

**FAMU/FSU College of Engineering
Department of Mechanical Engineering**



Final Report

Marine Keel Cooler Optimization Tool

EML 4551C Senior Design



Project Sponsor: Frank Ruggiero - Cummins Inc.

Faculty Advisor: Steven W. Van Sciver, Ph.D.

Submitted to: Nikhil Gupta, Ph.D.

Submission Date: 8 April 2016

Team 3

Melissa Allende ma10k@my.fsu.edu

Grady Beasley gob12@my.fsu.edu

Stanko Gutalj hjh11@my.fsu.edu

James Haga hjh11@my.fsu.edu

Jacob Ross jmr12@my.fsu.edu

Table of Contents

| | |
|--|-----|
| Table of Figures | iii |
| Table of Tables | iv |
| ABSTRACT | v |
| ACKNOWLEDGEMENTS | vi |
| 1. Introduction | 1 |
| 2. Project Definition | 4 |
| 2.1 Background Research..... | 4 |
| 2.2 Problem Statement..... | 9 |
| 2.5 Constraints..... | 12 |
| 3. Design and Analysis | 13 |
| 3.1 Software Design..... | 13 |
| 3.2 Hardware Design..... | 20 |
| 3.3 Functional Analysis..... | 23 |
| 3.4 Operation Instruction..... | 25 |
| 3.5 Troubleshooting..... | 27 |
| 3.6 Regular Maintenance..... | 27 |
| 4. Methodology | 28 |
| 4.1 Marine Keel Cooler Optimization Development..... | 28 |
| 4.2 Risk Analysis..... | 28 |
| 4.3 Designing for Manufacturing..... | 29 |
| 4.4 Design for Reliability..... | 30 |
| 4.5 FMEA Analysis..... | 31 |
| 4.6 Procurement..... | 31 |
| 4.7 Design for Economics..... | 32 |
| 4.8 Schedule..... | 33 |
| 4.9 Resource Allocation..... | 33 |
| 4.9.1 Work Breakdown..... | 33 |
| 5. Environmental and Safety Issues and Ethics | 37 |
| 6. Design and Experiment | 38 |
| 6.1 Experimental Procedure..... | 38 |
| 6.2 Results..... | 41 |
| 6.3 Impact to Industry..... | 43 |
| 7. Conclusion | 45 |
| References | 46 |
| Biography | 47 |
| Appendix A | 48 |

Table of Figures

| | |
|---|----|
| Figure 1. Simple Diagram of a Typical Keel Cooler System. | 1 |
| Figure 2. Minimum Keel Cooler Surface Area..... | 7 |
| Figure 3. LTA Inlet and Outlet Connections and Pressure Service Ports..... | 8 |
| Figure 4. JWAC and LTA cooling system loops..... | 8 |
| Figure 5. House of Quality..... | 11 |
| Figure 6. Current Specification Sheet..... | 14 |
| Figure 7. Flow Chart for the Keel Cooler Optimization Tool | 16 |
| Figure 8. Program File Structure | 18 |
| Figure 9. Marine Keel Cooler Optimization Tool Entry Form..... | 18 |
| Figure 10. Flow Chart of Program..... | 19 |
| Figure 11. Keel Cooler Testing Apparatus | 21 |
| Figure 12. Engine Selection View to User..... | 24 |
| Figure 13. First Information User Prompter to Enter | 25 |
| Figure 14. Example of Coolant Selection..... | 26 |
| Figure 15. Channel Material, Size and Number of Flow Paths Selection | 26 |
| Figure 16. Funds Graph | 32 |
| Figure 17. Gantt Chart | 35 |
| Figure 18. Experimental Procedure | 39 |
| Figure 19. Experimental Setup | 40 |
| Figure 20. Inlet and Outlet Water Temperatures for Multiple Flow Paths..... | 42 |
| Figure 21. Inlet and Outlet Water Temperatures for Single Flow Paths | 43 |
| Figure 22. Carolina Queen Ferry | 43 |
| Figure 23. A View of the Carolina Queen’s Hull with the Keel Cooler..... | 44 |

Table of Tables

Table 1. Decision Matrix 13
Table 2. Material Procurement 31
Table 3. Resource Allocation..... 36

Abstract

This report defines the project plans, product specifications and team methodology for the Marine Keel Cooler Optimization Tool. Cummins Marine is in need of a better tool which would enable the Marine Application Engineers to ensure proper validation of the marine keel cooler. The current tool was developed in the early 1980's and is limited to only steel keel coolers and only provides a pass/fail output to the user. The team is then faced with the creation of a new tool which will not only test the pass/fail cooling capability of the keel cooler but the tool will also be able to calculate box channel, half round and full pipe sections in steel or aluminum. It will evaluate an existing keel cooler system and be able to recommend other sizes which would optimize the cooling per vessel/engine installation. Such tool will allow the Marine Application Engineer to validate the keel cooler not only in extreme conditions but in different climates as well since most commercial vessels will navigate across international waters.

To ensure tool accuracy, research has been conducted to obtain adequate knowledge with regards to keel cooled systems and the design parameters needed to keep in mind. The following report includes an overview of the schedule being followed in order to complete the project. The overall plan, methodology and project approach decided upon by the team will ensure deliverables are met on time and an accurate product is delivered to the sponsor.

Acknowledgements

The team would like to acknowledge our advisor, Dr. Van Sciver for his insight and direction. The team has been given the opportunity to meet with Dr. Van Sciver weekly in order to talk about the science involved as well as the methodology the team should follow when approaching this project. He has provided the insight needed in order to ensure success in the project ventures. The team would also like to acknowledge the Senior Design instructor, Dr. Gupta who also has met with the team weekly to discuss schedule and make sure the team covers the project scope appropriately. The team would also like to extend an acknowledgement to Frank Ruggiero for the opportunity to work on a project for the Marine Application Engineering group as well for the sponsorship and technical support of this project.

1. Introduction

Modern ships and boats rely upon high-powered propulsion systems in order to successfully navigate through their respective environments. The delivered power of engines for typical commercial marine vessels ranges between 230-2700 hp (169-2013 kW)¹. In order for these vessels to function properly, heat must be dissipated effectively in order to achieve the optimal efficiency for sailing conditions.

There are two main types of cooling systems for marine engines; the first is known as a raw water system and the second is known as the freshwater (closed) cooling system. In a raw water system, surrounding water is drawn from the outside of the ship and is circulated through the engine block and then expelled from the exhaust as shown in *Figure 1*. This is hazardous to the system in both salt- and fresh-water application due to the corrosiveness of salt on the water pump impellers and the risk of foreign contaminants which could lead the system to foul. The

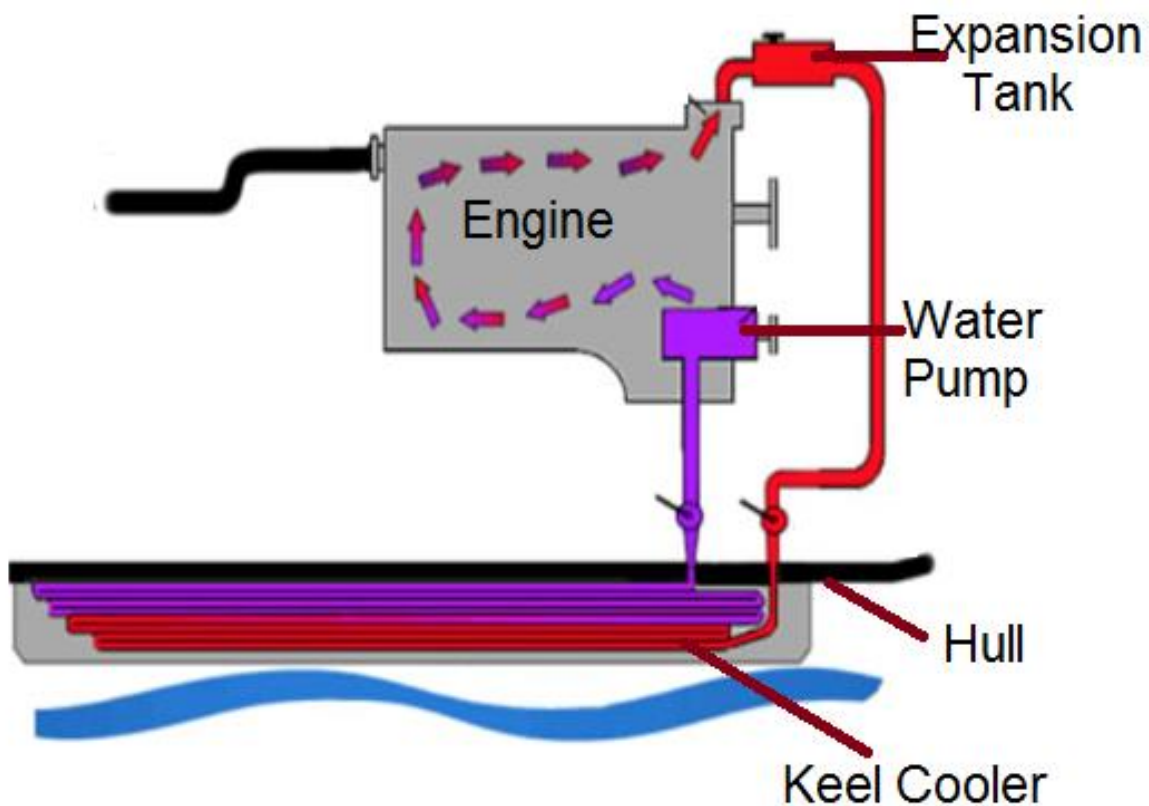


Figure 1. Simple diagram of a typical keel cool system.

engine components risk early failure and may lead to an engine overhaul before the vessel operator's expected to.

The second type of cooling system is known as a closed cooling system. These systems do not employ water as a direct cycling fluid rather, rather piping is used to separate the coolant and the surrounding medium. Some systems function similar to the radiator in automobiles where coolant is pumped through one side of a heat exchanger and raw water is pumped through another

Another type of closed cooling system removes the need for a heat exchanger by employing an external set of pipes which protrude from the bottom of the vessel to exchange heat directly with the surrounding water. Keel coolers operate by taking advantage of the surrounding water as an endless heat sink for a vessel's heat transfer fluid. Due to the risk of fouling from various contaminants contained in the water medium, these coolers typically do not run ocean water directly through the power cycle, but rather exchange heat via convection through external tubing between the engine coolant and the surrounding medium. This process is illustrated in Figure 1, explaining just how this process takes place. A pump draws coolant from the thermostat and sends it through an expansion valve, which sends coolant into the keel cooler at the bottom of the vessel. The heated fluid moves through a series of highly conductive pipes which remove the heat via convection with the heat sink. The cooled coolant is finally pumped back into the engine via a cooling pump. This process eliminates the need for a heat exchanger, and other components vital for closed cooling systems.

Cummins Marine is one of the different specialized markets of Cummins Inc. which specializes in the "Marinization" of engines and the design of new components to allow the current Cummins engine line survive in marine applications. Customers often times ask the Application Engineers to ensure the engine selected will work properly with the vessel it is going to be installed in. This includes the sanity check of ensuring that the keel cooler will provide the correct cooling for the engine. These factors are important to consider since the vessel must pass an Installation Quality Assurance Review in order to meet warranty. In order to meet customer's requirements, Cummins Marine makes use of a web-based optimization tool which allows the Application Engineers to predict whether or not a particular keel cooler design will successfully meet the vessels' requirements. The program operates on user-inputted parameters such as keel size, engine power, and temperature range². These values then predict whether the cooler will pass or fail based on extreme operating conditions. Although the tool has been in service for a long time, it has

several limitations. The tool only predicts keel cooler systems which are made from steel and does not offer an option to optimize the design. The program lacks feedback and is outdated as a user interface. The goal of the Keel Cooler Optimization Tool Senior Design Project is to create a tool that adds feedback alongside the pass-fail conditions. The program will suggest improvements in the design of the keel cooler based on a thermodynamic analysis. Such improvements can range from material selection, pipe configuration as well as an optimal temperature range of operating conditions. The successful implementation of this tool will result in an increase in company profit and customer satisfaction. With a program which successfully predicts improvements in keel cooler design, boat builders will be able to build the keel coolers with confidence knowing it will be more efficient and optimize the performance of their engine and work in different nautical waters.

To achieve this goal, extensive background research must be conducted on the variables important to the design of a keel cooler. Once the group has a full understanding of the analysis process, the method for creating this tool must be decided upon; this includes the programming language, program structure, user interface and a means for testing the accuracy of the program. The group must show sufficient understanding of the thermal design process and develop a product that is user-friendly, intuitive and provides meaningful feedback to the end user.

2. Project Definition

2.1 Background Research

A current model of this program exists and is used by Cummins. The program is used to evaluate the current and future keel coolers which will be installed in the vessels³. The current system of evaluating the keel coolers is done by looking at the engine which will be installed on the ship, the total heat generated by the engine, design speed of the vessel, maximum water temperature the vessel will face and currently only evaluates keel coolers made out of steel. This current program does not provide the user with suggestions on how to design a more efficient keel cooler and has been in commission since the early 1980's and is in dire need for an update.

Cummins Keel Cooler program was only developed to test keel coolers after production and would only determine as pass or fail given worst case scenarios. The new program will not only determine a pass or fail, it will suggest an optimal keel cooler size, design and material. The program underdevelopment is meant to be easily transferable and shared between users and eventually be converted to a web based program. The new program will most likely ask for the same input parameters as the previous program, but will be more accurate to ensure the keel coolers are properly sized and fitted for the vessels. The ease of evaluation needs to stay constant as well since the easier the program is to use, the more likely it will be utilized. The current design is web based, therefore the new design will need to be converted to a web based system once it is completed.

A keel cooler works as a radiator or heat exchanger attached to hull of the ship. One such textbook which will be referred to for future equations, charts, and tables on heat flow from a high temperature to a low temperature would be "Fundamentals of Thermal fluid Science's". For possibly making suggestions for better keel cooler designs the group would possibly need to suggest the addition of aluminum to the material selection. To do so, material properties would need to be known to allow proper suggestions. The group will possibly be referencing the materials book "Materials Science and Engineering an Introduction".

There is no opposition for this product due to it being geared towards Cummins engines to ensure that installed engines will work properly on the vessels without overheating. This program will be licensed by Cummins for its own use. The only program that would compete with this end product would be the current program the Marine Application group has been utilizing.

Keel cooling utilizes a group of tubes, pipes or channels attached to the outside of the hull below the waterline. Engine coolant is circulated through the keel cooler to remove excess heat. Fabricated keel coolers are manufactured by the boat builder as a part of the hull construction. Structural steel or aluminum shapes are usually used with 0.187 inch [4.8 mm] to 0.500 inch [12.7 mm] wall thickness. These materials must be compatible with materials used in the vessel's hull in order to prevent galvanic corrosion. Fabricated keel coolers must be designed oversized to allow for the decrease in effectiveness which occurs with the formation of rust, scale, pitting and marine growth on the keel cooler. Keel coolers can be sized given the following data from the Engine Data Sheet: Engine Model and Rating, heat rejection, engine coolant flow to keel coolers, coolant type, as well as the design speed of the vessel (in knots). Typical sizing speeds are 1-2 knots for tugboats/push boats and 0.1-1 knots for generator sets.

Fabricated keel coolers can be made from many different materials and type of construction. Most commonly used are steel channel and pipe, although this tool will also allow calculations for aluminum channel and pipes. The shape of the keel cooler is determined by the hull shape and size of the vessel. A fabricated keel cooler is not an efficient heat exchanger and therefore it is much larger in surface area than commercial keel coolers. Keel cooler length formulas for round pipe *Equation 1 (in feet) and 2 (in meters)*, and square channel *Equation 3 (in feet) and 4 (in meters)*. In order to calculate the required length for a half round pipe, one would use *Equation 1* and divide by 2. The "A" used in the formula is the keel cooler area coefficient. Utilizing these formulas the team will be able to calculate the cross sectional area required in order to size the keel cooler, which takes into consideration the channel design type being used, round pipe *Equation 5*, half round pipe *Equation 6* and for square channel *Equation 7*. This is dependent on the design speed of the vessel as well as the maximum raw water temperatures as shown in *Figure 2*. Flow path is also a critical part of the design, since the keel cooler can be sized smaller the more flow paths available. It is also important to note, the program will ask for the length of the vessel in order not to conduct unnecessary size recommendations.

Round Pipe:

$$\text{Length (ft)} = \frac{\text{Heat Rejection } \left(\frac{\text{BTU}}{\text{min}}\right) \times "A" \times 3.82}{\text{Pipe I.D. Diameter (in)}} \quad (1)$$

$$\text{Length (m)} = \frac{\text{Heat Rejection (kW)} \times "A" \times 1682}{\text{Pipe I.D. Diameter (mm)}} \quad (2)$$

Note: For half round piping, multiply the calculated length by two.

