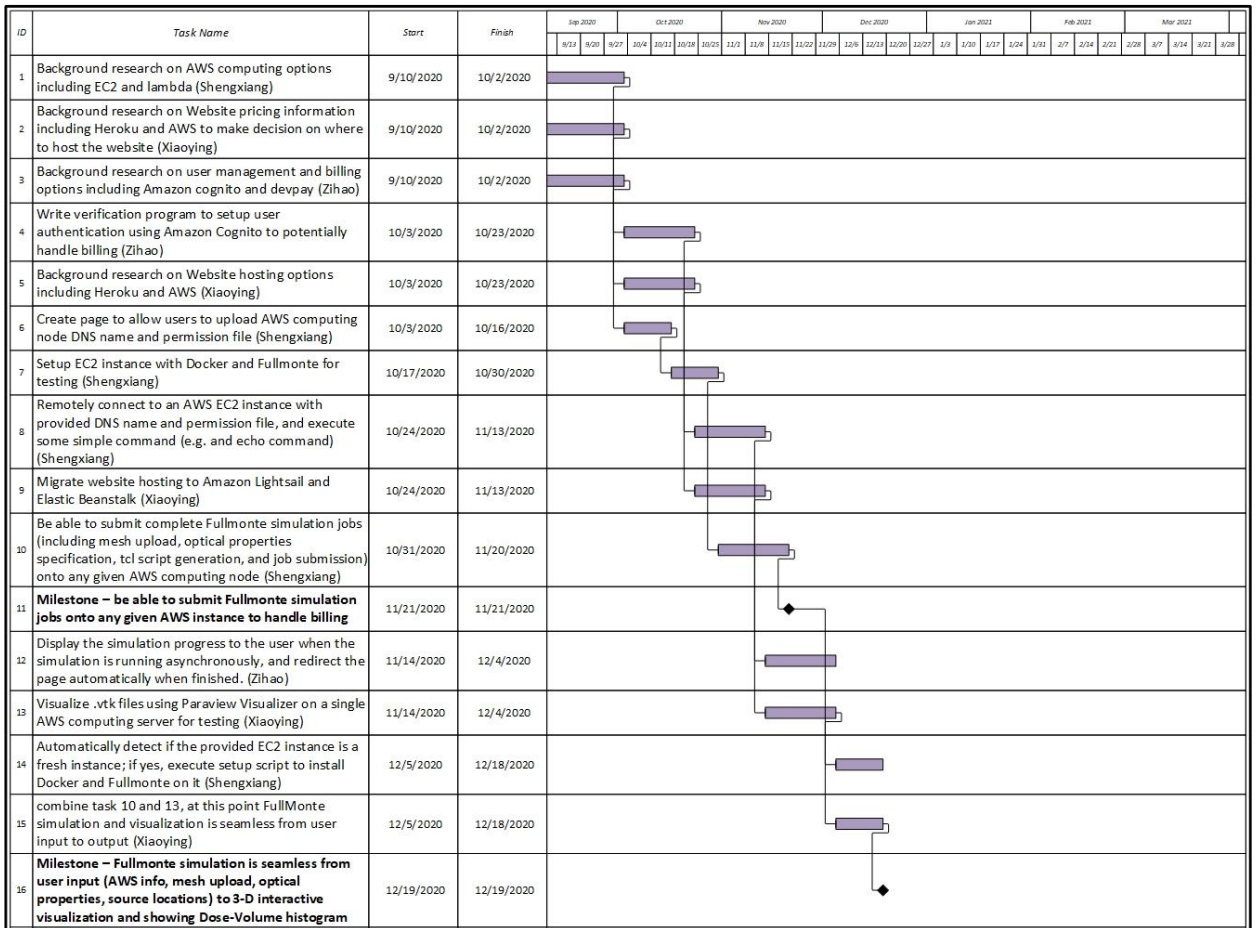
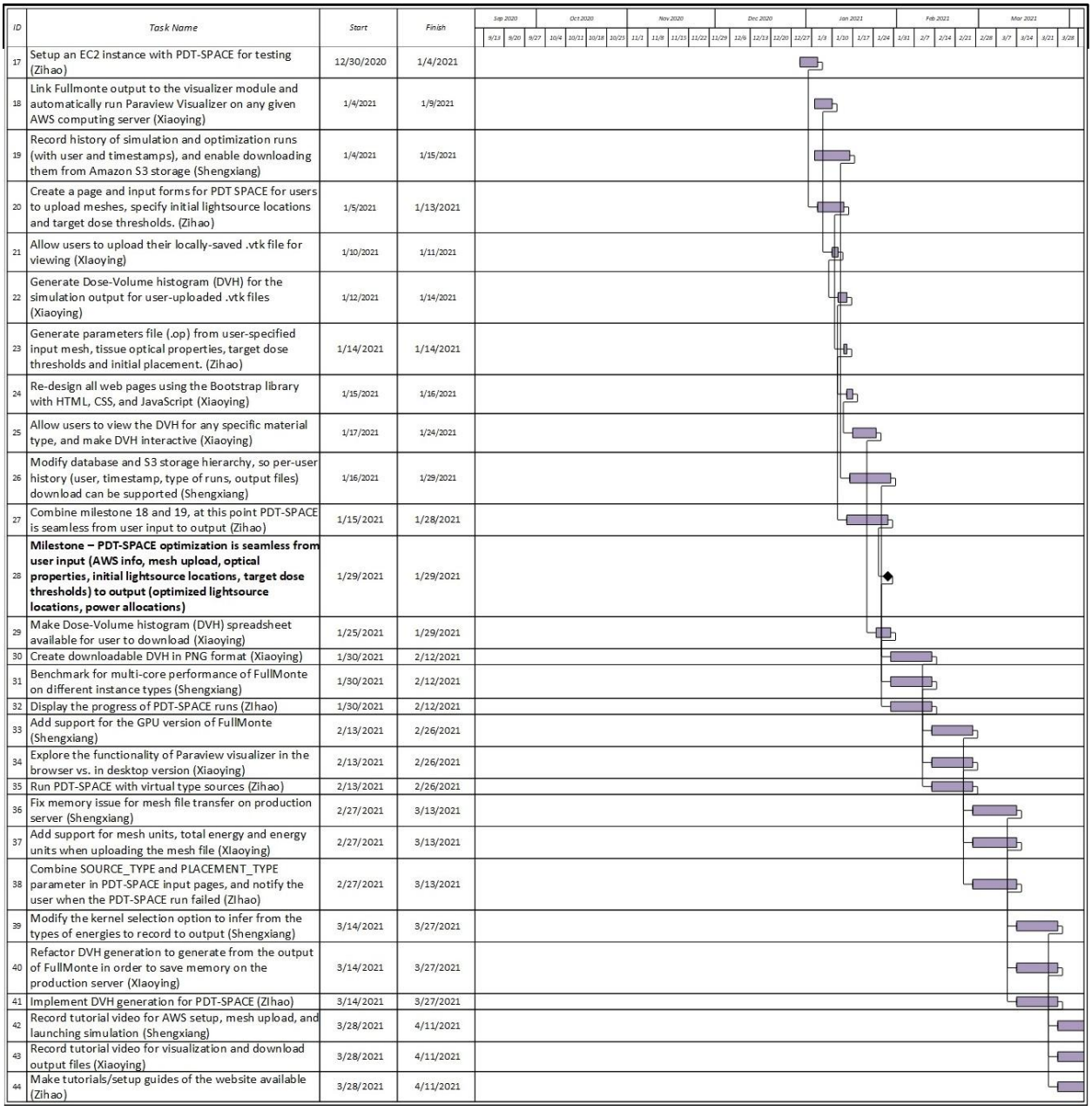