Square Channel:

$$\text{Length (ft)} = \frac{\text{Heat Rejection } \left(\frac{\text{BTU}}{\text{min}}\right) \times "A" \times 12}{\text{Width (in)} + [2 \times \text{Height (in)}]} \quad (3)$$

$$\text{Length (m)} = \frac{\text{Heat Rejection (kW)} \times "A" \times 5288}{\text{Width (mm)} + [2 \times \text{Height (mm)}]} \quad (4)$$

Round Pipe-Area:

$$\text{Round Pipe Cross Section Area} = \frac{[\text{Inside Diameter}]^2}{1.27} \quad (5)$$

Half Round Pipe-Area:

$$\text{Half Round Pipe Cross Section Area} = \frac{[\text{Inside Diameter}]^2}{2.54} \quad (6)$$

Square Channel-Area:

$$\text{Square Channel Cross Section Area} = \text{Inside Dimensions} \times \text{Width} \times \text{Height} \quad (7)$$

Note: Using **inch** dimensions gives **square inch** areas. Using **mm** dimensions gives **square mm** areas.

It is important to keep in mind the coolant velocity inside of the cooler. If the coolant flows through the keel cooler faster than 8 ft/sec [2.5 m/sec] the internal components will deteriorate, causing failure near manifold entrances and exits, elbows and other discontinuities in the coolant flow. Likewise, if the coolant flows through the keel coolers' passages too slowly rust particles or other particulate matter will settle out, choke off the flow and degrade the transfer of heat. In order to determine the proper flow pattern through the keel cooler, one needs to determine the minimum and maximum expected coolant flow through the keel cooler. This can be obtained from the performance data of the engine water pump. Calculating the difference between the maximum and minimum expected coolant flow and multiply the resultant by 2/3 and adding 2/3 will help determine the coolant flow and how to distribute the flow through the keel cooler passages. Then one would divide the coolant flow by the cross-sectional area of one keel-cooler passage to obtain the average velocity. If the average velocity through the keel cooler flow passages is greater than 8 ft/sec [2.5 m/sec], one would arrange the coolant flow in parallel so it would pass through two or more of the keel cooler passages per pass through the keel cooler. If the average velocity through the keel cooler flow passages is less than 2 ft/sec [0.6 m/sec], a keel cooler passage with a smaller cross section would be most adequate. The tool will focus on the three major data inputs for the engine in order to size/validate the keel cooler for the vessel, heat rejection, flow and change in temperature across the pressure ports. There are two 1-1/2 NPT threaded connection provided for

the installer to connect to the keel cooler circuit. *Figure 3* shows the inlet and outlet connection points to which the engine connects to the keel cooler via hoses. The pressure service ports is where Marine Application Engineers collect the data readings for pressure testing as well as collect the change in temperature. There are different types of keel cooling layouts, for example the Jacket Water After Cooler (JWAC) and Low Temperature After Cooler (LTA) *Figure 4*.

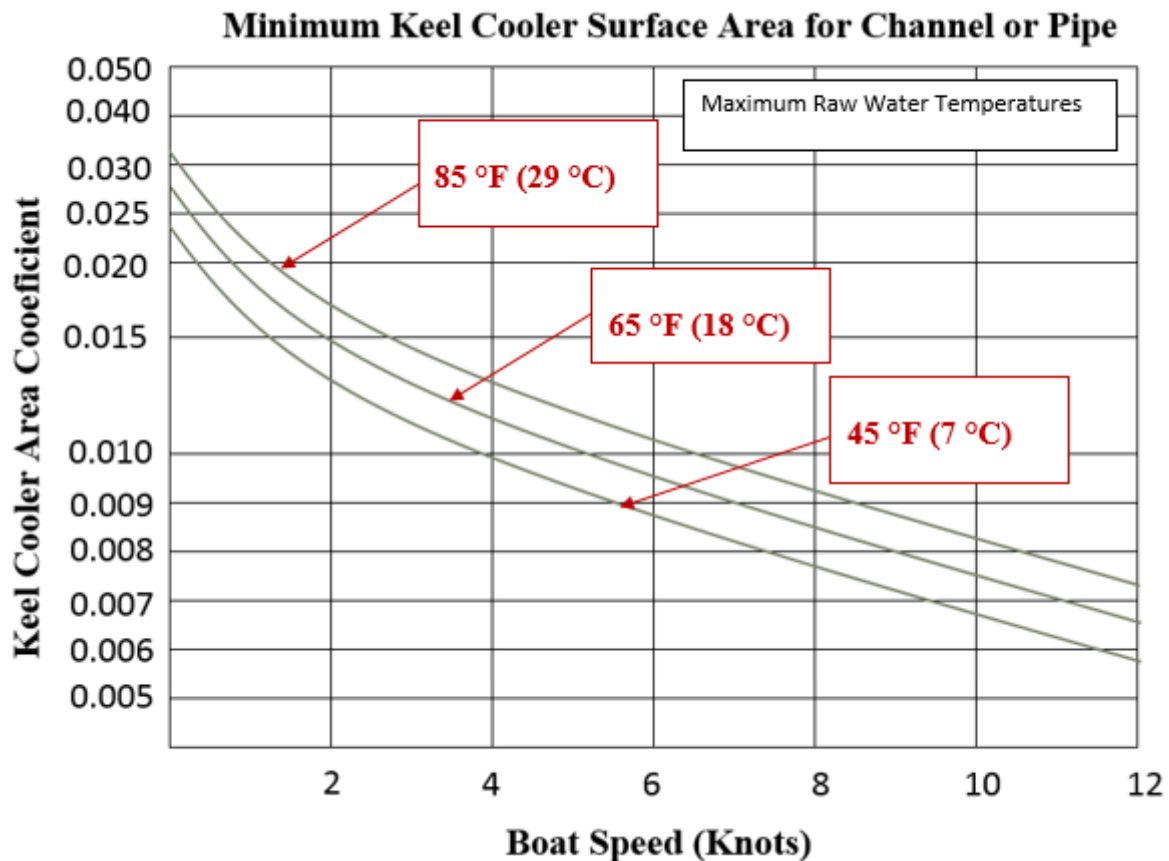


Figure 2: Minimum Keel Cooler Surface Area for Channel or Pipe.

The program will ask the user which type is being utilized in the vessel, such detail is important since the flow of the coolant is affected by the position of the thermostat in each system. For example, in an LTA system the thermostat is before the keel cooler. When the engine coolant is cold, the thermostat is closed and all coolant flows directly to the after cooler and is by-passing the keel cooler. When the thermostat begins to open (depending on the engine the opening temperature will vary) coolant is directed through the keel cooler and is returned back to the

thermostat housing where it is mixed with the main flow going through the after cooler. In an installation which is JWAC, the coolant passes through the keel cooler but is waiting for the thermostat to open in order for the coolant to pass through and enter the engine. This is important to take into account when designing the tool, especially if the team were to expand and include calculations for the expansion tank for the after cooler. In such case coolant capacity for the installation would then have to be considered as well.

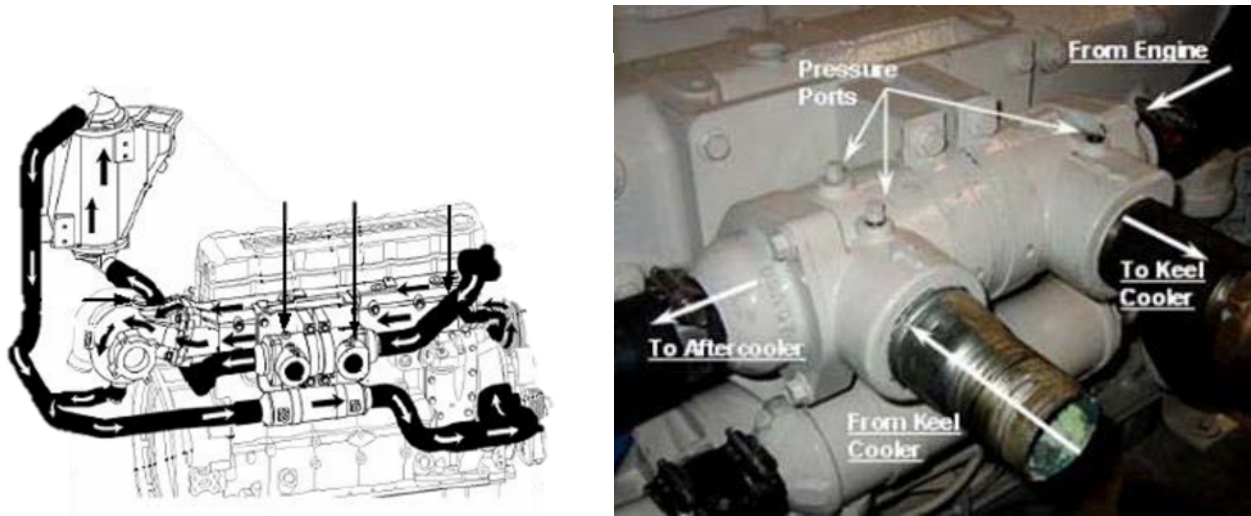


Figure 3. LTA Inlet and outlet connections and pressure service ports on engine

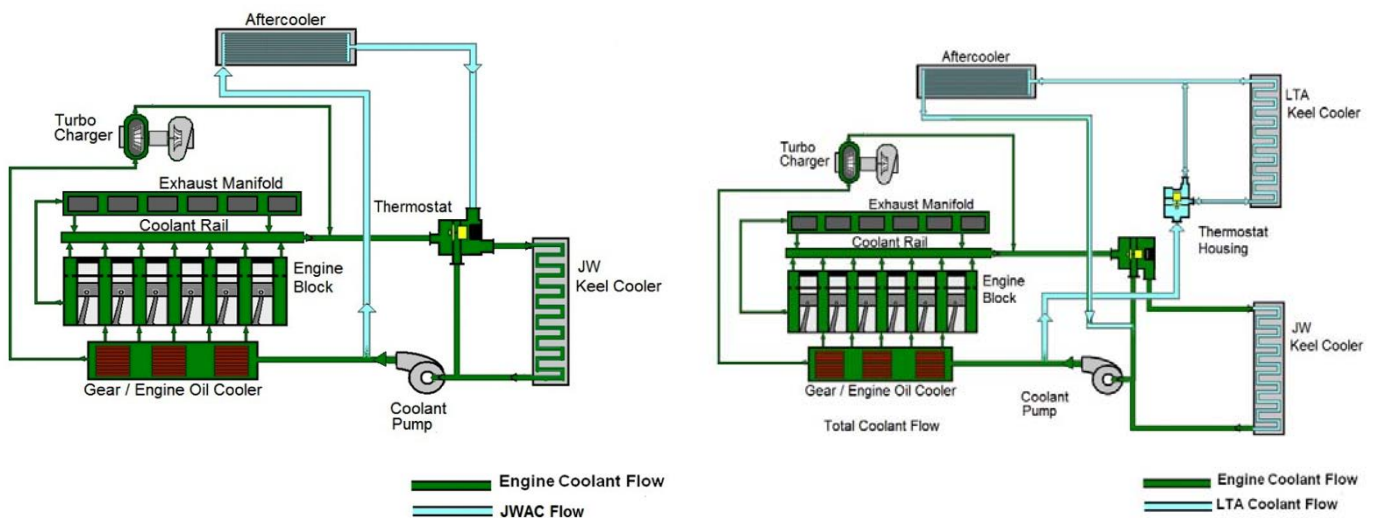


Figure 4. Jacket Water After Cooler system (left) and Low Temperature After Cooler (right)

2.2 Problem Statement

The Senior Design Project for Group 3 for the Marine Keel Cooler Optimization Tool is sponsored by Cummins Marine. The tool currently utilized by the Marine Application Engineers is severely outdated and only returns whether or not the user inputted parameters will result in a passing or failing keel cooler design. The program does not provide any feedback to the designer or operator. This limits the overall design process and does not validate the keel cooler design on the vessel for other nautical water climates and does not evaluate multiple materials which is a necessity for the clients.

Problem Statement for Team 3:

“The current Cummins keel cooler design tool provides no feedback on a particular design and is limited in its capability”

2.3 Project Goal & Objectives

The project should cover all marine engines offered by Cummins, both current production and out of production which will/are installed in keel cooled vessels. The tool is to be used not only to validate the keel cooler system but also suggest the optimal keel cooler design to the boat builder. The tool must be able to calculate and predict how the cooling system will behave under different engine loads and worst case water ambient temperatures. This tool will then be validated through mock keel cooler design and a sea channel keel cooler, depending on boat builder availability, it will be tested on a current installation.

Project Goal for Team 3:

“Design a more versatile design tool which generates feedback and provides a more user friendly interface”

The team decided that the program delivered to the client will provide customer feedback and will offer ways to increase efficiency in the system. The new layout will allow for more customer focused interface, through a quick and versatile programming language which will allow the tool to be converted into other languages easily.

Project Scope:

- The current design has no customer feedback
- Only provides user an output of “Pass/Fail” on design of keel cooler

- Needs to provide recommendations for design improvement
- The device needs to be able to evaluate the design of the keel cooler through the use of different materials (Currently only evaluates steel)
- Current tool is outdated and not user friendly

Objectives:

- Successfully predicts the heat dissipation, efficiency, as well as the optimal operation temperatures for a particular design
- Suggests useful design alterations that would increase the efficiency of the design
- Validate the keel cooler system in scenarios where the vessel is at low idle or relocated to a different body of water (different ambient water temperature)
- Must be user friendly and intuitive
- Needs to provide results that as accurate as possible
- Be able to evaluate keel cooler designs for more materials than just steel

In the house of quality as seen below in *Figure 5*, the two important inputs are customer characteristics and engineering characteristics. The customer characteristics come from talking to our sponsor and asking what exactly the sponsor would like to see in the new program. These characteristics go in the column on the far left side of the house of quality. The engineering characteristics come from our team brainstorming how to best incorporate the customer characteristics into the new program. These characteristics go across the top of the house of quality.

In the customer priority list is how important the customer thought each customer characteristics is. Inside of the matrix is how well each engineering characteristic correlates with each customer characteristic. A weak correlation is a 1, a medium correlation is a 2 and a strong correlation is a 4. After each box has a correlation number in it, each box is multiplied by the customer priority. Finally, each column in the matrix is summed and the largest number is the most important in the design and lowest number is the least important in the design.

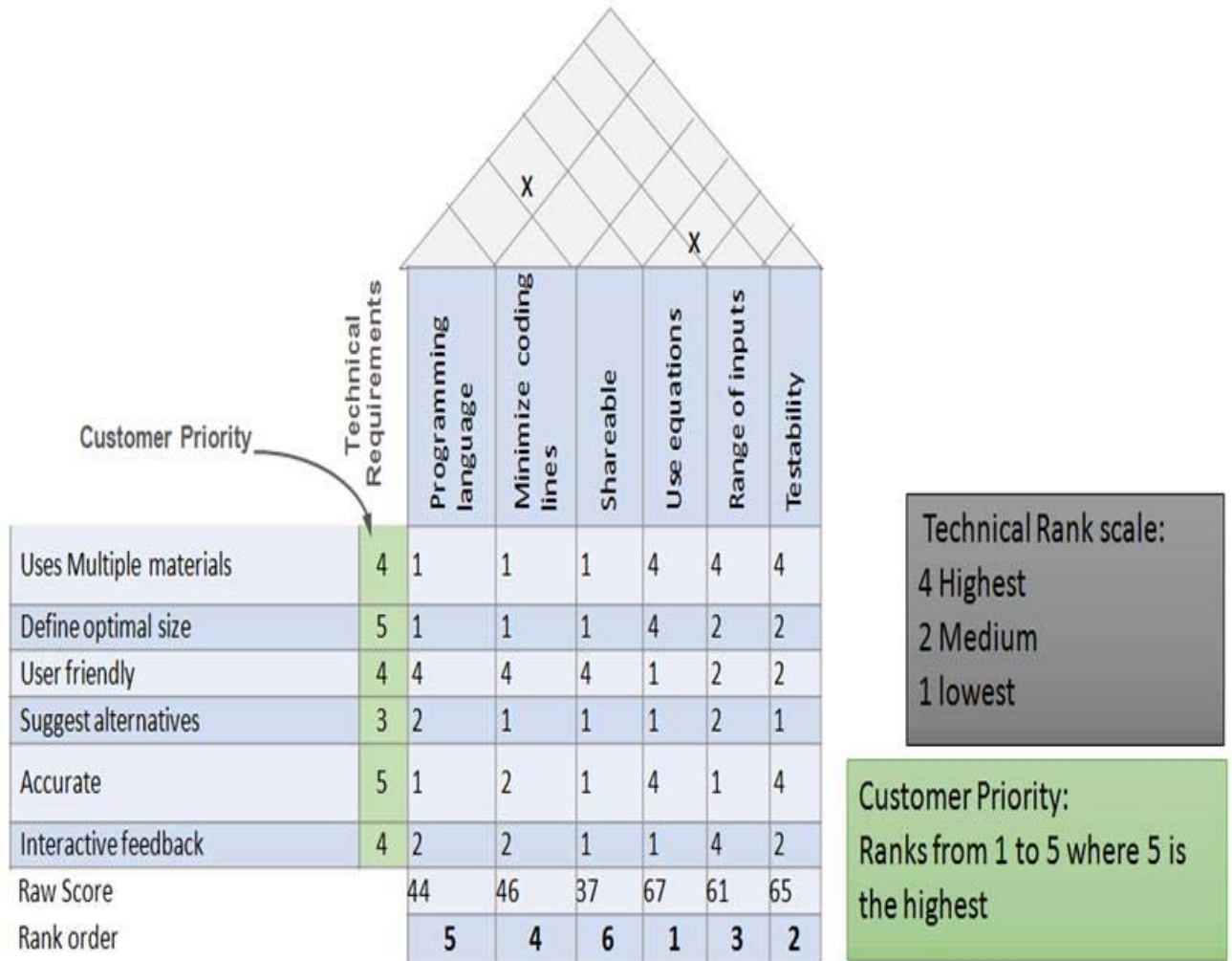


Figure 5. Team 3 House of Quality which helped determine which were the most important factors for the new keel cooler tool

So according to the house of quality, the most important characteristic for our program is the use of equations. For our design this means that the team will need to use more equations that take into account rust, paint, and marine life build up. It also needs to use more equations to account for multiple materials and to provide more customer feedback. The least important factor for our team to take into account is the share ability, the program, since it is electronic, will most likely be easily shared no matter how the program is made.