Figure 11. Gantt chart in Progress Report, second half.



ID	Task Name	Start	Finish
1	Background research on AWS computing options including EC2 and lambda (Shengxiang)	9/10/2020	10/2/2020
2	Background research on Website pricing information including Heroku and AWS to make decision on where to host the website (Xiaoying)	9/10/2020	10/2/2020
3	Background research on user management and billing options including Amazon cognito and devpay (Zihao)	9/10/2020	10/2/2020
4	Write verification program to setup user authentication using Amazon Cognito to potentially handle billing (Zihao)	10/3/2020	10/23/2020
5	Background research on Website hosting options including Heroku and AWS (Xiaoying)	10/3/2020	10/23/2020
6	Create page to allow users to upload AWS computing node DNS name and permission file (Shengxiang)	10/3/2020	10/16/2020
7	Setup EC2 instance with Docker and Fullmonte for testing (Shengxiang)	10/17/2020	10/30/2020
8	Remotely connect to an AWS EC2 instance with provided DNS name and permission file, and execute some simple command (e.g. and echo command) (Shengxiang)	10/24/2020	11/13/2020
9	Migrate website hosting to Amazon Lightsail and Elastic Beanstalk (Xiaoying)	10/24/2020	11/13/2020
10	Be able to submit complete Fullmonte simulation jobs (including mesh upload, optical properties specification, tcl script generation, and job submission) onto any given AWS computing node (Shengxiang)	10/31/2020	11/20/2020
11	**Milestone – be able to submit Fullmonte simulation jobs onto any given AWS instance to handle billing**	11/21/2020	11/21/2020
12	Display the simulation progress to the user when the simulation is running asynchronously, and redirect the page automatically when finished. (Zihao)	11/14/2020	12/4/2020
13	Visualize .vtk files using Paraview Visualizer on a single AWS computing server for testing (Xiaoying)	11/14/2020	12/4/2020
14	Automatically detect if the provided EC2 instance is a fresh instance; if yes, execute setup script to install Docker and Fullmonte on it (Shengxiang)	12/5/2020	12/18/2020
15	combine task 10 and 13, at this point FullMonte simulation and visualization is seamless from user input to output (Xiaoying)	12/5/2020	12/18/2020
16	**Milestone – Fullmonte simulation is seamless from user input (AWS info, mesh upload, optical properties, source locations) to 3-D interactive visualization and showing Dose-Volume histogram**	12/19/2020	12/19/2020

Figure 12. Final Gantt chart, first half.



ID	Task Name	Start	Finish
17	Setup an EC2 instance with PDT-SPACE for testing (Zihao)	12/30/2020	1/4/2021
18	Link Fullmonte output to the visualizer module and automatically run Paraview Visualizer on any given AWS computing server (Xiaoying)	1/4/2021	1/9/2021
19	Record history of simulation and optimization runs (with user and timestamps), and enable downloading them from Amazon S3 storage (Shengxiang)	1/4/2021	1/15/2021
20	Create a page and input forms for PDT SPACE for users to upload meshes, specify initial lightsource locations and target dose thresholds. (Zihao)	1/5/2021	1/13/2021
21	Allow users to upload their locally-saved .vtk file for viewing (Xiaoying)	1/10/2021	1/11/2021
22	Generate Dose-Volume histogram (DVH) for the simulation output for user-uploaded .vtk files (Xiaoying)	1/12/2021	1/14/2021
23	Generate parameters file (.op) from user-specified input mesh, tissue optical properties, target dose thresholds and initial placement. (Zihao)	1/14/2021	1/14/2021
24	Re-design all web pages using the Bootstrap library with HTML, CSS, and JavaScript (Xiaoying)	1/15/2021	1/16/2021
25	Allow users to view the DVH for any specific material type, and make DVH interactive (Xiaoying)	1/17/2021	1/24/2021
26	Modify database and S3 storage hierarchy, so per-user history (user, timestamp, type of runs, output files) download can be supported (Shengxiang)	1/16/2021	1/29/2021
27	Combine milestone 18 and 19, at this point PDT-SPACE is seamless from user input to output (Zihao)	1/15/2021	1/28/2021
28	**Milestone – PDT-SPACE optimization is seamless from user input (AWS info, mesh upload, optical properties, initial lightsource locations, target dose thresholds) to output (optimized lightsource locations, power allocations)**	1/29/2021	1/29/2021
29	Make Dose-Volume histogram (DVH) spreadsheet available for user to download (Xiaoying)	1/25/2021	1/29/2021
30	Create downloadable DVH in PNG format (Xiaoying)	1/30/2021	2/12/2021
31	Benchmark for multi-core performance of FullMonte on different instance types (Shengxiang)	1/30/2021	2/12/2021
32	Display the progress of PDT-SPACE runs (ZIhao)	1/30/2021	2/12/2021
33	Add support for the GPU version of FullMonte (Shengxiang)	2/13/2021	2/26/2021
34	Explore the functionality of Paraview visualizer in the browser vs. in desktop version (Xiaoying)	2/13/2021	2/26/2021
35	Run PDT-SPACE with virtual type sources (Zihao)	2/13/2021	2/26/2021
36	Fix memory issue for mesh file transfer on production server (Shengxiang)	2/27/2021	3/13/2021
37	Add support for mesh units, total energy and energy units when uploading the mesh file (XIaoying)	2/27/2021	3/13/2021
38	Combine SOURCE_TYPE and PLACEMENT_TYPE parameter in PDT-SPACE input pages, and notify the user when the PDT-SPACE run failed (ZIhao)	2/27/2021	3/13/2021
39	Modify the kernel selection option to infer from the types of energies to record to output (Shengxiang)	3/14/2021	3/27/2021
40	Refactor DVH generation to generate from the output of FullMonte in order to save memory on the production server (XIaoying)	3/14/2021	3/27/2021
41	Implement DVH generation for PDT-SPACE (ZIhao)	3/14/2021	3/27/2021
42	Record tutorial video for AWS setup, mesh upload, and launching simulation (Shengxiang)	3/28/2021	4/11/2021
43	Record tutorial video for visualization and download output files (Xiaoying)	3/28/2021	4/11/2021
44	Make tutorials/setup guides of the website available (Zihao)	3/28/2021	4/11/2021

Figure 13. Final Gantt chart, second half.

Appendix B: Financial Plan (Author: X. Dai)

The project does not require funding, as shown in the budget table in Figure 14 below.

Computing Services									
Item	**Priority**	**Standard Cost (USD)/Unit**	**Actual Cost (USD)/Unit**	**Quantity**	**Total Cost (USD)**	**Total Cost (CAD)**	**Requires Funding**	**Kept/Paid for by Students**	**Cost Reference**
Amazon EC2 instance (t3.micro)	1	$0.0104/h	$0	8 mo	$0	$0	N	Y	(1)
Heroku web hosting server	1	$25/mo	$0	8 mo	$0	$0	N	Y	(2)
Heroku postgreSQL database management system	1	$50/mo	$0	8 mo	$0	$0	N	Y	(2)
Amazon S3 storage	1	$0.023 /GB mo	$0.023 /GB mo	5GB/mo, 2 mo	$0.23	$0.29	N	Y	(1)
Mosek license for PDT-SPACE	1	$1950 + $487.5/mo	$0	8 mo	$0	$0	N	Y	(3)
AWS GPU instance (Tesla T4)	2	$0.52/h	$0.52/hr	10 h	$5.20	$6.56	N	Y	(1)
Amazon Lightsail instance	3	$5/mo (fixed)	$5/mo	5 h	$5	$6.31	N	Y	(1)
Amazon Elastic Beanstalk service	3	$0.0084/h	$0	5 h	$0	$0	N	Y	(1)
Amazon prostgreSQL database management system	3	$30/mo	$30/mo	0 h	$0	$0	N	Y	(1)
Total Computing Services					**$10.43**	**$13.16**			
Total Required Funding					**$0.00**	**$0.00**			

Student Labour				
Member	**Salary (CAD)/ Member**	**Quantity (# of hours)**	**Total Salary (CAD)**	**Salary Reference**
Zihao (Samuel) Chen	$20.65	480	$9,912.00	(4)
Xiao Ying (Helen) Dai	$20.65	480	$9,912.00	(4)
Shengxiang (Bob) Ji	$20.65	480	$9,912.00	(4)
Total student labour (unfunded)			**$29,736.00**	

Funding	
Students ($100 ea)	$300.00
Supervisor	$0.00
Request from Design Center	$0.00
Total funding	**$300.00**

Total Cost of Project (CAD)	**$29,749.16**
Total Cost Requiring Funding (CAD)	**$0.00**

Figure 14. Budget table for project.

Priority 1) Currently used services required to achieve functional requirements.

The standard cost of Amazon Web Services products are shown in the "Standard Cost (USD)/Unit" column, but Amazon offers a small EC2 instance, S3 storage, and PostgreSQL database with a one-year free trial, so the actual costs for the team is $0 for these services. However, we have exceeded the Amazon S3 storage's free-tier limit, so there was some overhead for the last two months of development. Heroku Web Hosting also offers a small 250 MB server as a free-tier option, and we can use it as long as we want. For future reference, we have also documented the standard cost of Heroku services in case we need to scale up the website. A Mosek license is required for running PDT-SPACE, but the academic license is free with an university email.

Priority 2) Currently used service for achieving optional objectives.

One of our nice-to-have features was to try launching simulations on an AWS GPU core, so that resulted in some expenses.

Priority 3) Services tested as alternatives during research.