2.4 Constraints

This project will need to be able to take parameters from the different marine engine models both current production and out of production and be able to calculate the adequate size and cooling needed dependent on the vessel application. With over 15 engines models and each different performance rating available, it is going to be important to find the common variables which can serve as inputs for the tool. The tool must be able to conduct the test for low and high temperature and pressure ranges to ensure the engine receives adequate cooling under different conditions. Since the tool is to be used by Cummins Distributors and Marine Application Engineers around the world, the tool needs to be accessible through the Cummins Marine website as well as available for download (for when the tool needs to be accessed in areas where there is no connection to the web).

The team is also faced with designing a mock keel cooler for testing/validation of the tool. Then possibly creating a keel cooler based on the tools recommendation for one of the engines. The constraint would be adequately sizing a keel cooler for a vessel and not have the tool suggest a keel cooler length longer than the vessel itself. Due to the inherent inefficiencies of keel coolers, the tool needs to be able to process when to suggest different numbers of flows to improve cooling and ensure coolant velocity is below critical speeds.

Constraints:

- Budget of \$2,000
- Time
- Material Acquisition

3. Design and Analysis

3.1 Software Design

At the onset of the project, the proposed plan was to design a tool that evaluates keel coolers and suggests design improvements. This tool was to be later developed into a web-based application. This project was to serve as a replacement for the current validation tool that Cummins Marine employs for generating a pass/fail condition for C-channel keel coolers. The project began as a simple C-program that would run through a terminal but was later developed into a python program that ran-through a web server. The group determined that it would be extremely beneficial to create this web-based application as their final product an exceed Cummins expectations for this project.

The choices for the programming language were based on a comprehensive list of the suitable languages that the members of the project group had exposure to. In order to select a language for consideration, the project group agreed upon options that they felt familiar with, had the capability to do mild computing, and had useful functions for implementation. The three final choices were reduced to C programming, MATLAB, and Python. A decision matrix was implemented in order to make the proper selection for the task. The group chose four main attributes and weighed them in order of importance to evaluate their ranks. The judging criteria was knowledge (the groups familiarity with the language), structure (does the program contain useful functions to structure a logic based selection system), aesthetics (user friendly interface), and relevance (how universal is the language). The attributes were ranked from 1 to 10 and were given a weighted multiplier with knowledge and structure given 60% and 20% weights respectively. These values were added up to produce a score. Python ultimately prevailed to its superiority in knowledge, structure, and relevance. The decision matrix can be seen in *Table 1*.

Table 1. Decision Matrix

| Program: | Knowledge | Structure | Aesthetics | Relevance | Total: |
|----------|-----------|-----------|------------|-----------|--------|
| C | 9 | 10 | 1 | 8 | 7 |
| Matlab | 8 | 1 | 8 | 6 | 6.4 |
| Python | 7 | 9 | 10 | 9 | 8.75 |

The second step of the design process was to identify the user in order to satisfy their needs in a program. After a coordinated sponsor meeting, the team was able to identify the primary users as application engineers as well as shipyard workers. After further inspection into the uses of the program, the design team concluded that the two primary uses of the optimization tool are for the validation and design of keel coolers. This led to the implementation of two main program modes, a verify mode and a design mode. The ‘verify’ user is one that has a premade keel cooler or keel cool design and wishes to simply evaluate whether their model will pass or fail. If the users design passes, the program would indicate so. If the users design fails, the program would allow the option of providing feedback in order to produce a passing design. An emphasis was placed on useful feedback of the program, providing the user with only the information that they seek, optimizing the user interface. The design mode of the program offers an alternative approach for a user who is seeking the optimal size for their particular parameters. The user would provide constraints such as available material, hull size, amongst others and the program would evaluate the correct size (if possible) for a passing keel cooler design.

| Engine Data | | |
|---|------------------|---|
| Engine Model | | from General Data Sheet |
| Engine Brake Horsepower [BHP] | | from Performance Data Sheet |
| Engine Speed [rpm] | | from Performance Data Sheet |
| Select a Cooling Circuit Type | | from Performance Data Sheet |
| Total Circuit Heat Rejection [BTU/min] | | from Performance Data Sheet |
| Coolant Flow to Keel Cooler [gpm] | | from Performance Data Sheet |
| Engine Coolant Capacity [gallons] | | from General Data Sheet |
| Coolant Type (50/50 glycol or Water/DCA) | Make a Selection | 50/50 Glycol solution preferred |
| Maximum Sea Water Temperature [deg F] | 85 | Typical sea water temperature is between 75-85 deg. F |
| Design Speed [knots] | | Typical sizing speeds are: 1) Tugs/Pushboats: 1-2 knots 2) Generator set: 0.1-1 knots |
| Keel Cooler Data | | |
| Standard Channel Size | Make a Selection | C depth (inches) x Weight Per Unit Length (pound force per foot) |
| Channel Width [inches] | | from standard steel channel tables |
| Channel Height [inches] | | from standard steel channel tables |
| Web Thickness [inches] | | from standard steel channel tables |
| Cross Sectional (Web) Area [sq. inches] | | from standard steel channel tables |
| Coolant Velocity [ft/sec] | | Best if kept between 2-8 ft/sec |
| Channel Material | Steel | |
| Total Installed Keel Cooler Length [feet] | | Increase cooler length or number of flow paths until Pass/Fail criteria is met |
| Thermal conductivity "k" [BTU/hr-F-ft] | 26.5 | |
| Number of Flow Paths | | |
| Results | | |
| Actual KC Exterior Area [sq. feet] | | |
| Calculated Exterior Area [sq. feet] | | |
| Minimum Keel Cooler Length [feet] | | |
| Minimum Expansion Tank Capacity [gallons] | | from Installation Directions bulletin No. 3884744 |
| Passing Criteria [Pass / Fail] | | Increase cooler length or number of flow paths until Pass/Fail criteria is met |

Figure 6: Current Specification Sheet

The program would use the same input parameters as the current validation tool, while adding the variability of adding design parameters for additional analysis. *Figure 6* shows the specification sheet for the current tool.

Several considerations were given to the program structure. In order to minimize run time and maximize coding efficiency, the program was structured with a main function with conditionally accessed sub functions. The program will open up from the start and prompt the user to choose whether they would enter design or verify mode. The selection decisions are prompted by the use of switch statements that access sub-functions depending on the number that the user enters. If the user enters the analyze mode, the engine selection tool will ask the user for their engine selection. Depending on their response, the program will access separate functions which will store the parameters corresponding to their selection. Following the engine selection, the user will enter their coolant selection followed by their channel size dimensions. The program takes the information stored from these inputs and calculates the minimal cooler length and heat dissipation and compares it to the users input. If the user's cooler parameters correspond to a passing design, the program will display a message indicating passing and the program will terminate. If the user enters parameters that prompt a failing condition, the user will be informed as well as given the option to enter the programs design mode. The design mode will invoke a similar structure to the verification mode, with the exception that the user will be able to select additional parameters such as boat hull size that constrain their design. The program will evaluate the user's criteria and generate design parameters that will provide a passing condition. Because of the vast number of sub-functions the program employs, most of the variables will be redefined by the use of pointers. This reduces the number of variables and the memory required by the program minimizing run time. *Figure 7* shows a simplified flow chart for the Keel Cooler Optimization Tool.

As the first program prototype was being developed, the group continuously sought out avenues for making the best possible user-interface. This is where the group began to re-evaluate its selection for the design of this tool. C-programming is a low-level programming language that is best executed in machine programming, it generates instructions directly to a computer's central processing unit (CPU). It is ran and compiled in a computer's terminal, eliminating the potential for a great user-interface.

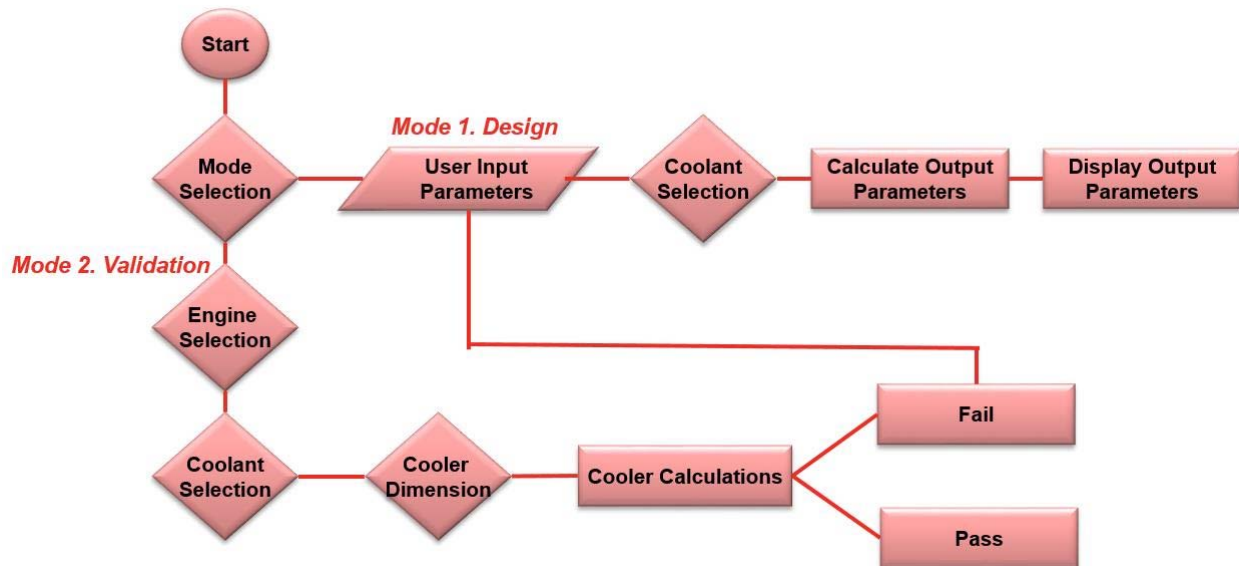


Figure 7: Flow Chart for Keel Cooler Optimization Tool

The general deployment of this program would require the user to install a program to run and compile the script for the tool. This is when the group began researching popular languages for web-development that would enable the creation of a custom user interface via an html web-page while allowing the computations to be carried out in the back-end. Python became the ideal choice because of it is a high-level dynamic programming language, its design emphasizes program readability and syntax. It is widely used by large organizations such as Google, Yahoo, and NASA.

In order to create the web-application in a timely fashion, a web framework was employed. A web framework is a software framework that enables the development of web applications and eliminates the tedious programming associated with writing code for session management and templating. Django was the chosen framework because it is free and open-source. Django is well known and reputable and is employed by various well-known sites such as Pinterest, Instagram, Mozilla, and The Washington Times. The Django framework enabled the online deployment of the keel cooler optimization tool and was an incredible resource in the creation of the project. For more information on Django, or to make a donation to their free software foundation, please visit “djangoproject.com/fundraising”.

In order to fully understand the development of the Keel Cooler Optimization Tool, it is important to understand the structure of the python classes that set up the program. *Figure 8* shows the file structure along with the classes of the program. The entire program is contained in a folder named FirstBlog. The first three subdivisions of the program are the folders blog, FirstBlog, and manage.py. Manage.py runs the server for the program and can be seen as the programs heart. The folder FirstBlog carries the same name as the main program folder and it contains the settings.py, urls.py, and wsgi.py. These three classes set up administrative access to the program, receive URL requests, and communicate between the servers. The folder blog contains the largest bulk of the program. The subfolder static contains the static files such as the css, JavaScript, and image files, while the templates subfolder contains the html files that serve as the user interface. The admin.py, apps.py, forms.py, and models.py set up templates, enable the use of forms, and define site administrators. The views.py file serves as the brain of the program. The views file

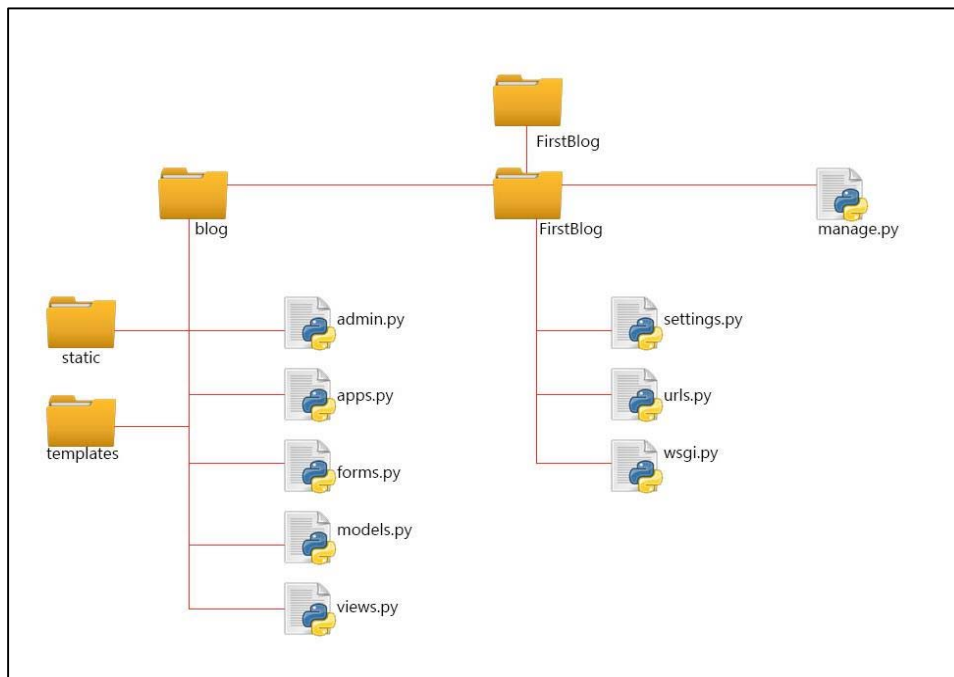


Figure 8. Program file structure

receives URL requests and renders html files and performs the thermodynamic calculations for keel cooler evaluation.





The html file and the views file interact extensively in order to create generate the required calculations. Information is taken within the html file by forms which send user inputs to the views file once the user presses submit. The options that are prompted by the program are shown in *Figure 9*. The first form presented is the engine selection. The four main engine types that were

focused on were the QSK models 19, 38, 50, and 60 marine propulsion engines.

Welcome to the Keel Cooler Tool 1.0

Please select an engine, coolant, material, channel size and flowpaths to receive the proper measurement of heat dissipation.

Engine Selection:

QSK19  QSK38  QSK50  QSK60 

Coolant:

Channel Material:

Channel Size:

Number of Flow Paths:

Installed Keel Cooler Length:

Vessel Speed (knots):

. Figure 9. Marine Keel Cooler Optimization Tool Entry Form

Once the user has made an engine selection, the program would internally store an integer number value corresponding to the engine. If the user selected the QSK19 engine and submit the form, the views.py function would store the integer value in a variable called “engine_select”. The program uses an ‘if else’ statement to decide which function to employ that would set variables for calculation. If the variable “engine_select” equals one, the program then calls upon the QSK19() function which assigns values to the heat dissipation, inlet temperature, and volume flow rate, all of which correspond to the QSK19 engine. The same sort of process is used to calculate

the parameters for coolant type, channel material, and channel size, due to those values being conveyed as integers within the html file.