We have tried some alternative web hosting services provided by Amazon. Amazon Lightsail offers a standard plan at $5USD/month. Even Though we only used the service for 5 hours, we had to pay the fixed month price in whole. Amazon Elastic Beanstalk allows users to select the EC2 instance for web hosting, so the only cost comes from the EC2 instance. In addition, a PostgreSQL database is required for both options, which comes with an additional charge.

The reference for the list of costs in the table are as follows:

(1) Amazon Web Services pricing page
https://aws.amazon.com/pricing/

(2) Heroku pricing page
https://www.heroku.com/pricing

(3) Mosek website pricing page
https://www.mosek.com/sales/commercial-pricing/

(4) PayScale average web developer salary in Canada
https://www.payscale.com/research/CA/Job=Web_Developer/Salary

Appendix C: Verification for F1 (Author: Z. Chen)

For available meshes in Fullmonte repository(Table 4), we will upload the mesh and specify corresponding parameters on our web interface, and then launch the simulation. We then check whether the simulation is running on AWS instance. Our design passes this test if the simulation is running on AWS instance after we launched the simulation on web interface.

Table 6: Test results for project requirement F1

Mesh Name	Test results: whether successfully launched
Colin27_tagged_tumor1-10	Yes
BonePDT	Yes
Bladder	Yes
HeadNeck	Yes
HeadNeck_Tumor	Yes
Mouse	Yes

Appendix D: Verification for F2 (Author: S. Ji)

As shown in the figure below, the details of finished simulations will be recorded, and the website will show the simulation history on a per-user basis. All the available files in the simulation history are hyperlinked to their storage location in Amazon S3, and will be downloaded to the user's local machine when clicked.

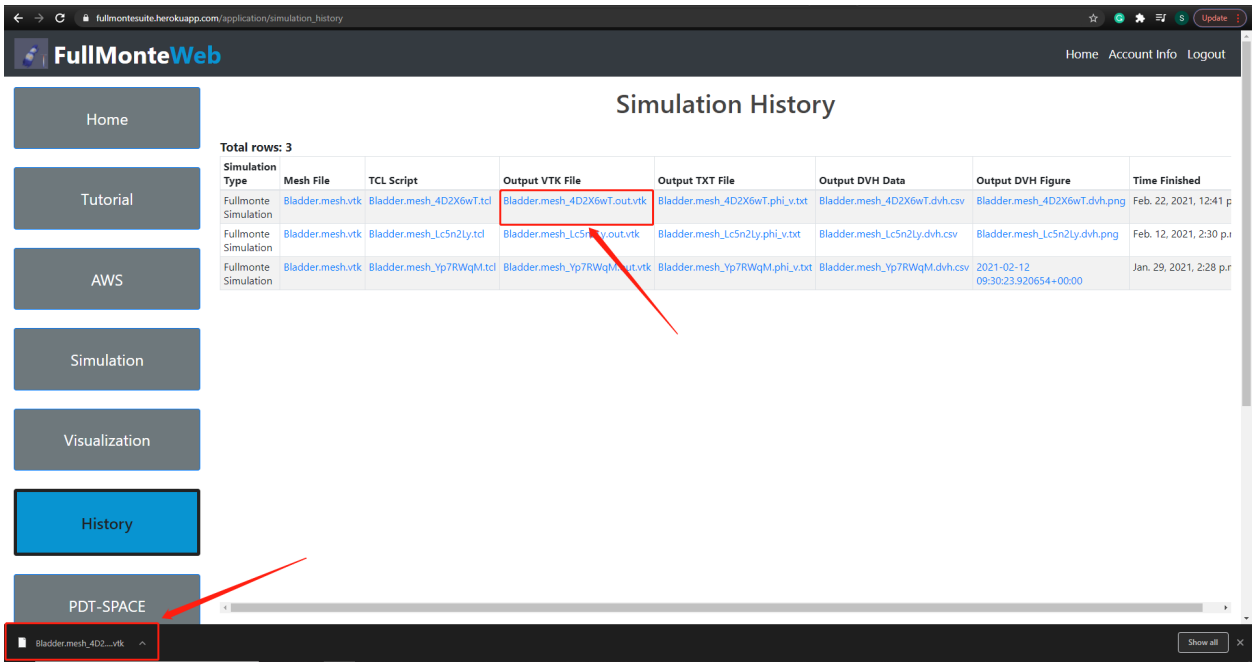


Figure 15: Downloadable simulation history details

Appendix E: Verification for F6 (Author: Z. Chen)

The first step is to generate the correct results from PDT-SPACE and we can use these correct results to determine the correctness of our design. To do this, we manually launched PDT-SPACE optimization on AWS instances for all available PDT-SPACE input mesh(Table 5) and saved the optimization results into log files. We then launched PDT-SPACE optimization for each available input mesh on our web interface and, for each input mesh, we compared the results on our web page with the correct results generated previously. Our design is considered to have the correct optimization result if the values of power allocation and v100 match the values of correct results. One more test is necessary to test whether our design is able to handle the situation if the invalid parameters are given. To test this, input invalid parameters and launch the PDT-SPACE optimization, and see whether our design is able to detect and report the error.

Based on the test results(Table 5), we can conclude that our design successfully accomplished the F6 project requirement.

Table 7: Test results for project requirement F6

Input mesh name	Test results
Colin27_tagged_tumor_1	Correct results
Colin27_tagged_tumor_2	Correct results
Colin27_tagged_tumor_3	Correct results
Colin27_tagged_tumor_4	Correct results
Colin27_tagged_tumor_5	Correct results
Colin27_tagged_tumor_6	Correct results
Colin27_tagged_tumor_7	Correct results
Colin27_tagged_tumor_8	Correct results
Colin27_tagged_tumor_9	Correct results

Colin27_tagged_tumor_10	Correct results
BonePDT	Correct results
Invalid input parameters	Detect and report the error

Appendix F: Verification for F7 (Author: Z. Chen)

After the simulation or optimization launched, our web page will display the running status and progress and update it every five seconds. To verify the status and progress is correct, we manually check the running progress and status by reading the progress percentage number in the log file and CPU time used by the simulation or optimization. Our design passes this test if the numbers on the website match the number I manually checked, and redirects to the corresponding page when the process finished. As a result, for simulation and optimization, the running status and progress are identical with the value in the log file, and the web page is able to redirect when the process is completed. Therefore, our design passes the verification test for this requirement.

Appendix G: Verification for C1 (Author: X. Dai)

In theory, our website should allow multiple users to use the website functionalities at the same time if they are using different accounts and different EC2 remote computing instances. This is because all extensive computations are completed on the user specified EC2 instance, including simulation, optimization, and 3D visualization jobs. To test it out, we first did internal testing following the procedures outlined below:

1. All three team members login to each of their own FullMonteWeb accounts.
2. Each member set up AWS using a different EC2 remote computing instance.
3. Launch FullMonte simulation at the same time.
4. Launch PDT-SPACE optimization at the same time.
5. Launch visualization at the same time
    a. Zoom in on the dose-volume histogram, toggle histogram data points on and off, and toggle different histogram region data on and off.
    b. Click on the “Open 3D Interactive Visualizer” button and try to zoom in, rotate, and set filters.

All three team members were able to use the website smoothly without any issues when we use different accounts and set up different EC2 instances. We then took the test further by using the website simultaneously with three other users, one Ph.D. student in medical biophysics and two high school students, All six users were able to use the website without any issues and did not notice any significant degradation in functional performance. Therefore, our web application passed the tests and meets this constraint.

Appendix H: Verification for C2 (Author: Z. Chen)

To test our design is able to work properly on popular browsers, run the F3 verification test using browsers like Google Chrome, Mozilla Firefox, Safari and Microsoft Edge. Based on the results (Table 6), we can conclude that our design is able to work properly on the aforementioned browsers.

Table 8: Test results for project requirement C2

Browser Name	Pass the F3 verification test or not
Google Chrome	Yes
Mozilla Firefox	Yes
Safari	Yes
Microsoft Edge	Yes

Appendix I: Verification for O2 (Author: Z. Chen)

To measure the redirect time for each web page, we use the performance analyzer in the developer tool of Google Chrome browser. We went through workflows for simulation and optimization three times, and then calculated the average redirect time for each webpage. If the average redirect time is less than 800 ms for all web pages, our design meets this objective. The Table 7 shows the average redirect time for all web pages. Therefore, our design does not pass this test since the redirect time for launching simulation and display visualization is greater than 800ms.

Table 9: Redirect time for web pages

application/aws: 553 ms	application/AWSsetup: 502 ms
application/aws_complete: 461 ms	application/simulator: 250 ms
application/simulator_material: 483 ms	application/simulator_source: 441 ms
application/simulation_confirmation: 470 ms	application/running: 1856 ms
application/simulation_finish: 386 ms	application/runningDVH: 451 ms
application/displayVisualization: 974 ms	application/simulation_history: 452 ms
application/pdt_spcae_wait: 489 ms	application/pdt_space_material: 310 ms
application/pdt_spcae_license: 427 ms	application/pdt_space_lightsource: 386 ms
application/pdt_space_running: 656 ms	application/pdt_space_finish: 490 ms
application/pdt_visualization_wait: 474 ms	application/pdt_space_display_visualization: 840 ms
application/mesh_upload: 531 ms	

Appendix J: Verification for OF1 (Author: X. Dai)

During input parameter setup for FullMonte simulations, the user must specify a light source position for the simulation. The original plan was that users should be able to visualize the mesh in 3D and the website should be able to capture the user's mouse click in real-time for light sources positions. However, upon investigations into the problem, we discovered that there are currently no available tools or browser application libraries to achieve real-time mouse position capturing for the VTK file format. Instead, we are only supporting the first half of this objective, which is providing 3D visualization of the input mesh so that users can turn on Axes Grid to examine the mesh file and determine the desired coordinates for light source positions. Several mesh files were tested, whereby we tried clicking on the 3D visualization button on the light source input page to view the mesh. Results are recorded below

Table 10: Testing for input mesh visualization

File Name	Visualize in 3D	Turn on Axis Grid to view coordinates	Capture user mouse click for coordinates	Result
Bladder.mesh	✓	✓	×	Fail
HeadNeck.mesh	✓	✓	×	Fail
HeadNeck_Tumor.mesh	✓	✓	×	Fail
Colin27	✓	✓	×	Fail
Colin27_tagged_tumor_1	✓	✓	×	Fail
Colin27_tagged_tumor_2	✓	✓	×	Fail
Colin27_tagged_tumor_3	✓	✓	×	Fail
Colin27_tagged_tumor_4	✓	✓	×	Fail

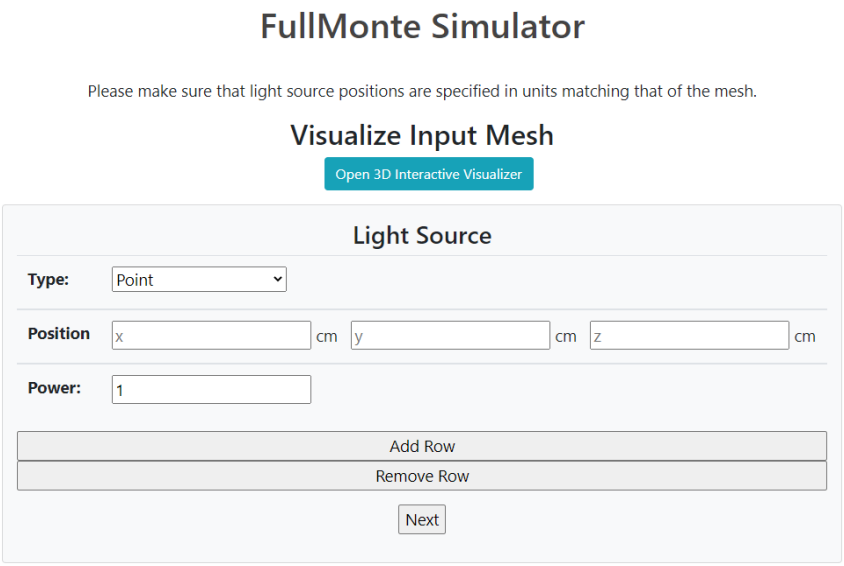


Figure 16. Light source setup page for FullMonte simulation with a button to visualize the mesh.