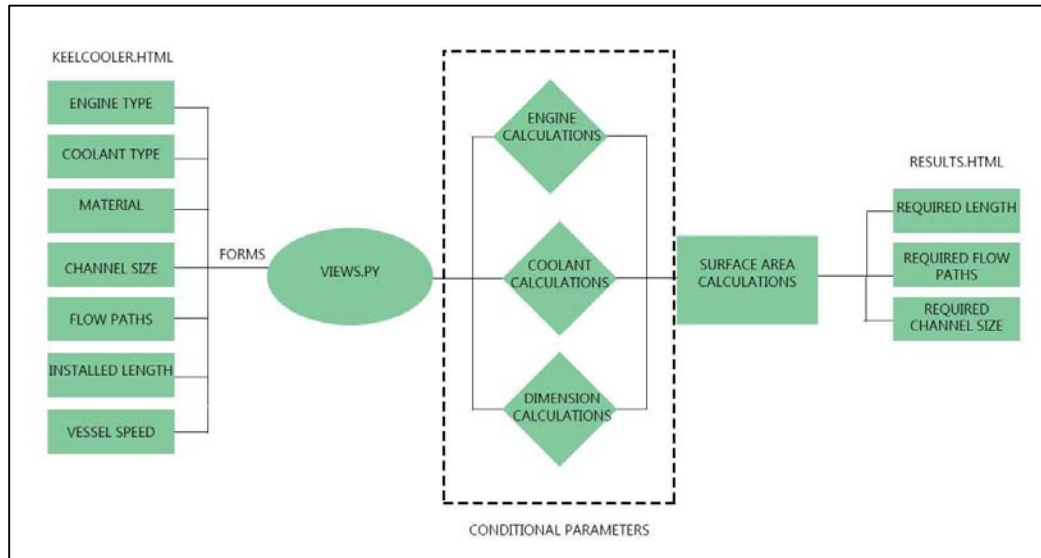


Figure 10. Flow chart of program

For the number of flow paths, the user is prompted to enter an integer value corresponding to the desired number of channels in their proposed keel cooler design. If the user enters an incorrect value (a decimal or negative number or character), the box around that selection will display in red with an error message. The user will also enter their planned keel cooler length and vessel operating speed. Unlike the engine selection, the number of flow paths, keel cooler length, and vessel speed values are directly passed into the program with no need to conditionally calculate variables. Once all of the data is collected, the submit button sends the information to a function called `formview()`. This function receives the URL request as well as the data from the forms and performs the engineering calculations for required surface area and renders the calculated results to a `results.html` page. This value corresponds to the surface area that would be necessary with the selected channel size to dissipate the required heat load for their selected engine type. The program calculates the proposed surface area with the length specified in the form and compares it to the required surface area. If the required area is less than the proposed area, then the proposed design passes and a pass condition is stated. If the required area is less than the proposed area, then the 'Fail' condition is initiated. The 'Fail' condition will then calculate the minimal keel cooler length according to the required surface area, specified number of flow paths, and the selected channel type. As an additional alternative, the program will calculate the required number of flow paths if

the user wants to keep the length and c-channel size the same in their design. For a third alternative, the program will keep the length and number of flow paths the same and run a while loop that calculates required surface area with various channel sizes until the passing condition is met. *Figure 10* shows a flow chart of the overall program sequence.

3.2 Hardware Design

Throughout the design process there are three underlying motivations: manufacturability, reliability, and economy. When designing a product for manufacture there is an implicit obligation to make the manufacturing process as efficient as possible. Having this foresight during the design stage will make for an easy transition to production later. Once manufactured though, how consistent will the product be and for how long? This is typically determined by the intended application and is ultimately limited by cost. This leads to the final motivation: economy. It is important to consider cost when designing a product and how these costs may limit the overall design. If there are similar products on the market, it would be beneficial to design a product at a competitive price. These concepts will help streamline all processes from concept generation all the way to post-production. Our team made every effort to include these motivations into the design of our Keel Cooler Optimization Tool.

Although the project is based on the creation of a software, the team found it useful to design and construct a testing apparatus in order to validate the predictive engineering written in the software. The team would have to create an apparatus which would simulate enough aspects if an installed engine/vessel package in order to ensure the most accuracy. Ideally, it would have helped the team to ultimately have a rig which consists of a piece of a keel, which would have required an immense water bath of flowing water. However, such simulation would have been impractical. The team then decided on a smaller design, one which would involve a heat exchanger that models a keel cooler assembled using readily available manufactured parts. The testing apparatus the team will create is illustrated and explained in *Figure 11*. The testing apparatus will consist of a 55 gallon drum. This will hold the water which will be run through the keel cooler which will be heated by a heating element to a constant 40 °C. The drum will also be equipped with a water pump rated at 20 gallons per minute. The water pump will cause the water to be carried through a hose which will connect to the keel cooler the team will make. The keel cooler will be submerged in a water bath. The water inside the water bath where the keel cooler is located will not be flowing,

since this will simulate the worst case operating scenario for a vessel, wide open throttle at 0 knots. This is common for vessels such as tugboats which push or pull other vessels and at times find themselves operating at open throttle but stationary in the water. The keel cooler will have an outlet pipe which will carry the used water to a rejection tank. This testing apparatus provides the team different locations to record temperatures in order to calculate thermal efficiency. Temperatures will be recorded through the use of a resistance temperature detector (RTD) sensors at the 55 gallon drum to ensure the temperature is held constant, the water bath where the keel cooler is located to see how the effects of the warm water carried through the keel cooler affect the surrounding simulated sea water, and the outlet water which will allow the team to calculate the thermal efficiency of the keel cooler.

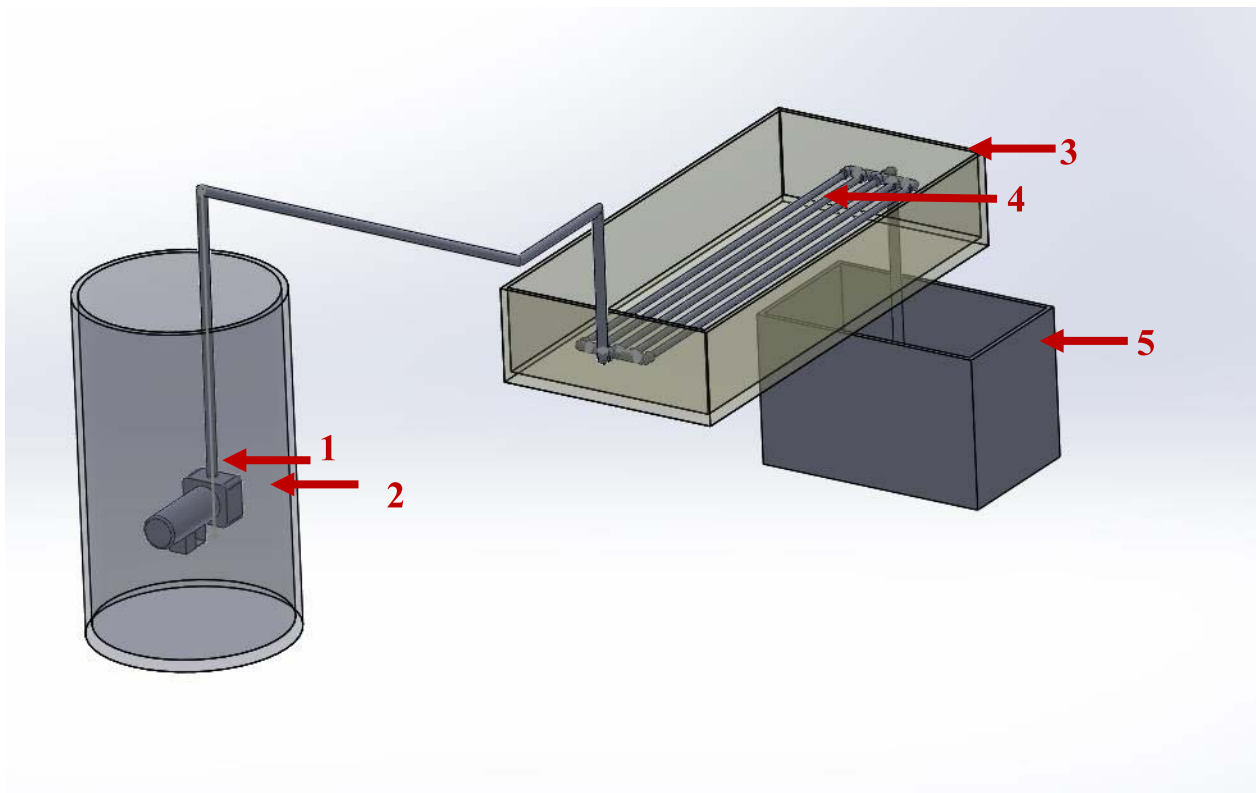


Figure 11: The keel cooler testing apparatus is illustrated above. The water pump (1) installed inside the 55 gallon drum (2) which will carry the heated water. The water bath (3) where the keel cooler (4) will be submerged while testing. The outlet water tank (5) will provide the team a means to measure outlet water temperatures to calculate the thermal efficiency of the keel cooler.

In order to properly size a keel cooler which would work for the testing apparatus, the team took several design aspects into consideration. The created keel cooler could not be of an

overwhelming size, since it needed to be practical in size for portability. The team had originally thought about designing a keel cooler for the QSK19 engine model, which would have caused the team to have to create a keel cooler that was too long for this purpose. The team then planned on creating a downsized version of the keel cooler for the QSK19, since it was believed the keel cooler would still represent the same cooling values. However, it was advised this was not done, since the thermodynamic properties do not size how the team thought it would. The team then shifted the brainstorming from utilizing the QSK19 engine to a 6B engine. It is a lot smaller engine, rated at 150 HP at 1800 RPM, but it allowed a more practical keel cooler testing apparatus. From the technical information available on the Cummins Marine website, utilizing the database for Application Engineers, the sea water pump performance for the 6B engine was obtained. Provided the engine size, the sea water pump performance would need to be able to operate within the range of 6 gallons per minute through 20 gallons per minute. This would allow the team to simulate the operation of a 6B engine operating through a range of 1500 RPM through 1900 RPM. The reason why it is beneficial for the team to operate the testing apparatus 100 RPM beyond the rated power for the engine, is when an engine is installed on a vessel, the operator is allowed to run the engine 100 RPM over the rated value, since the power band follows along the governor's curve. Therefore, the team selected a water pump which can operate to a 20 gallons per minute flow rate, but can also be operated at a lower flow rate. Such ability would enable the team to test the keel cooler in more simulations than just one.

Basing the design on an actual engine allowed the team to compare the design to previous installations. The team wanted the keel cooler to be a multipurpose testing apparatus. The team wanted to be able to test the keel cooler as a single flow path and multiple flow path cooler. This presented itself to be another design challenge since the equations utilized to size the keel cooler take the number of flow paths into consideration. Therefore, the creation of a keel cooler which would be the same length, yet carry a different number of flow paths would not have the same thermal efficiencies. This was kept in mind when designing the keel cooler for the hardware testing apparatus, but it was said it would be adequate for the simulation the team was conducting since it would show a dramatic difference thermal efficiency of what a single flow path versus a multiple flow path keel cooler. Since the software will have the keel cooler validation and design aspect, it will help validate the engineering with real life results.

The keel cooler was designed to use aluminum tubes for the keel cooler. This material was chosen over steel since it is more cost effective to construct out of aluminum, and since the current tool is limited to evaluating steel keel coolers, the team felt it was best to carry out the experiment with the material which is being added to the program. The sizing of the pipe for the keel cooler as well as the required surface area of the cooler was calculated utilizing the equations 1 and 5 from before. It was calculated that a four foot long keel cooler would be adequate with 5 flow paths. Since the team will be able to simulate both a one path keel cooler and a 5 flow path keel cooler, the team will be able to compare the thermal efficiencies between the both as well as better understand what calls for a better designed keel cooler.

3.3 Functional Analysis

The goal of the keel cooler optimization tool was to develop a more functional and user-friendly user interface with more feedback to replace Cummins current Keel cooler validation software. This is done by performing thermal calculations that measure heat dissipation of a proposed keel cooler design with the necessary heat dissipation of a desired engine. In order to achieve the best possible user experience and to increase the accessibility of the program, a web-based application was developed. The program can be divided into three main components, the front-end interface, the server framework, and the back end python code. The front-end user interface was designed using html, css, and JavaScript. The Python Web framework that was used was Django, which is free and open source and provided a pre-made template for rapid web-development. The program language used was python due to its relative ease to use, and its commonality in modern web-development.

The program by itself is structured by dividing its components into classes which carry out various functions. The `manage.py` class runs the server and relays information between the server and client to provide a web host. Within the program, the `settings.py` class establishes databases, authorizes users, hosts, and adds templates. The `urls.py` class handles URL requests from the webserver. This class tells the program when an address is entered and sends it to the `views.py` class. The `views.py` class accepts requests and processes the user inputs. When a URL request is made, the `views.py` class sends the user to their requested page.

The thermal analysis that evaluates the user's data takes place within the views.py class. The class is composed of a division of several sub functions that perform calculations or obtain values based upon the user's inputs on the html page. The functions can be categorized as data-collecting and data-processing. Data-collecting functions are conditionally accessed based upon the user's inputs on the html page. The html page contains several different types of selection options which translate to calculation parameters in various ways. The engine selection, shown in *Figure 12*, uses a bubble selection in order to decide which parameters to use. When the user selects the QSK19 Engine, the html page sends the value 1 to the views.py class which uses an if-else statement to access the QSK19 function that fills various thermal parameters unique to that engine.

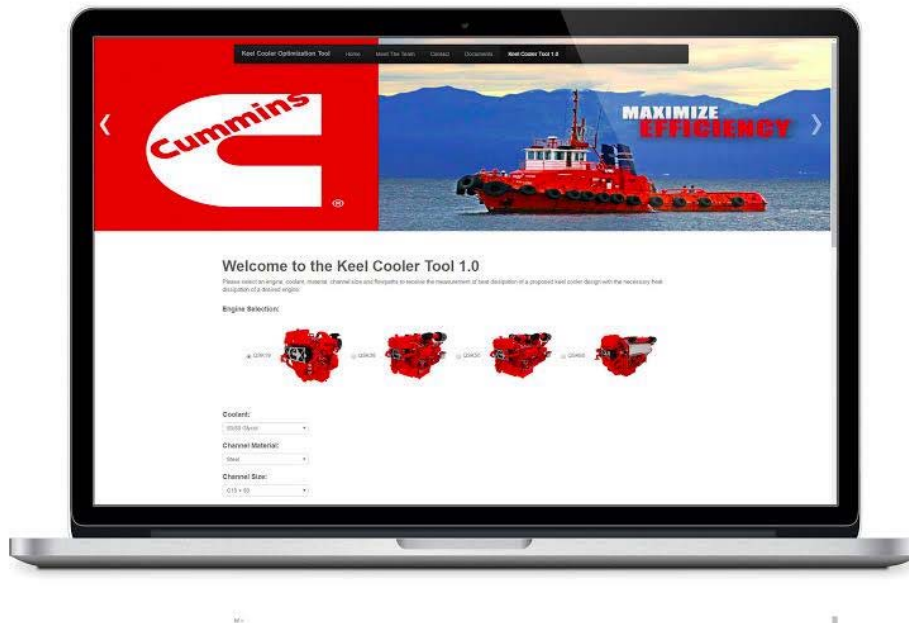


Figure 42. Engine Selection View to user

A similar type of information processing is used by the drop-down box which prompts the user to select a standard channel size. The program does not actually make calculations based on the name that is selected, rather it references the selections order in the drop-down box. For example, when the C15 x 33.9 channel size is selected, the program receives the number 3, which corresponds to the third value in the channel size drop-down box. This value is used to access the depth, area, thickness, and thermal conductivity of that channel size.

Once the user inputs their value for the engine selection, coolant type, channel size, number of flow paths, and selected length, the program takes those stored values and uses an array

corresponding to length values to compute the minimal length required to dissipate the required heat for the engine. The minimal keel cooler length is displayed along with a pass or fail condition based on whether the analyzed length met the minimal required length standard.

In order to generate a result, all data fields must be inputted correctly. The user has the option of inputting their own number of flow paths but the value must be a valid integer number. Decimal values, symbols, and other non-numerical characters will return an error prompting the user to make a valid selection.

3.4 Operation Instruction

To use the keel cooler optimization tool to its full capacity, the user must have some information gathered which pertain to the vessel being worked on. The user will select whether to use the validation aspect of the program or the design aspect. If the user selects the option for keel cooler sizing validation the important parameters to have on hand are: engine model, coolant type, keel cooler channel material, channels size and the number of flow paths. In order to properly evaluate your application requirements, the user must first select which engine you will be using.

Engine Selection:



Figure 13. The first information the user will be prompted to enter.

The user will then make an engine selection by simply clicking the engine to which the user wants to use, as shown above in *Figure 13*. Once the user completes the engine selection, the user will be prompted to select which type of coolant the engine will be using from the drop-down list, as shown in *Figure 14*. (Note: Only the coolant types shown are compatible with this tool. No guarantees can be made if another coolant type is used, since only the specified coolant constants have been accounted for).



Coolant:

50/50 Glycol ▼

Figure 14. Example of Coolant selection

The user will then select which material the keel cooler will be made from. This is important since steel and aluminum have different thermal properties. This program allows the user to select between steel and aluminum, a new validation parameter introduced due to the voice of the customer.

Once the material for the keel cooler is selected, the channel size must be selected. The program uses the American Standard Steel and Aluminum which includes the most common used channel sizes which frequent many boatyards. This ensures accuracy in the validation calculation. Following this step, the user will enter the number of flow paths in the design of the keel cooler. (Note: The number entered must be an integer). If an integer value is not entered the tool will generate an error message in red located at the bottom of the window.



Channel Material:

Steel ▼

Channel Size:

C15 x 50 ▼

Number of Flow Paths:

Figure 15. Channel Material, Channel Size, and Number of Flow Paths selection.

Once the user has entered all of the information, the user must click the submit button and the relevant data will be produced under the “RESULTS:” heading. Here the user will find data which evaluates the proposed keel cooler specifications and provides relevant feedback (ie. pass/fail, recommendations for meeting the cooling requirement). If the user’s selection criteria is unable to effectively cool the engine, additional suggestions will be provided. If the selection criteria is satisfactory for the intended application, the tool will display an “APPROVED” notification at the bottom of the window. (Note: only the parameters that are provided are eligible

for evaluation. The use of any other configurations other than those provided by this tool cannot be guaranteed.)