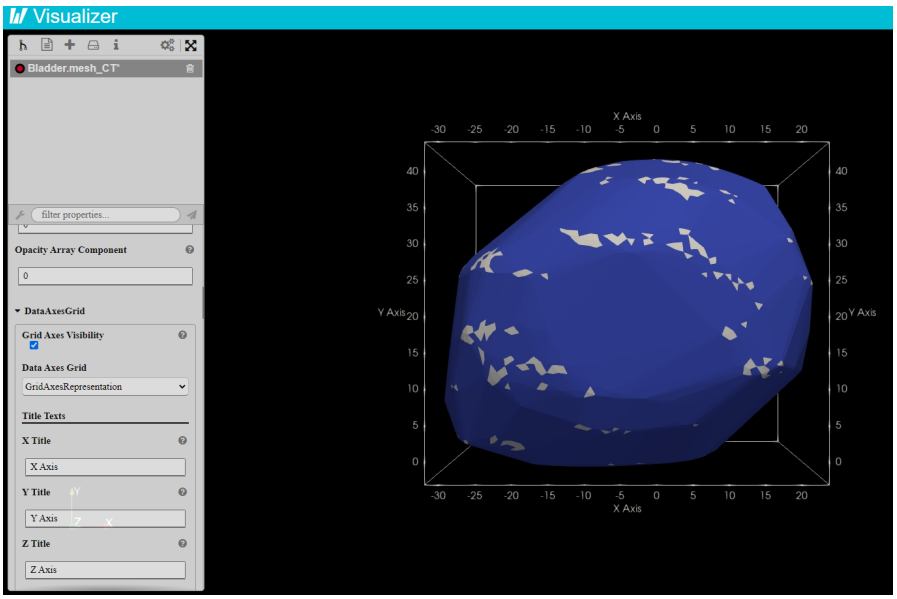


Figure 17. 3D representation of the input mesh with Axis Grid turned on to view coordinates.

Appendix K: Verification for OF3 (Author: S. Ji)

When the FullMonte simulator is running in the GPU mode (using TetraCUDA*Kernel specified in the TCL script), the “nvidia-smi” command is used to check the status of the GPU. As shown in the figure below, the GPU utilization is 100%, and a process from the FullMonte simulator is running on the GPU.

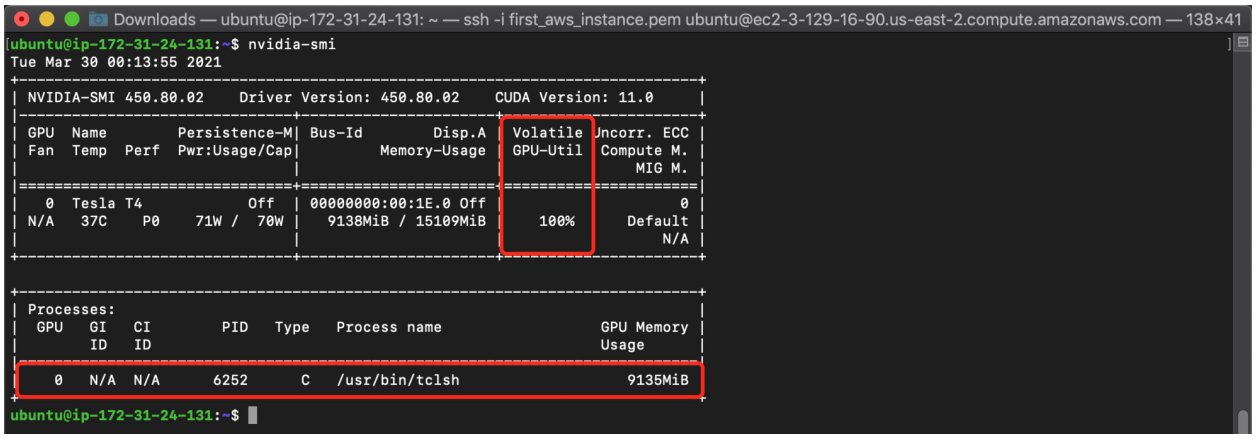


Figure 18. GPU usage for FullMonte in GPU mode.

Appendix L: Table of Links (Author: X. Dai)

The following table documents all links relevant to the project development.

Item	URL
FullMonteWeb application	http://fullmontesuite.herokuapp.com/
Github repository	https://github.com/jishengx97/FullMonteWeb
FullMonteWeb Heroku dashboard	https://dashboard.heroku.com/apps/fullmontesuite
AWS Management Console	https://aws.amazon.com/console/

Appendix M: Investigations for Website Hosting Options (Author: X. Dai)

Elastic Beanstalk

Features:

- direct deployment from local repository to the cloud
- provides a default CNAME like <name of your choice>.ca-central-1.elasticbeanstalk.com
- Elastic Beanstalk was created to compete with Heroku

Following these procedures, everything works except for the VTK library:

Role info:
Access Key ID: <your access key ID>
Secret Key: <your secret key>
IAM User ARN: <your ARN>
Tutorial: https://realpython.com/deploying-a-django-app-and-postgresql-to-aws-elastic-beanstalk/
Note: delete Django-heroku from requirements.txt
delete all references to heroku from settings.py
Note: remember to always commit your changes before calling "create" and "deploy"
Note: vtkOpenGLKitPython package results in a ModuleNotFoundError when the website is deployed. Turns out it needs a libgl1-mesa-glx deb package, which the Amazon Linux AMI does not have. There is no intuitive way to configure this on the server though.
1. Set up python 3.6 virtual environment:
virtualenv ~/Capstone
source ~/Capstone/bin/activate
2. Follow tutorial starting section "CLI for AWS Elastic Beanstalk"
> eb init parameters:
region: 15) Canada-Central
Access Key ID: <your access key ID>
Secret Key: <your secret key>
create an application called FullMonteWeb
select python 3.6 platform
skip CodeCommit
skip set up SSH instance
> eb status
used to check status (same as what's shown in console) and to see the default CNAME given by Elastic Beanstalk
copy this CNAME and add it to ALLOWED_HOSTS in settings.py
3. On AWS, open Elastic Beanstalk. Go to your Application > Environment > Configuration > Software Configuration.
Under Environment Properties you will find a list of properties you can configure. Set the environment variables (e.g. SECRET_KEY)

Drawbacks

- Only supports the Amazon Linux AMI, which does not work well with VTK packages that need to install libgl1-mesa-glx: https://github.com/Azure/azure-functions-python-worker/issues/543

- Unlike Heroku, there is no free tier server (for web hosting) or PostgresSQL management (used for our user authentication model)

Workaround

- Create a custom AMI, which is now supported by EB, and apparently you can customize it to use Ubuntu: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/custom-platforms.html?icmpid=docs_elasticbeanstalk_console
- Not much documentation on how to do this correctly
- Looking at last year's code, it seems like the VTK package was only used to display the DVH, not for interactive visualization, so if we find a different way to display the DVH, we might not have to import this package after all

Pricing

- There is no additional charge for AWS Elastic Beanstalk. You pay for AWS resources (e.g. EC2 instances or S3 buckets) you create to store and run your application. You only pay for what you use, as you use it; there are no minimum fees and no upfront commitments.
- Pricing options for the EC2 instance are as follows:

  On Demand - pay by the hour

	vCPU	Memory (GiB)	Price (USD/h)
a1.medium	1	2	0.0255
t4g.micro	2	1	0.0084
t4g.small	2	2	0.0168
t3.micro	2	1	0.0104
t2.micro	1	1	0.0116

  Spot Instance - request spare Amazon EC2 computing capacity (up to 90% off)

	vCPU	Memory (GiB)	Price (USD/h)
a1.medium	1	2	0.0049
t4g.micro	2	1	N/A

t4g.small	2	2	N/A
t3.micro	2	1	0.0031
t2.micro	1	1	0.0035

Reserved Instance - annual plans

- For a 1-year plan, you can get 40% discount off standard on-demand pricing.
- For a 3-year plan, you can get 60% discount off standard on-demand pricing.

Amazon Lightsail

Features

- Can pipe changes directly from local repository to Lightsail using CodeDeploy and CodePipeline, although it is easier to upload the code or git clone to the AWS VM and deploy from there
- Provides up to 5 static IP addresses, but no default CNAME
- It is convenient if you want to make a Wordpress website in Lightsail, but if we are making a Django website, we basically just make an operating system instance (an Ubuntu virtual server in our case), setup an external port, associate it with a static IP address in the Lightsail Console, and run manage.py to deploy to the port.
- Note: there are some complications with using mod-wsgi for Python 3.7. For a site with Python 3.6, we could easily run apt install libapache2-mod-wsgi-py3. However, there is currently no such easy package for 3.7 (tried a couple of methods online and nothing worked properly), so we are downgrading the project to Python 3.6

Two ways to copy project over to AWS:

1. use CodeDeploy and CodePiplining to pipeline any changes from git to the server:

Tutorial:
https://aws.amazon.com/blogs/compute/using-aws-codedeploy-and-aws-codepipeline-to-deploy-applications-to-amazon-lightsail/
Tried using this and it is extremely complicated. We need to keep track of 3 types of AWS services, some accounts like IAM user and IAM role, as well as policies for them. We also need to write an app specific configuration file called appspec.yml, and some startup scripts.

Note: do not use Amazon Linux, use Ubuntu 18.04. Also, for some reason the startup script would not execute, so after opening the terminal session,
go into /etc/codedeploy-agent/conf and manually type in the following:

```
sudo wget https://aws-codedeploy-us-west-2.s3.us-west-2.amazonaws.com/latest/install
sudo chmod +x ./install
sudo ./install auto
```

Note: before the last step, may also need to install ruby if it does not already exist:
sudo apt install ruby
Note: before the last step, may also need to install gdebi
sudo add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc) universe"
sudo apt-get update
sudo apt-get install gdebi

2. A simpler way is to have a local git repo on the server and just pull changes once in a while. The end product is exactly the same.