3.5 Troubleshooting

The program has been tested stringently so that there is no need for troubleshooting the program itself. The program responds to errors in in data input by displaying an error message. If after inputting the data for a validation result, an error message will appear. Make sure that all fields are filled and that they have the correct data inputs. The lone open text box for inputting number of flow paths will only accept integer values. Other character types will generate an error message prompting the user to input a valid integer.

3.6 Regular Maintenance

In order to stay up to date with the latest engine catalogs, the persons in charge of website oversight should add or remove additional functions with new engine types along with their respective specifications to the views.py class as well as the keelcooler.html script. The thermal analysis will not change therefore the remainder of the code will stay valid.

Other updates to the program will have to be made in the future to ensure it continues to be relevant as Cummins develops more engine models and releases new fuel ratings. This final step is important since it will ensure continuity, reliability, and accuracy. The tool developed for Cummins will allow for such updates and will continue to serve the Application Engineers, technicians and boat builders for many years to come.

4. Methodology

4.1 Marine Keel Cooler Optimization Development

To ensure accuracy and Cummins industry standards were met, extensive research was conducted in the design and science behind marine keel coolers. It is important to properly define the input design parameters since they were needed to be able to be utilize cross engine models and performance ratings and provide the user accurate results.

Once the proper parameters were defined, the program was written to utilize the proper equations, constants and provide proper feedback to the user. Ultimately, not only provide a pass/fail result, but allow the option of the material used as well as a recommendation for the adequate sizing of the keel cooler per engine/vessel installation.

4.2 Risk Analysis

When planning a project it is important to first look into and assess risk factors that may affect you and your project. When considering risk we break up the project into 4 different sections. First we start with concept generation which is a no risk section of the project that involves the gathering of information and the formation of ideas on how to tackle the project. Next is product assembly, for our project of the keel cooler optimization tool this involves programming and calculations. This creates a very low risk situation due to the very little physical labor and no use of heavy machinery. Risks during this process are limited to exhaustion. After product assembly comes product testing, in this stage of the project risk starts to become a serious factor and is considered a high risk stage. The Risk is due to the mechanical machining that is required in order to create the keel cooler needed for testing. Possible risks include cuts burns abrasions and possible loss of limb. The large bath in which the keel cooler was submerged could be another area of risk. This is due to the fact that the pool is very large. This leaves room for drowning and in open source of water. Additionally electric shock can occur if electric components were to get wet. In order to insure safety while completing this process it is important to wear proper safety gear, be properly trained on each machine used, work in groups, be properly supervised and to file a safety plan that outlines other possible hazardous situations and how to deal with them. Lastly there is risk in product implementation, the risk in product implementation is a different type of risk that occurs if our keel cooler optimization tool is not successful. This risk includes ruining motors and

stranding boats in the middle of the ocean this also includes the ruining of reputations of both our team members and Cummins marine division. With so much risk involved throughout this project it is important to be careful and precise in every step of our project.

4.3 Design for Manufacturing

Since our project is code-based, the term “assembly” is used loosely. Although the team is technically putting the code together, the process consists of more than just designing/ordering parts and putting them together. In addition, the team’s general lack of proficiency in various programming languages exacerbated this effect. With the framework and logic established using C-programming language, the team began by cataloging all of the relevant constants and parameters that are involved in keel coolers. This included engine data for various Cummins power plants, properties of various fluids, dimensions for standard steel c-channels, etc. and debugged. Next we incorporated all of the thermodynamic principles pertaining to keel coolers. We modeled our application to the most appropriate heat exchanger configuration and arranged all of the pertinent equations in a manner that the program could process. Once debugged, we then added functionality for multiple materials, namely aluminum, which included all of the relevant dimensions for aluminum c-channels. It was at this point we decided to transition to Python for a cleaner and graphical user interface, and also to allow for web-based interaction.

After the code was converted and web functionality established, everything was debugged one last time. The program is still being tweaked and modified and will continue to be optimized as necessary. The timeline for progress on the program is very sparse, with intermittent breakthroughs occurring here and there and not a lot in between. This is the nature of coding in a new language; there is much to learn along the way. Although the progress of the program came in bursts, the overall time required to put it together seems adequate, given the circumstances. The process could have been expedited had the team been more familiar with different coding languages, or for an added cost, outsourced to someone more skilled. The number of components involved in our optimization tool would most appropriately refer to the amount of complexity in the code itself. After the program was first completed, we evaluated it for efficiency and made efforts to simplify its structure. We cleaned it up and made it so that the code was more organized and efficient without removing any functionality. There can always be improvements, and we will continually make efforts to enhance our product.

4.4 Design for Reliability

The lifetime of the software designed in this project is meant for a long life. The previous program used by Cummins was designed in the 70's. This shows that the program needs to be designed with plenty of future use.

The program our team designed will ideally perform the same task as many times as necessary. There is no limit to how many times it can evaluate a keel cooler design. The program will deliver consistent results every time it evaluates since the formula in the program does not change. The only exception would be if the program was altered after the program was delivered and Cummins felt it necessary to do so.

The main reliability concern for the program is that it will not evaluate every keel cooler design perfectly. This is due to the fact that there is a small possibility that our team may have used some assumptions or formulas that might not evaluate perfectly. What this means is that a cross sectional area may have been assumed to be rectangular when in fact it could be more of a trapezoid shape. Although the two shapes can be very similar, they can also be fairly different as well. The way to address this problem would be to test the program against data that already exists and see how close the program is to the actual numbers. If there is a discrepancy, our team could potentially go back through the program and try to figure out where we could tweak the formulas that would give a possibly more accurate result.

As for the hardware side of the design, it was designed to evaluate our engineering intuition that went into developing the software side of the program. This testing hardware was designed so that it would be cheap, easy to construct and deconstruct and have little to no manufacturing time. So for the reliability of the hardware, we made it taking into account possible leak points. This would be the worst case scenario for our hardware since we do not want any hot fluid to leave the system. This would give us inaccurate data and possibly skew our results.

This hardware setup should ideally perform the same each time it runs. The only changes that could possibly happen within the hardware would be pipe erosion or corrosion. This could just be due to age and use. The leak points (between pipes, at flanges, and at hoses) would need to be checked regularly to ensure that no leaks have occurred over time if the hardware is used often.

The main reliability concern for the hardware would be leaks as mentioned before. These leak points have been mitigated as much as possible so far by using pipe tape and by securely

fastening flanges together. This would be addressed by checking the pipe connections every so often to ensure they have not loosened up or that the pipe tape does not need to be replaced

4.5 FMEA Analysis

Although the project is based on the creation of a software, the biggest possibility of system failure would occur if the program outputted the incorrect feedback. For example, if the tool told the user the keel cooler being validated passed the cooling requirements, when it actually did not. The engine in question could end up overheating in the worst case scenario, leading to engine failure. This would lead to an immense cost to the company to cover for warranty.

4.6 Procurement

Table 2. Material Procurement

| Description | Price | Quantity | Total Price |
|---------------------------------|----------|----------|-------------|
| BriskHeat Plastic Drum Heater | \$239.00 | 1 | \$239.00 |
| 55 Gallon Plastic Drum | \$143.00 | 1 | \$143.00 |
| Flanges - Steel | \$16.88 | 10 | \$166.80 |
| Flanges - Aluminum | \$15.22 | 10 | \$152.20 |
| Breadboard | \$30.00 | 1 | \$30.00 |
| Connectors - pipe - 3" | \$1.11 | 8 | \$8.88 |
| Connectors - pipe - 2" | \$0.95 | 12 | \$11.40 |
| Connectors - tee | \$2.30 | 4 | \$9.20 |
| Connectors - elbows | \$1.73 | 6 | \$10.38 |
| Connectors - cross | \$8.72 | 2 | \$17.44 |
| Connectors - tee (extras) | \$2.30 | 2 | \$4.60 |
| Connectors - elbows (extras) | \$1.73 | 2 | \$3.46 |
| 1/2" hose clamps | \$6.00 | 1 | \$6.00 |
| 10' long 1/2" diameter hosing | \$14.50 | 3 | \$43.50 |
| 1/2" RTD sensors | \$15.00 | 2 | \$30.00 |
| Arduino 3 | \$60.00 | 1 | \$60.00 |
| Connectors - pipe - 3" (extras) | \$1.11 | 4 | \$4.44 |
| Connectors - pipe - 2" (extras) | \$0.95 | 4 | \$3.80 |
| Test Section - Aluminum - 4' | \$49.57 | 5 | \$247.85 |
| Gasket fit w/ Fasteners | \$6.61 | 10 | \$66.10 |
| Lock nuts 1/2" black iron/steel | \$2.47 | 24 | \$59.28 |
| 1" to 1/2" barb tube fitting | \$8.05 | 3 | \$24.15 |
| 1/2" to 1/2" barb tube fitting | \$4.34 | 3 | \$13.02 |
| 12' Diameter swimming pool | \$79.00 | 1 | \$79.00 |
| Water Pump | \$374.52 | 1 | \$374.52 |
| Remaining Budget | | | \$191.98 |

Our group was given \$2000.00 for purchasing and building our hardware. These purchases can from a variety of vendors. *Table 2* shows the parts that were ordered to make the keel cooler. Of the \$2000.00 budget we spent a total of \$1808.02 which left us with a \$191.98 amount remaining. This budget was plenty for our project and every piece ordered was used to the fullest extent.

4.7 Design for Economics

The Keel Cooler Optimization Tool is an upgraded version of the current tool Cummins marine uses that is able to evaluate keel coolers with Steel C channels. Senior Design Team 3 has upgraded the tool to a web based application that will be able to evaluate multiple materials in C channel form. Because this is an upgrade from an original Cummins design there are no competitive products.

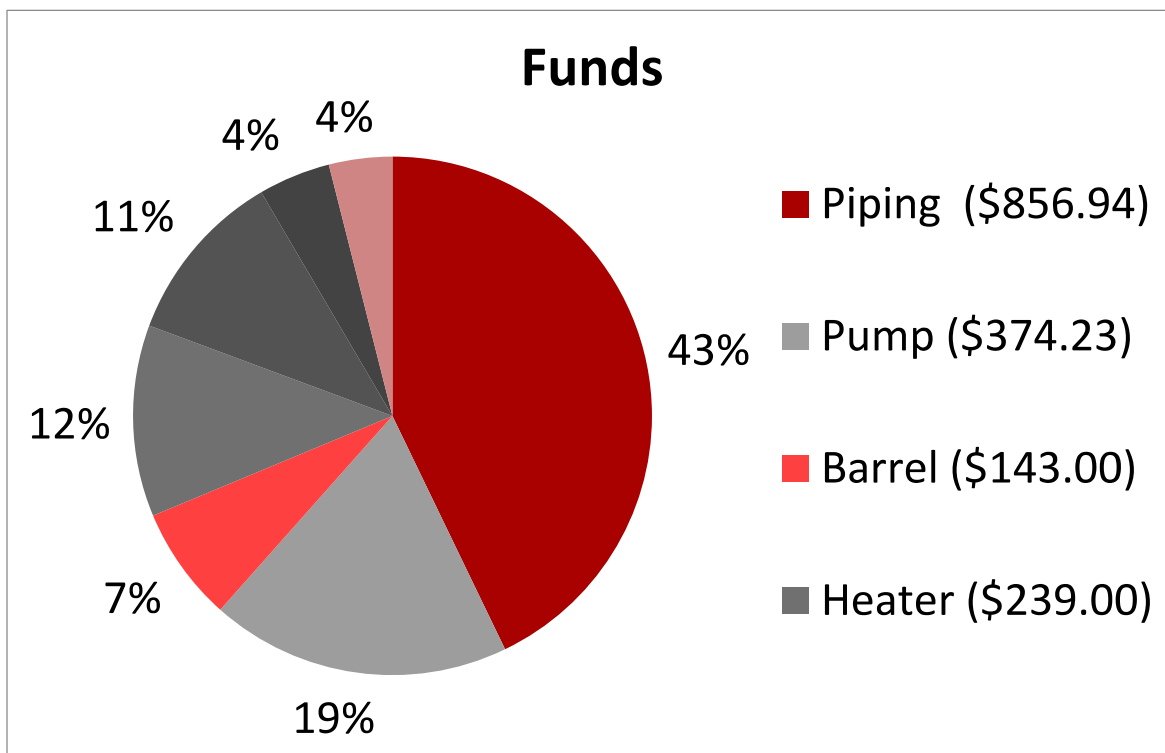


Figure 16. The graph shows the allocation of the funds for Senior Design Team 3.

The project costs a total of \$1,782.17 out of a \$2,000 budget given to us by Cummins, Inc Marine division. *Figure 16*, shows 89% of the budget was spent to achieve our project goal.

Due to our product being a web based optimization tool we were able code our product on a free downloadable version of python coding program. Though we were able save money on the product we still required a validation tool, and this is where our budget was spent. The most expensive item for the mock keel cooler was the five 4 foot pieces of aluminum piping and the connecting fixtures costing a total of \$856.94 - which was 43% of our budget. The second most expensive item was the water pump costing a total of \$374.23 and 19% of the budget. Additionally, a heater (\$239.00, 12%), Barrel (\$143.00, 7%), Sensors (\$90, 4%), and Reservoir (\$79, 4%) were purchased in order to create our mock keel cooler.

4.8 Schedule

In order to efficiently make use of the time the team has until the end of the semester, a Gantt chart *Figure 17*, created through the use of Microsoft Project has been prepared which lays out a breakdown of the work that needs to be completed. According to the chart, the team has accomplished the required tasks up to the present date and has completed the project. The team has completed the code and is able to evaluate keel coolers for the engines made by Cummins Marine and is able to evaluated both steel and aluminum keel coolers. The scheduling has been successfully managed throughout the project. This was accomplished by setting hard dates for the completion of different parts of the project and consistently sticking to them. The only complication of the schedule that was encountered was due to the slow delivery of parts from vendors, but we were lucky enough to fix the issue and resume our set schedule on our Gantt chart.

4.9 Resource Allocation

4.9.1 Work Breakdown

Each team member contributed to the overall completion of our project through individual tasks. This helped make the work as a whole less daunting, and also aided in staying on schedule. Team members' strengths and weaknesses were taken into account when allocating tasks. Stanko is the most comfortable with programming so he has volunteered to take on the major brunt of the coding. Since Melissa works with our sponsor at Cummins, and can mostly only correspond electronically, she provided the relative equations, principles, and all other data associated with

the design and assessment of keel coolers, aiding in our deliverables, and keeping us in close alignment with our sponsor and his guidelines. Jacob, James, and Grady were best suited for working out the thermodynamics and fluid mechanics associated with the operation of keel coolers and the design of our testing apparatus. This arrangement was optimized for efficiency according to the team dynamic. Sometimes it can be counter-productive for more than one person to be contributing blocks of code for a single program, which is why only Stanko is coded. The other group members provided debugging to help catch any errors and give insight to possible alternative or more efficient scripts. Unlike programming, however, the thermodynamics, fluid mechanics, and also the design for our testing apparatus can be broken down into segments which are more manageable. A better representation of the current team resource allocation can be seen in *Table 3*.

For this project, our group came in under budget with parts. Our group had decided to work together on the hardware portion of the software portion of the project. This was to ensure every group member was up to date on what was expected for the project. The group finished up hardware testing and the software development. Dr. Van Sciver assisted in the project all the way up until his departure for the spring semester. Once Dr. Van Sciver left, the group found help with Dr. Shih with thermal concepts and how to approach the further questions we encountered. To ensure that the group stayed on track for the duration of the project, our group met weekly to discuss work that needed to be done.

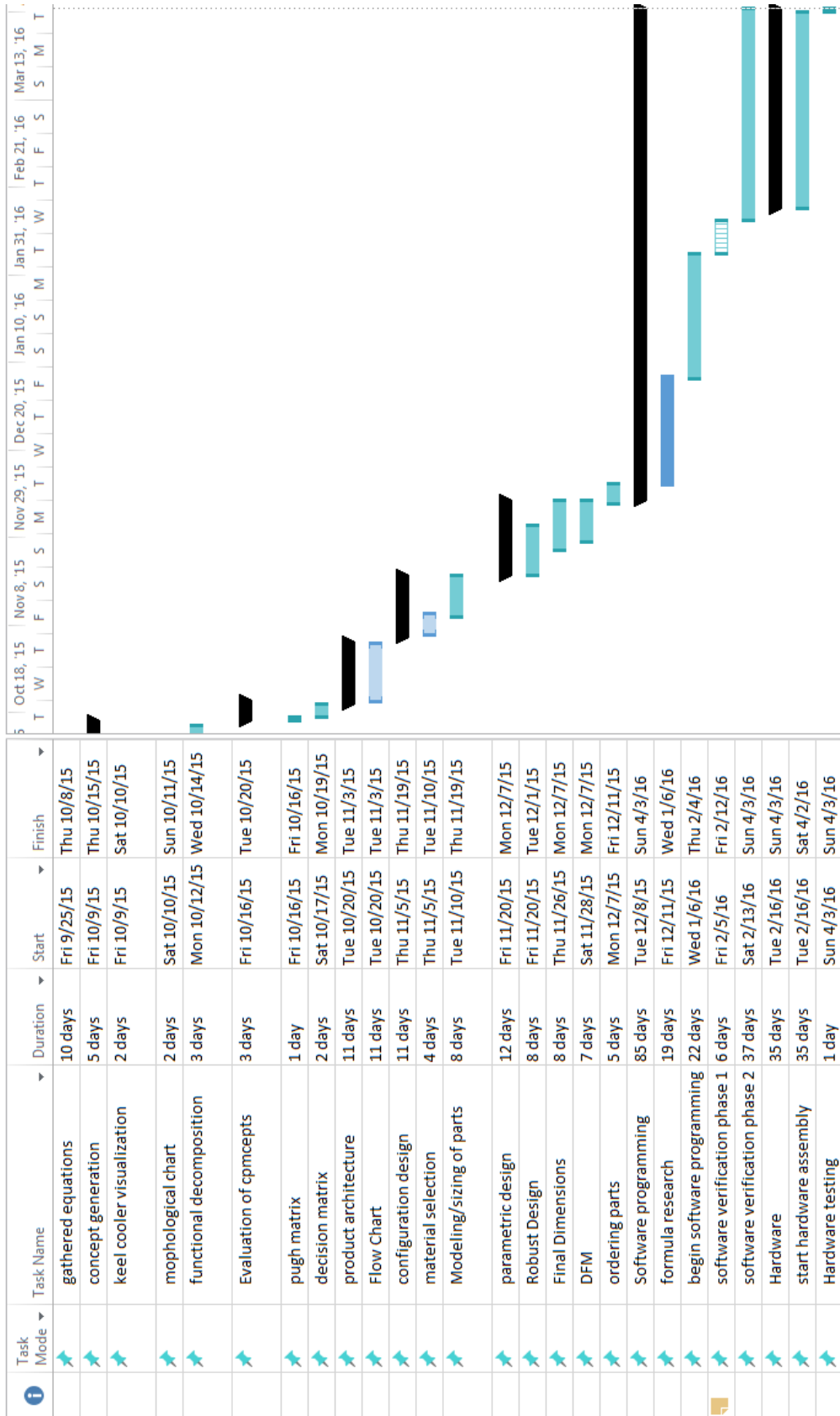


Figure 17: Gantt chart

Table 3: Resource Allocation

| Team Member | Task | Time Allotted |
|--------------------|---|----------------------|
| Melissa Allende | Research design specification for keel cooled systems | 2 weeks |
| | Create flow chart for optimization tool | 1 week |
| | Assist with creation/development of tool | 4 weeks |
| | Secure test vessel for tool validation | Continuous |
| Grady Beasley | Research thermal fluids/relevant equations | 2 weeks |
| | Material acquisition for development of sea channel fabrication | 2 weeks |
| | Assist with fabrication of sea channel | 3 weeks |
| Stanko Gutalj | Research thermal fluids/relevant equations | 2 weeks |
| | Create flow chart for optimization tool | 1 week |
| | Assist with creation/development of tool | 4 weeks |
| James Haga | Research thermal fluids/relevant equations | 2 weeks |
| | Developing webpage | Continuous |
| | Assist with fabrication of sea channel | 3 weeks |
| Jacob Ross | Research thermal fluids/relevant equations | 2 weeks |
| | Assist with creation/development of tool | 4 weeks |

5. Environmental and Safety Issues and Ethics

Team 3 felt that it was of utmost importance that our project be carried out in a safe, ethical, and environmentally-friendly manner. Safety concerns mainly come into play during the construction and testing of the keel cooler test module. The greatest safety concerns for testing the keel cooler model are high temperature fluids running through the keel cooler, and high pressures at which the water pump operates. In order to contain these safety issues, team 3 will maintain a safe distance from the testing apparatus during operation as well as wear proper safety gear. Proper safety gear includes long clothes covering exposed parts of the body and eye protection. Taking environmental issues into concern the keel cooler model will be made out of completely modular parts so that the model can be disassembled and the parts can be repurposed or properly disposed of. Also, the cooling fluid used in the experiment will be water in order avoid disposability and environmental issues that other coolants may pose. Though large amounts of water will be used for testing, the waste will be miniscule because water is a renewable resource and it will not be contaminated with hazardous substances. Ethical choices have been and will continue to be made throughout the entirety of the project. This will be accomplished by following the National Society of Professional Engineers code of ethics.

6. Design of Experiment

6.1 Experimental Procedure

The design and development of the keel cooler optimization tool was split into two main parts: software, and experimental testing. The two sections work together to validate and materialize the theoretical calculations made. The purpose of the experimental testing is to provide physical data to illustrate how various design characteristics of a keel cooler effect the heat dissipation of the proposed design. Since the software evaluates already made designs and suggests new designs, having physical data to validate the calculations made will provide insight into the accuracy of these calculations along with a predicted error.

The experimental setup is designed to showcase how varying the volume flow rate of the heat transfer fluid, the number of flow paths, and the surface area, will affect the overall heat dissipation within a keel cooler. The experiment will be broken down into three main stages in order to test these parameters. The first test will be conducted by keeping a constant configuration of the keel cooler set up and using a controller to vary the voltage (thus the volume flow rate) of the pump. The second test will be conducted by keeping a steady volume flow rate and varying the number of flow paths in the keel cooler by adding or removing the . The third setup keeps the number of pipes *Figure 18* shows a diagram for the experimental setup of the first and second tests.

During the first experiment, the number of pipes are kept constant while the voltage is varied by the controller in order to vary the mass flow rate. The temperature readings at the inlet and the outlet of the device will be measured by two thermocouples in order to calculate the heat dissipated by the keel cooler. This value will be used to calculate the efficiency of the keel cooler for different flow rates.

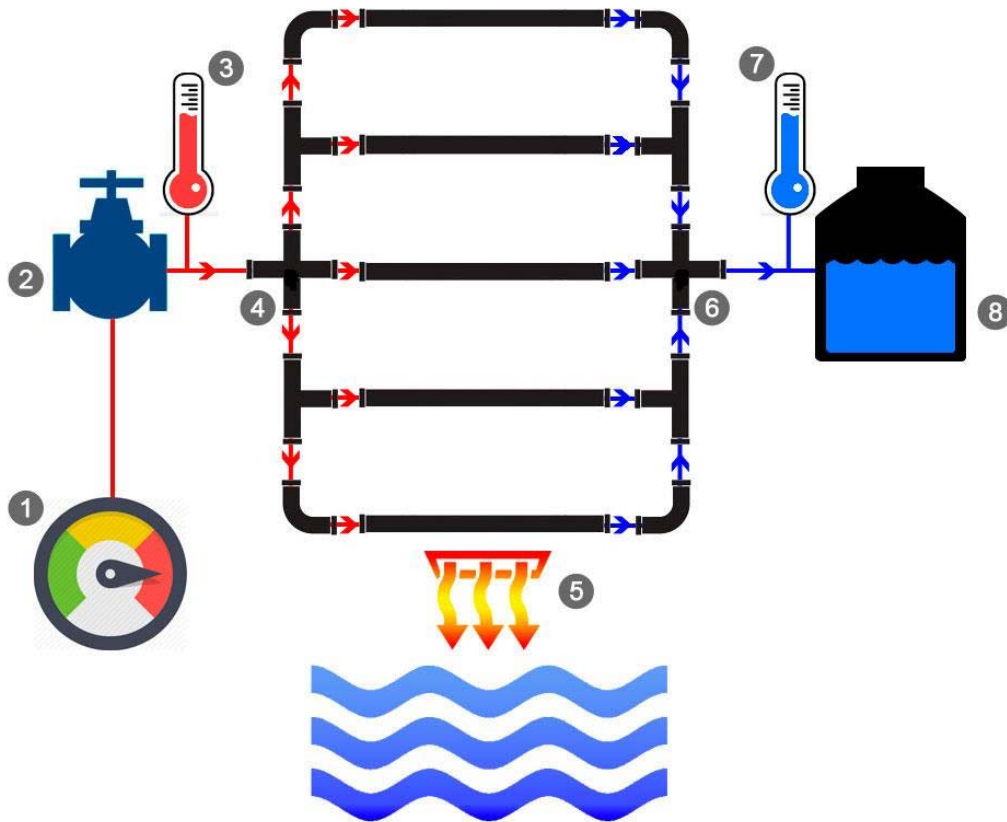


Figure 18: The controller (1) power input to pump (2) to determine the volume flow rate produced. A thermocouple (3) measures the temperature at the inlet of the piping system. The heated fluid enters as a single stream and then diverts into multiple paths (4). The heat is then dissipated from the piping section to the water heat sink (5). The cool fluid converges into a single path (6) where the temperature is then measured by another thermocouple (7). The fluid is exhausted into a container (8).

The second experiment is shown in *Figure 19*. The number of pipes is held at a constant along with the volume flow rate of the pump. During this experiment the connection is changed in order to alter the number of flow paths while keeping surface area constant. During this setup, the flow begins at an inlet and instead of diverging into multiple paths, it follows one continuous path and snakes around. Ultimately, this will examine which configuration works better for a constant surface area.

These experiments will serve as the experimental validation for the improvements suggested by the keel cooler optimization tool. Once a prototype for the software is finalized, its' predictions for heat dissipation can be tested against various configurations of the keel cooler.

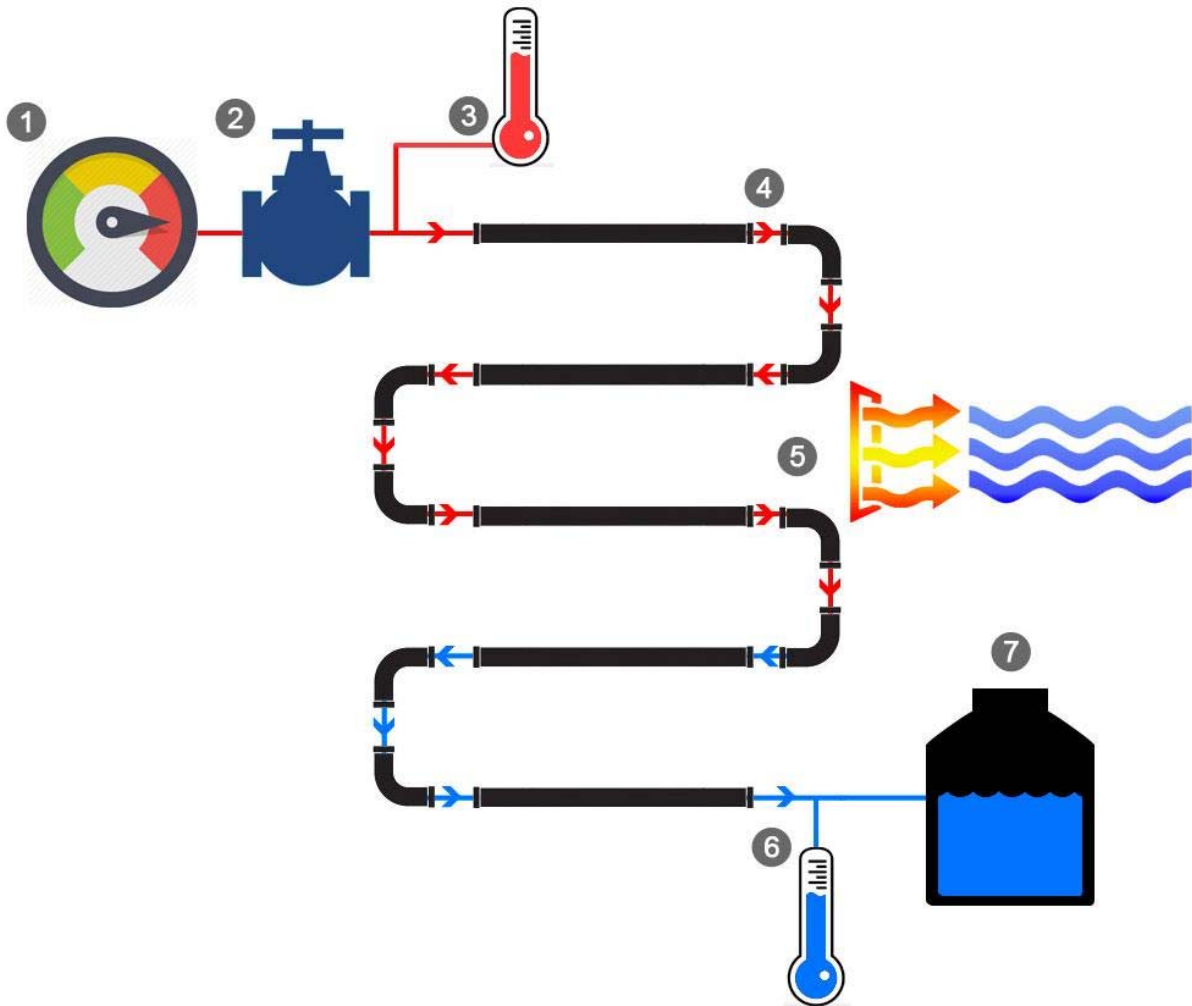


Figure 19: Experimental setup of the five pipe – one flow path configuration. The voltage is set by the controller (1) power input to pump (2). A thermocouple (3) measures the temperature of the heated coolant at the inlet. A single flow path (4) is followed and heat is dissipated to the heat sink (5). A second thermocouple (6) measures the temperature of the cooled fluid which is collected by a container (6).

6.2 Results

The data collected from the hardware during testing can be found in *Figures 20 and 21*. The two sets of data were collected for a multiple flow path configuration and a single flow path configuration respectively.

For the multiple flow setup, the initial temperatures are very similar because the system was not running thus allowing the water in the pipes to converge to nearly the same temperatures. Once the pump was turned on, the temperatures became drastically different since the hot water was introduced to the system. The inlet temperature peaks once the inlet has become as hot as the water in the tank. Once this occurs it then begins to cool down because the heat transfer has begun since the pipes have now heated up and can now begin to release heat from the keel cooler to the water reservoir in which the keel cooler is located. After about 40 seconds, the keel cooler heat dissipation begins to level off. This hints that the keel cooler is reaching steady state which means this is where it will level at. At this point the temperature difference between the inlet and the outlet is about five degrees Celsius. This is a relatively high heat transfer since the cool bath is actually only 72 C and the hot water was measured at 105 C.

Some experimental error that could have occurred in this experiment could be that the pipes we used may not have transferred heat precisely as we had predicted. The hot water in the tank was also supposed to be near 140 C for the experiment since the heater for the barrel was rated to 160 C. This was unclear to the team why it did not get to 140 C since it was on for more than 2 straight days. We were able to get around a 15% temperature drop in the pipes which is near the heat dissipation necessary for the larger ships that have operating temperatures that are higher than the temperatures used in this experiment. Our group was expecting a slightly higher temperature difference but were satisfied with the data recorded for the multiple flow path experiment.

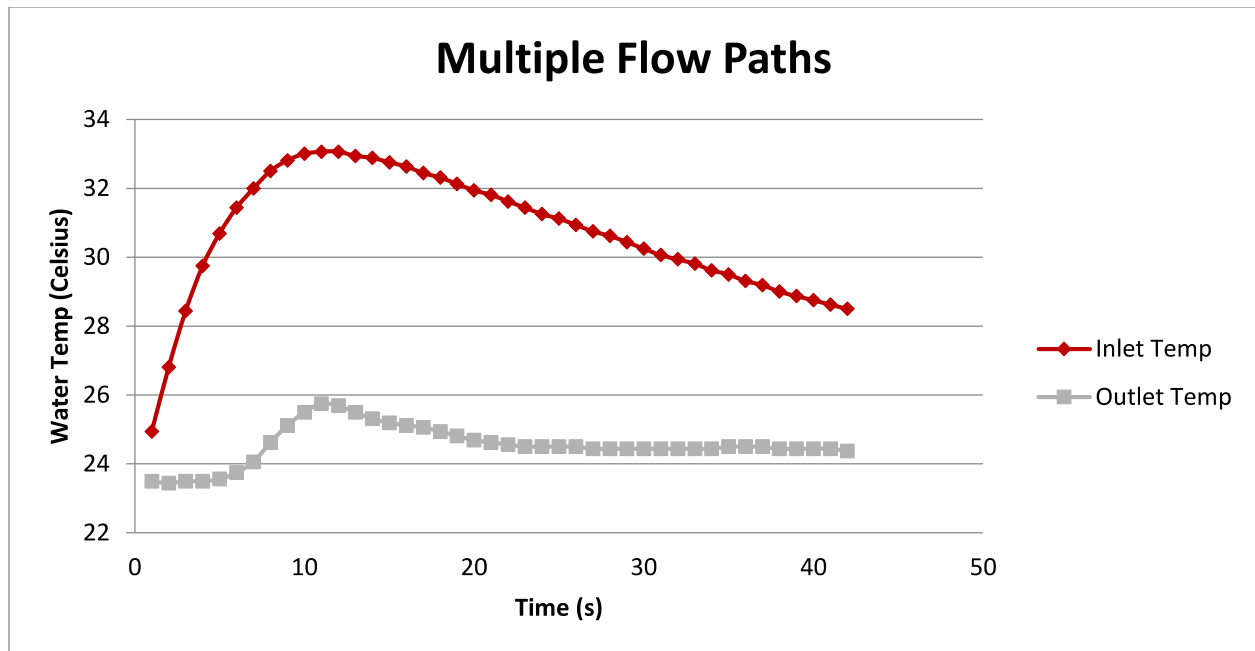


Figure 20: Inlet and Outlet water temperatures of keel cooler with respect to time for a multiple flow path configuration.