Tutorial: https://silverspringenergy.com/deploying-an-existing-django-application-to-aws-ubuntu-with-apache/
See above link starting in section "Configuring a virtual environment". Because we are now running with Python 3.6, we need to use virtualenv.
Note: in the environment, before installing everything from required.txt, you may have to install these first:
sudo apt-get install python-psycopg2
sudo apt-get install libpq-dev
Note: if you run "manage.py runserver" and close the terminal session before typing Ctrl+C, the next time when you reopen it, you will not be able to stop the server
If you need to stop it for some reason, type:
ps auxw | grep runserver (This will return the process and its respective PID, which is the second number, then do)
kill <pid>

Drawbacks

- Unlike Heroku, there is no free tier server or database management
- Unlike Elastic Beanstalk, Lightsail only offers at a fixed monthly costs for the virtual server and database

Pricing options for the EC2 instance are as follows:

	vCPU	Memory (GB)	Price (USD/mo)
Option 1	1	0.512	3.5
Option 2	1	1	5
Option 3	2	2	10
Option 4	2	4	20

Amazon PostgreSQL

Both Amazon web hosting options require the web developer to purchase PostgreSQL as a separate charge. The pricing options are as follows:

	Price (USD/hr)
db.t3.micro	0.018
db.t3.small	0.036
db.t3.medium	0.072
db.t3.large	0.145

Appendix N: Dose-Volume Histogram FLow (Author: X. Dai)

The dose-volume histogram for FullMonte simulation is generated using our own algorithm, which works as follows:

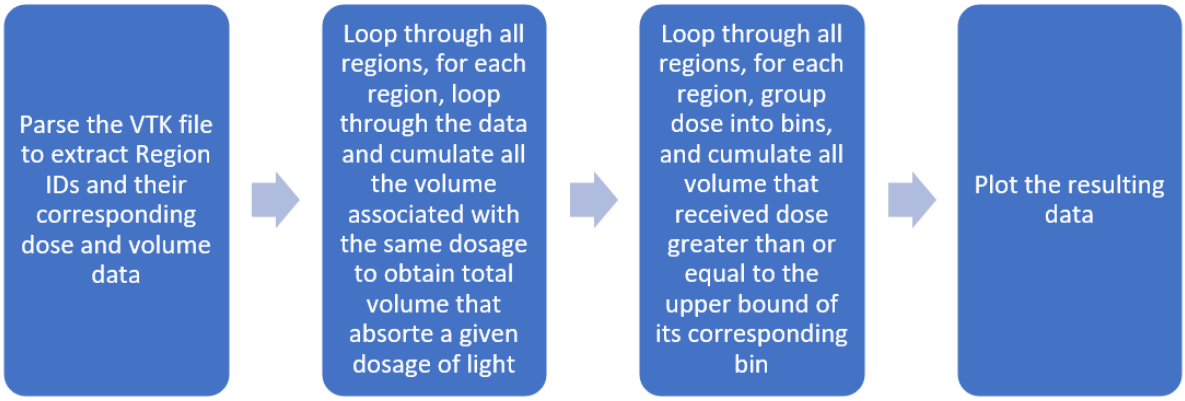


Figure 19. Flowchart outlining the DVH generation algorithm.

Appendix O: Investigations for Interactive 3D Visualization (Author: X. Dai)

The purpose is to find an existing 3D Visualization tool that works on the browser and supports the UnstructuredGrid format VTK files. An alternative is to find a browser library that can be used to implement a visualizer from scratch. Research was conducted and documented in this section.

Tools and packages from Kitware

Paraview	• an open-source multiple-platform application for interactive, scientific visualization • can be used with Paraview Desktop or various other web-based visualizer packages
ParaViewWeb	• A JavaScript Library for Building Web-based Applications with Scientific Visualization • Does not support rendering, so any applications that use this library must work with a ParaView server to do the processing/rendering • Kitware has designed many applications that leverages this Library; all can be found at: https://www.paraview.org/web/
VTK.js	• A rendering library made for Scientific Visualization on the Web; Its implementation consists of an ES6 JavaScript class library and leverages WebGL • applications that use this library do not need a ParaView server and can do rendering from the browser • Kitware has designed many sample applications that leverages this Library; all can be found at: https://kitware.github.io/vtk-js/docs/index.html under Examples

ParaViewWeb vs. VTK.js	• https://discourse.vtk.org/t/confusion-with-tool-selection-vtk-vs-paraview-for-web-based-volume-rendering-of-large-data-sets/1384

Kitware Applications Built with ParaViewWeb Library

ParaView Visualizer	• A client-server based application that uses the ParaViewWeb library to provide a ParaView-like experience inside the Web browser ○ client-server meaning it needs to work with a ParaView server • The ParaViewWeb library contains all the components needed to build the UI and the data access (I/O) routines to communicate to the ParaView server using WebSocket connectivity • Docs and tutorials: https://kitware.github.io/paraviewweb/docs/architecture.html#Simplicity
ParaView LightViz	• same as Visualizer but is supposedly more intuitive for the user since a lot of functionalities are bundled • Also client-server based
Arctic Viewer	• Unlike the previous two, this one does not require a rendering server (ParaView) - what we are ideally looking for • BUT it needs the data to be pre-processed, which is no good since we don't expect users to have pre-processed data
HPCCloud	• Cloud/Web-based simulation environment platform that uses Amazon EC2 instances • Setup tutorial: https://github.com/Kitware/HPCCloud-deploy • A paper on this: https://www.researchgate.net/publication/304407029_HPCCloud_A_CloudWeb-Based_Simulation_Environment

Kitware Applications Built with VTK.js Library

GeometryViewer	• Main page: https://kitware.github.io/vtk-js/examples/GeometryViewer.html • Last year's team reported that this does not support certain types of .vtk files • Looks like Kitware made some more changes to their supported file formats:

	https://www.earthmodels.org/software/vtk-and-paraview/vtk-file-formats ● .vtk is the old “Legacy format”, and they now have XML-based formats (extension: depends; e.g. *.vtp for polygonal data, *.vtu for unstructured grid) ● They recommend using .vtk for ParaView, and covert into the appropriate newer format when you want to use any of the VTK.js applications
Build Our Own Application	● We are only dealing with .vtu (UnstructuredGrid) type files, and none of the existing VTK.js applications (provided as examples) support this file format, so we thought maybe we could build up our own application and integrate it into our FullMonte web application ● Later discovered that this is impossible because the VTK.js library does not support the UnstructuredGrid format at all.

It turned out that the only option that works for our purposes is the client-server-based ParaView Visualizer.