For the Single flow path configuration, the test ran exactly the same as the multiple flow path except the keel cooler had a different configuration. The initial temperatures recorded were very similar then slowly grew apart until the system started to heat up. Once the system had heated up and began to dissipate heat, the temperatures slowly started to converge to a more constant temperature difference. Once the steady state region began to be recorded our group ended the experiment.

For this data, the single flow path had a smaller change in temperature from the inlet to the outlet. The steady state temperature difference was about 4 C. This was slightly lower than expected. This could have been due to the slightly lower hot water temperature and the warmer pool water used in the experiment. The percentage in the change in temperature is about 11%, this is slightly lower than necessary but still very close to real world applications. What could have been changed to make this data better could have been a better heating source of the hot water. Our group attempted to heat the water in other ways but was unsuccessful. Our team had looked into other heating sources as well but none were available within the required budget.

Although the single flow path was not exactly as expected, it was still very close to what we needed to get out of it. It offered a very good representation of what happens in a keel cooler and how they behave in an environment where the water around the keel cooler is not in motion.

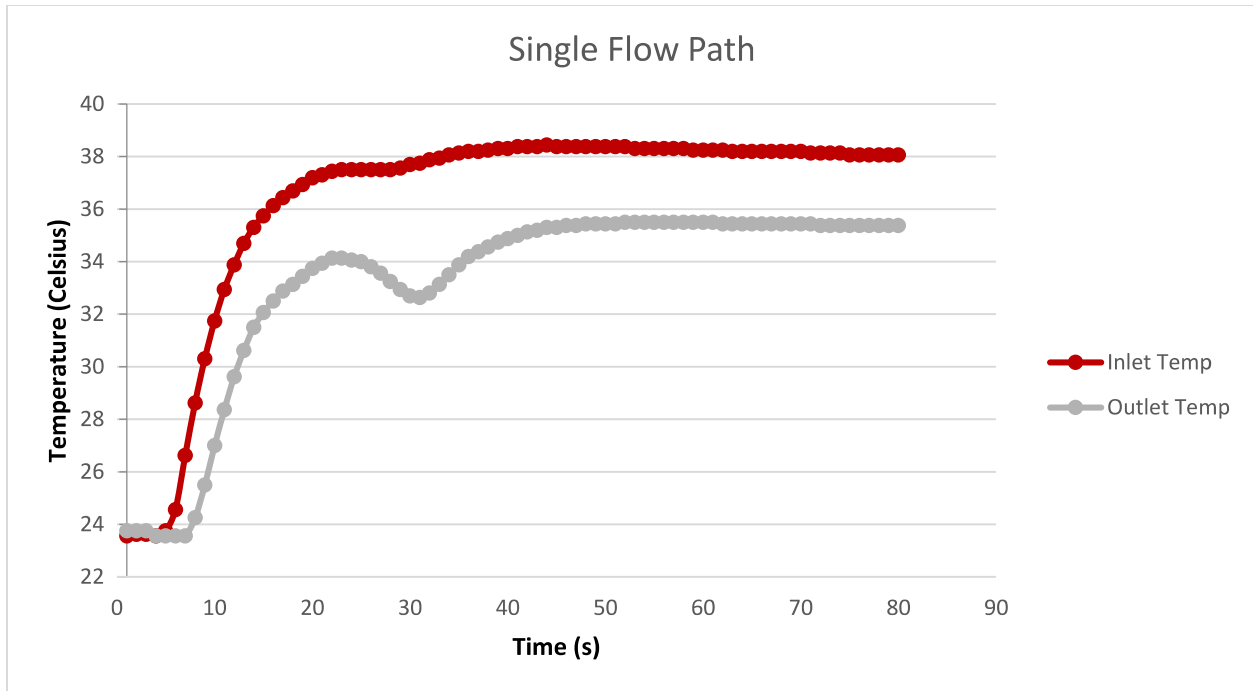


Figure 21: Inlet and Outlet water temperatures of keel cooler with respect to time for a single flow path configuration

6.3 Impact to Industry

The Marine Keel Cooler Optimization tool is expected to be a great aid for the Marine Application Engineers at a global scale. Designed to be a web-based tool, it is readily available around the world at a moment's notice. Making it the ideal tool whether the user is running numbers at an office or at the shipyard itself. The instant feedback allows the user to input numbers and seamlessly reach a result.



Figure 22. Carolina Queen Ferry

This tool was developed to help aid in the design of a keel cooler for a user specified engine given the vessel application. This tool was also developed to validate a current keel cooler in an installed vessel. The team was able to validate the developed tool's accuracy not only with the manufactured keel cooler testing apparatus but was fortunate to validate the tool with an actual installation. The vessel which lent her keel coolers for analysis was the *Carolina Queen*, as shown in *Figure 22*, a passenger ferry vessel which sails in Charleston's harbor. The *Carolina Queen* was facing an engine repower, which required analysis of its current cooling system capabilities.

The case of the *Carolina Queen* is a typical one faced by the Marine Application Engineers at Cummins. This is a vessel which was made in the late 30's with two Detroit Diesel engines. These two engines were being replaced by two QSL9 engines with 404 hp at 2100 RPM. The task was to calculate whether the current keel coolers on the boat would provide substantial cooling for the new engines. If not, what would be required to ensure proper cooling and ensure this was communicated to the shipyard.

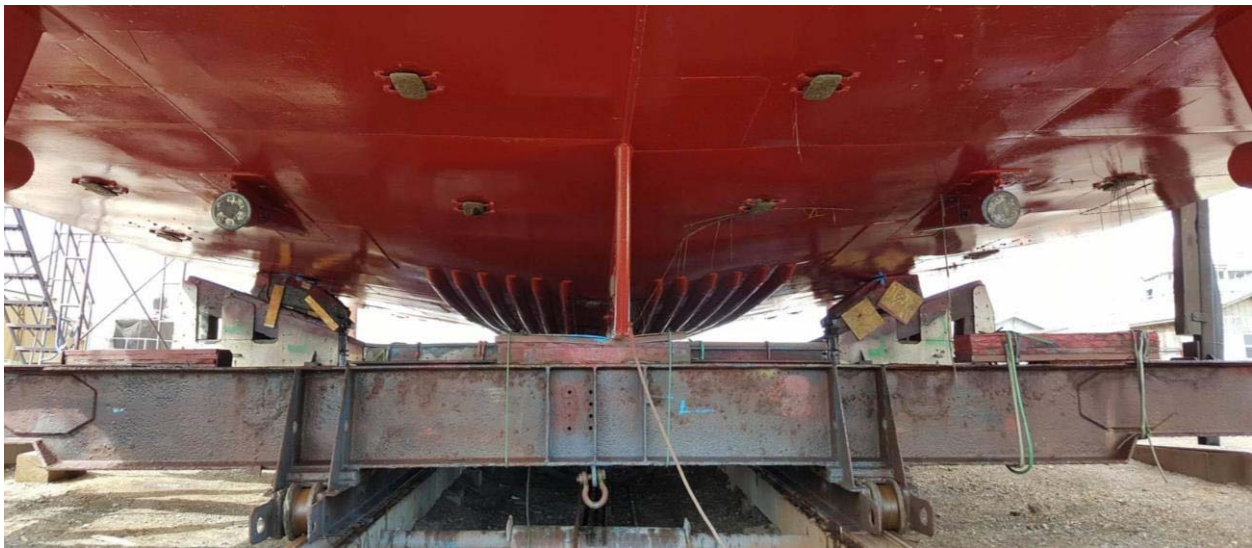


Figure 23. A view of the Carolina Queen's hull with the keel coolers.

The QSL9 engines would provide more power, but the current keel cooler would have to work against a heat rejection to the engine coolant of 15,400 BTU/min. The ship owner had provided current installation data to which the team was able to run through the program, validate the current keel cooler with the current engines and compare it with the QSL9 engines. The team was able to ensure the shipyard moving forward with the new engines would fit the current design of the vessel.

7. Conclusion

The Marine Keel Cooler Optimization tool hopes to meet the needs of Cummins Marine in providing a tool that is up to date, user-friendly, and reliable. The current tool utilized by Cummins Marine was commissioned in the early 1980's, is limited in its ease of use, and only provides a pass/fail output for the user. Cummins Marine is in need of an updated tool which not only validates a proposed design (pass/fail), but can also provide additional design requirements and specifications that will ensure proper cooling performance in the application environment. The team was tasked with writing a program that will utilize these features and provide accurate results. The group researched general information and implementing the knowledge from thermodynamics, fluid mechanics, and heat transfer in order to successfully achieve the project goal. In addition to writing the program, the team built a testing apparatus that was used to model and evaluate the differences in performance for various design configurations, i.e. number of flow paths, and flow that used to verify the accuracy of the program.

As with an engineering project there are always ways to improve and make the project work more efficiently. Because of this future work for this project could include adding more materials for the keel cooler, the location of the keel cooler on the ships, and include more testing to insure the accuracy of the tool.

Looking back on the completion of our project it is easy to realize a few things that we could have done better. First it would have been beneficial to have conducted more extensive research on the products purchased could have helped limit the error and necessity to make quick fixes as we went. Finally team communication would have increased our efficiency in the ability to get things done in a timely manner which could have been accomplished by having set mandatory meeting weekly. However, the team was able to pull through and finalize and complete the final product.

References

- [1] Shaw, Courtney. "Cummins Marine Propulsion." *Cummins Marine*. Web. 31 Mar.. 2016. <<http://marine.cummins.com/>>.
- [2] "Marine Keel Coolers for Heat Dissipation." *Marine Keel Coolers for Heat Dissipation*. Web. 31 Mar. 2016.
- [3] *Cummins Keel Cooler Sizing Tool*. Computer software. Vers. 2.0. Cummins, n.d.Private Web. 29 Sept. 2016.
- [4] "Code of Ethics." *Code of Ethics*. N.p., n.d. Web. 26 Mar. 2016.

Biography

Stanko Gutalj – *Project Leader*

Stanko Gutalj is currently a Systems Engineer at Siemens Building Technologies as well as a research assistant at the Florida Center for Advanced Aero-Propulsion. Stanko is currently involved in an ongoing research investigation on the control of trailing edge vortices under the guidance of Dr. Louis Cattafesta and Adam Edstrand. His contributions include the design of a structural support column for an NACA0012 airfoil and an automated turntable system for precise angular control for wind tunnel testing.

Melissa Allende – *Technical Liaison*

Melissa Allende is currently her second Co-Op working alongside the Marine Application Engineers at Cummins Inc., while completing her senior year in Mechanical Engineering. Melissa was first introduced to marine diesel engines at Caterpillar Inc. where she as an intern for Caterpillar Inc. alongside the Marine Product Health Engineers. She would later complete her first six month Co-Op with Cummins Inc.

Grady Beasley – *Web Based Technician*

Grady is currently a senior Mechanical Engineering student at the FSU-FAMU College of Engineering. He is on track to graduate in the spring of 2016 with a specialization in aeronautics and a minor in mathematics. Outside of his studies, Grady spends his time doing various hands-on projects, working on anything with a motor, and playing the drums. After graduation he plans to move to the west coast to capitalize on mechanical engineering knowledge and explore entrepreneurial prospects.

Jacob Ross – *Financial Advisor*

Jacob Ross is a senior in Mechanical Engineering at Florida State University. During his time studying engineering, he became more interested in aerodynamics and applications of aerodynamics in modern flight. Thirsting for more knowledge, he began training to become a pilot. While continuing his pilot's training, Jacob also works for Professors at Florida State University as a teaching assistant, after graduation plans to finish his master's degree in mechanical engineering and enter the Aerospace Industry.

James Haga – *Administrative Assistant*

Herman "James" Haga IV was born in Fort Valley, GA in 1993 where he lived on a dairy farm for seven years before moving to St. Augustine, FL in 2000. Now James is a senior in mechanical engineering at Florida State University and plans to graduate in the spring of 2016 with his BS in mechanical engineering. James has held one internship at Northrop Grumman as a liaison engineer in the summer of 2013. While in the internship, James gained knowledge of E-2D and F-5 aircraft.

Appendix A: Code

MANAGE.PY

```
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "FirstBlog.settings")

    from django.core.management import execute_from_command_line

    execute_from_command_line(sys.argv)
#WRITTEN BY: STANKO GUTALJ
```

SETTINGS.PY

```
"""
Django settings for FirstBlog project.

Generated by 'django-admin startproject' using Django 1.9.1.

For more information on this file, see
https://docs.djangoproject.com/en/1.9/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.9/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.9/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'by#7x@321f3$r@((2#t@d0_d@wb6shikow6vf{s-0+@jn_don('

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'FirstBlog',
    'blog',
]

MIDDLEWARE_CLASSES = [
    # 'django.middleware.security.SecurityMiddleware',
    # 'django.contrib.sessions.middleware.SessionMiddleware',
    # 'django.middleware.common.CommonMiddleware',
    # 'django.middleware.csrf.CsrfViewMiddleware',
    # 'django.contrib.auth.middleware.AuthenticationMiddleware',
    # 'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    # 'django.contrib.messages.middleware.MessageMiddleware',
    # 'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'FirstBlog.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'FirstBlog.wsgi.application'

# Database
# https://docs.djangoproject.com/en/1.9/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'data.sqlite3',
        'USER': '',
```

```
'PASSWORD': "",
'PORT': "",
#NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
}
}

# Password validation
# https://docs.djangoproject.com/en/1.9/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/1.9/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.9/howto/static-files/

STATIC_URL = '/static/'
#WRITTEN BY: STANKO GUTALJ

"""FirstBlog URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/1.9/topics/http/urls/
Examples:
```

Function views

1. Add an import: `from my_app import views`
2. Add a URL to `urlpatterns`: `url(r'^$', views.home, name='home')`

Class-based views

1. Add an import: `from other_app.views import Home`
2. Add a URL to `urlpatterns`: `url(r'^$', Home.as_view(), name='home')`

Including another URLconf

1. Import the `include()` function: `from django.conf.urls import url, include`
2. Add a URL to `urlpatterns`: `url(r'^blog/', include('blog.urls'))`

```

"""
from django.conf.urls import patterns, include, url
from blog import views

urlpatterns = patterns("",

    url(r'^$', 'blog.views.home', name='home'),
    url(r'^index.html', 'blog.views.index', name='index'),
    url(r'^contributors.html', 'blog.views.contributors', name='contributors'),
    url(r'^documents.html', 'blog.views.docs', name='docs'),
    url(r'^contact.html', 'blog.views.contact', name='contact'),
    url(r'^keelcooler.html$', 'blog.views.keelcooler', name='keelcooler'),
    url(r'^result.html$', 'blog.views.formview', name='formview'),
    url(r'^demo_form.asp', 'blog.views.formr', name='formr'),
)

#WRITTEN BY:STANKO GUTALJ

```

WSGI.PY

```

"""
WSGI config for FirstBlog project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/1.9/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "FirstBlog.settings")

application = get_wsgi_application()

#WRITTEN BY: STANKO GUTALJ

```

VIEWS.PY

```

from django.shortcuts import render_to_response, render
from blog.models import posts
from .forms import CommentForm
import json
from django import forms
from math import sqrt, acos, log

def QSK19(flowpaths ):
    Qout= 27273/flowpaths #heat dissipation per flow path BTU/min
    Tin=120 #temperature of coolant into keel cooler in deg False
    Volume_rate=82/448.833/flowpaths #volumetric flow rate in ft^3/s
    QSK19_results=[Qout,Tin,Volume_rate]
    return(QSK19_results)

def QSK38(flowpaths ):#AFTERCOOLER LTA CIRCUIT
    Qout= 17647/flowpaths #heat dissipation per flow path BTU/min
    Tin=120 #temperature of coolant into keel cooler in deg False
    Volume_rate=158/448.833/flowpaths #volumetric flow rate in ft^3/s
    QSK38_results=[Qout,Tin,Volume_rate]
    return(QSK38_results)

def QSK50(flowpaths ):#AFTERCOOLER LTA CIRCUIT
    Qout= 24122/flowpaths #heat dissipation per flow path BTU/min
    Tin=120 #temperature of coolant into keel cooler in deg False
    Volume_rate=138/448.833/flowpaths #volumetric flow rate in ft^3/s
    QSK50_results=[Qout,Tin,Volume_rate]
    return(QSK50_results)

def QSK60(flowpaths ):#AFTERCOOLER LTA CIRCUIT
    Qout= 35515/flowpaths #heat dissipation per flow path BTU/min
    Tin=120 #temperature of coolant into keel cooler in deg False
    Volume_rate=144/448.833/flowpaths #volumetric flow rate in ft^3/s
    QSK60_results=[Qout,Tin,Volume_rate]
    return(QSK60_results)

def Resistance1(area,bf,tw,tf,d,L,Tin,Volume_rate):
    a1=(bf-tw)/12#inside height in feet
    b1=(d-2*tf)/12#inside width in feet
    Dh=4*(a1*b1)/(2*a1+b1)
    As1=(2*a1+b1)*L #inside surface area in ft^2
    Ac=(d*bf-area)
    k1=-0.000002*Tin**2+0.0009*Tin+0.2935#thermal conductivity of water in BTU/hr-F-ft-F
    p1=(-0.00007*Tin**2+0.0009*Tin+62.557)#density in lbm/ft^3
    u1=(0.00002*Tin**2-0.0088*Tin+1.1442)*0.001#absolute viscosity in lbm/ft sec
    v1=u1/p1#kinematic viscosity in ft^2/sec
    Cp1=1 #Specific Heat in Btu/lbmF

```

```

Pr1= Cp1*u1/(k1/60**2) #Prandtl number
V=Volume_rate/(a1*b1)#fluid velocity in ft/s
Re1=V*L/v1 #Reynolds Number
Nu1=0.036*Re1**0.8*Pr1**(1/3)
hc=0.023*k1*(V**0.8)*((u1/p1)**-0.8)*(Dh**(-0.2))
d_eqo=(4*Ac/acos(-1))**0.5
d_eqi=d_eqo-2*tw
R1=d_eqo/d_eqi*0.0005+1/hc
Resistance1_results=[R1,hc,k1,d_eqo,d_eqi]
return(Resistance1_results)

def Resistance2(k_material,L, tw):
    R2=tw/12/(k_material)
    return(R2)

def Resistance3(L,vessel_speed,tw):
    Tw=85#Degrees Farenheit of water
    k2=-0.000002*Tw**2+0.0009*Tw+0.2935#thermal conductivity of water in BTU/hr-F-ft-F
    p2=(-0.00007*Tw**2+0.0009*Tw+62.557)#density in lbm/ft^3
    u2=(0.00002*Tw**2-0.0088*Tw+1.1442)*0.001#absolute viscosity in lbm/ft sec
    k2=-0.000002*Tw**2+0.0009*Tw+0.2935#thermal conductivity of water in BTU/hr-F-ft-F
    v2=u2/p2
    Cp2=1 #Specific Heat in Btu/lbmF
    Pr2= Cp2*u2/(k2/60**2) #Prandtl number
    ReL=0.65*vessel_speed*1.68*L/v2
    Nu2=0.036*ReL**0.8*Pr2**(1/3)
    ho=k2*Nu2/L
    R3=1/ho
    Resistance3_results=[R3,ho]
    return(Resistance3_results)

def channel(Channelsize):
    channel_area=[14.7,11.8,9.96,8.82,7.35,6.09,8.82,7.35,5.88,4.49,5.88,4.41,3.94,5.51,4.04,3.38,4.
    33,3.6,2.87,3.83,3.09,2.4,2.64,1.97,2.13,1.59,1.76,1.47,1.21,10.053,7.036,7.109,5.218,5.927,4.237,4.923,
    3.526,4.009,2.725,3.427,2.41,2.627,1.881,1.982,1.478,1.358,0.965,0.911,0.491]
    channel_area=channel_area[Channelsize]
    channel_depth=[15.000,15.000,15.000,12.000,12.000,12.000,10.000,10.000,10.000,10.000,9.000,
    9.000,9.000,8.000,8.000,8.000,7.000,7.000,7.000,6.000,6.000,6.000,5.000,5.000,4.000,4.000,3.000,3.000,
    3.000,12.000,12.000,10.000,10.000,9.000,9.000,8.000,8.000,7.000,7.000,6.000,6.000,5.000,5.000,4.000,4
    .000,3.000,3.000,2.000,2.000]
    channel_depth=channel_depth[Channelsize]
    channel_bf=
    [3.716,3.520,3.400,3.170,3.047,2.942,3.033,2.886,2.739,2.600,2.648,2.485,2.433,2.527,2.343,2.260,2.29
    9,2.194,2.090,2.157,2.034,1.920,1.885,1.750,1.721,1.584,1.596,1.498,1.410,5.000,4.000,4.250,3.500,4.00
    0,3.250,3.750,3.000,3.500,2.750,3.250,2.500,2.750,2.250,2.250,2.000,1.750,1.500,1.250,1.000]
    channel_bf=channel_bf[Channelsize]
    channel_tf=[0.650,0.650,0.650,0.501,0.501,0.501,0.436,0.436,0.436,0.436,0.413,0.413,0.413,0.3
    90,0.390,0.390,0.366,0.366,0.366,0.343,0.343,0.343,0.320,0.320,0.296,0.296,0.273,0.273,0.273,0.620,0.4
    70,0.500,0.410,0.440,0.350,0.410,0.350,0.380,0.290,0.350,0.290,0.320,0.260,0.290,0.230,0.260,0.200,0.2
    60,0.130]

```

```

channel_tf=channel_tf[Channelsize]
channel_tw=[0.716,0.520,0.400,0.510,0.387,0.282,0.673,0.526,0.379,0.240,0.448,0.285,0.233,0.4
87,0.303,0.220,0.419,0.314,0.210,0.437,0.314,0.200,0.325,0.190,0.321,0.184,0.356,0.258,0.170,0.350,0.2
90,0.310,0.250,0.290,0.230,0.250,0.190,0.210,0.170,0.210,0.170,0.190,0.150,0.190,0.150,0.170,0.130,0.1
70,0.130]
channel_tw=channel_tw[Channelsize]
if Channelsize<29:
    k_material=26.5 #thermal conductivity of Steel BTU/hr-ft-F
else:
    k_material=118.0 #thermal conductivity of Aluminum BTU/hr-ft-F
channel_results=[channel_area,channel_depth,channel_tf,channel_bf,channel_tw,k_material]

return (channel_results)

```

```

class MyForm(forms.Form):
    Coolant = forms.DecimalField(initial = 1.00)
    Material = forms.DecimalField(initial = 1.00)
    Engineselect = forms.DecimalField(initial = 0.00)
    Channelsize = forms.IntegerField(initial = 1.00)
    flowpaths = forms.IntegerField(initial = 1.00)
    kclength = forms.FloatField(initial = 0.00)
    vessel_speed = forms.FloatField(initial = 0.00)

```

```

def home(request):
    return render_to_response("index.html")
stankogut = "0"

```

```

def index(request):
    template = "index.html"
    content= {'stankogut' : '0'}
    return render_to_response(template, content)

```

```

def contributors(request):
    entries = posts.objects.all()[:10]
    return render_to_response('contributors.html', {'posts' : entries})

```

```

def docs(request):
    entries = posts.objects.all()[:10]
    return render_to_response('documents.html', {'posts' : entries})

```

```

def contact(request):
    entries = posts.objects.all()[:10]
    return render_to_response('contact.html', {'posts' : entries})

```

```

def keelcooler(request):

```



```
return render(request, 'keelcooler.html')

def formview(request):
    if request.method == 'POST':
        form = MyForm(request.POST)
        if form.is_valid():
            Coolant = form.cleaned_data['Coolant']
            Material = form.cleaned_data['Material']
            Engineselect = form.cleaned_data['Engineselect']
            Channelsize = form.cleaned_data['Channelsize']
            flowpaths = form.cleaned_data['flowpaths']
            kclength = form.cleaned_data['kclength']
            vessel_speed = form.cleaned_data['vessel_speed']

            if Engineselect==1:
                QSK19_results=QSK19(flowpaths)
                Qout=QSK19_results[0]
                Tin=QSK19_results[1]
                Volume_rate=QSK19_results[2]

            elif Engineselect==2:
                QSK38_results=QSK38(flowpaths)
                Qout=QSK38_results[0]
                Tin=QSK38_results[1]
                Volume_rate=QSK38_results[2]

            elif Engineselect==3:
                QSK50_results=QSK50(flowpaths)
                Qout=QSK50_results[0]
                Tin=QSK50_results[1]
                Volume_rate=QSK50_results[2]

            elif Engineselect==4:
                QSK60_results=QSK60(flowpaths)
                Qout=QSK60_results[0]
                Tin=QSK60_results[1]
                Volume_rate=QSK60_results[2]

            channel_results=channel(Channelsize-1)

            L=kclength

            area=channel_results[0]
            d= channel_results[1]
            bf= channel_results[3]
            tf= channel_results[2]
            tw= channel_results[4]
```

```

k_material= channel_results[5]
R2=Resistance2(k_material,kclength, tw)

Resistance1_results=Resistance1(area,bf,tw,tf,d,kclength,Tin,Volume_rate)
R1=Resistance1_results[0]
hi=Resistance1_results[1]
kw=Resistance1_results[2]
do_eq=Resistance1_results[3]
di_eq=Resistance1_results[4]

Resistance3_results=Resistance3(L,vessel_speed,tw)
R3=Resistance3_results[0]
ho=Resistance3_results[1]

Rf_seawater=0.0005
Rf_coolant=0.001
Cp_eg=-0.0000003*(Tin**2) + 0.0005*(Tin) + 0.7762
U_overall=1/(R3+Rf_seawater+R2+Rf_coolant+R1)

dT_required=
Qout*flowpaths/(Volume_rate*448.833*flowpaths*.1337*64*Cp_eg)
Th=Tin+dT_required
Tc=85#Degrees

dTA=Th-Tc
dTB=Tin-Tc
LMTD=(dTA-dTB)/log(dTA/dTB)
A_required=(Qout*flowpaths*60)/(U_overall*LMTD)
perimeter=(2*bf+d)/12
A_actual=(perimeter)*L*flowpaths

if(A_actual<A_required):
    KCRESULT="FAIL"
    Lnew=A_required/perimeter/flowpaths
    fpnew=A_required/(perimeter*L)
    if((fpnew-int(fpnew))>0):
        fpnew=int(fpnew)+1

    NOTE1="AT THE SAME NUMBER OF FLOW PATHS, THE
REQUIRED KEEL COOLER LENGTH IS %.2f ft." %Lnew
    NOTE2="AT THE SAME LENGTH, THE REQUIRED NUMBER OF
FLOW PATHS IS %.0f" %fpnew

else:
    KCRESULT="PASS"
    NOTE1=" "
    NOTE2=" "

template = "keelcooler.html"
#-----VARIABLE DEFINITION-----
content={

```

```

'Coolant' : Coolant,
'Material' : Material,
'Engineselect' : Engineselect,
'Channelsize' : Channelsize,
'flowpaths' : flowpaths,
'kclength' : kclength,
'Qout' : Qout,
'area' : channel_results[0],
'depth' : channel_results[1],
'bf' : channel_results[3],
'tf' : channel_results[2],
'tw' : channel_results[4],
'k_material' : channel_results[5],
'R1' : R1,
'hi' : hi,
'kw' : kw,
'do_eq' : do_eq,
'di_eq' : di_eq,
'R2' : R2,
'R3' : R3,
'ho' : ho,
'U_overall' : U_overall,
'Temp_increment' : dT_required,
'LMTD' : LMTD,
'A_required' : A_required,
'A_actual' : A_actual,
'NOTE1' : NOTE1,
'NOTE2' : NOTE2,
'KCRESLT' : KCRESLT,
}

```

```
#-----
```

```
#-----CALCULATIONS-----
```

```
#-----
```

```

else:
    template = "keelcooler.html"
    content={
        'message': "[error]"
    }
return render(request, template, content)

```

```
#WRITTEN BY: STANKO GUTALJ
```

KEELCOOLER.HTML

```

        {% load staticfiles %}
        <link rel="stylesheet" type="text/css" href="{% static 'blog/bootstrap.css' %}" />
        <link rel="stylesheet" type="text/css" href="{% static 'blog/bootstrap-responsive.css' %}" />
        <link rel="stylesheet" type="text/css" href="{% static 'blog/bootstrap.css' %}" />
        <link rel="stylesheet" type="text/css" href="{% static 'blog/bootstrap-responsive.min.css' %}" />
        <link rel="stylesheet" type="text/css" href="{% static 'blog/carousel.css' %}" />

        <!DOCTYPE html>
        <html lang="en">
        <head>
        <meta charset="utf-8">
        <title>Senior Design | Cummins</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta name="description" content="">
        <meta name="author" content="">

        <!-- Le styles -->
        <link href="../assets/css/bootstrap.css" rel="stylesheet">
        <link href="../assets/css/bootstrap-responsive.css" rel="stylesheet">
        <link href="css/bootstrap.min.css" rel="stylesheet" media="screen">

        <!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
        <!--[if lt IE 9]>
        <script src="../assets/js/html5shiv.js"></script>
        <![endif]-->

        <!-- Fav and touch icons -->
        <link rel="apple-touch-icon-precomposed" sizes="144x144" href="../assets/ico/apple-touch-icon-144-
        precomposed.png">
        <link rel="apple-touch-icon-precomposed" sizes="114x114" href="../assets/ico/apple-touch-icon-114-
        precomposed.png">
        <link rel="apple-touch-icon-precomposed" sizes="72x72" href="../assets/ico/apple-touch-icon-72-
        precomposed.png">
        <link rel="apple-touch-icon-precomposed" href="../assets/ico/apple-touch-icon-57-
        precomposed.png">
        <link rel="shortcut icon" href="../assets/ico/favicon.png">
        </head>

        <body>

        <!-- NAVBAR
        ===== -->
        <div class="navbar-wrapper">
        <!-- Wrap the .navbar in .container to center it within the absolutely positioned parent. -->
        <div class="container">

```

```

        <div class="navbar navbar-inverse">
            <div class="navbar-inner">
<!-- Responsive Navbar Part 1: Button for triggering responsive navbar (not covered in tutorial).
            Include responsive CSS to utilize. -->
            <button type="button" class="btn btn-navbar" data-toggle="collapse" data-target=".nav-
                collapse">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="brand" href="#">Keel Cooler Optimization Tool</a>
<!-- Responsive Navbar Part 2: Place all navbar contents you want collapsed withing .navbar-
                collapse.collapse. -->
            <div class="nav-collapse collapse">
                <ul class="nav">
                    <li><a href="index.html">Home</a></li>
                    <li><a href="contributors.html">Meet The Team</a></li>
                    <li><a href="contact.html">Contact</a></li>
                    <li><a href="documents.html">Documents</a></li>
                    <li class="active"><a href="keelcooler.html">Keel Cooler Tool
                        1.0</a></li>
                    <!-- Read about Bootstrap dropdowns at
                        http://twbs.github.com/bootstrap/javascript.html#dropdowns -->
                </ul>
            </div><!-- /.nav-collapse -->
        </div><!-- /.navbar-inner -->
    </div><!-- /.navbar -->

</div> <!-- /.container -->
</div><!-- /.navbar-wrapper -->

```

```

<!-- Carousel

```

```

===== -->
<div id="myCarousel" class="carousel slide">
    <div class="carousel-inner">
        <div class="item active">
            
            <div class="container">
                </div>
            </div>
            <div class="item">
                
                <div class="container">
                    <div class="carousel-caption">
                        <h1></h1>
                        <p class="lead"></p>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        </div>
    </div>
    <div class="item">
    
    <div class="container">
    <div class="carousel-caption">
    <h1></h1>
    <p class="lead"></p>
    </div>
    </div>
    </div>
    </div>
    </div>
    <a class="left carousel-control" href="#myCarousel" data-slide="prev">&lsaquo;</a>
    <a class="right carousel-control" href="#myCarousel" data-slide="next">&rsaquo;</a>
    </div><!-- /.carousel -->

```

```

    <!-- Marketing messaging and featurettes
    ===== -->
    <!-- Wrap the rest of the page in another container to center all the content. -->

```

```

    <div class="container marketing">

```

```

    <!-- Marketing messaging and featurettes
    ===== -->
    <!-- Wrap the rest of the page in another container to center all the content. -->

```

```

    <!-- Three columns of text below the carousel -->

```

```

    <!-- START THE FEATURETTES

```

```

    <hr class="featurette-divider">-->

```

```

        <div class="row">
        <form action="result.html" method="POST"> {% csrf_token %}
            <p>
                <div class="ui-widget">
                    <h1>Welcome to the Keel Cooler Tool 1.0</h1>
                    <p>Please select an engine, coolant, material, channel size and
flowpaths to receive the proper measurement of heat dissipation.</b>
                    <br><br>
                    <label for="id_Engineselect"><h4>Engine
Selection:</h4></label>
                </div>
            </p>
        </div>
    </div>
    <center>

```

```

        <input id="id_Engineselect" type="radio"
name="Engineselect" value=1 checked> QSK19 
        <input id="id_Engineselect" type="radio"
name="Engineselect" value=2> QSK38 
        <input id="id_Engineselect" type="radio"
name="Engineselect" value=3> QSK50 
        <input id="id_Engineselect" type="radio"
name="Engineselect" value=4> QSK60 
        </center>
    </div>

    <br>
    <label for="id_Coolant"><h4>Coolant: </h4></label>
    <select for="id_Coolant" name="Coolant">
        <option id="id_Coolant" value=1>50/50
        Glycol</option>
        <option id="id_Coolant"
value=2>Water/DCA</option>
    </select>
    <br>
    <label for="id_Material"><h4>Channel Material: </h4></label>
    <select for="id_Material" name="Material">
        <option id="id_Material" value=1>Steel</option>
        <option id="id_Material"
value=2>Aluminum</option>
    </select>
    <br>
    <label for="id_Channelsize"><h4>Channel Size: </h4></label>
    <select for="id_Channelsize" name="Channelsize">
        <optgroup label="Steel">
            <option id="id_Channelsize" value=1>C15 ×
            50</option>
            <option id="id_Channelsize"
value=2>C15 × 40</option>
            <option id="id_Channelsize"
value=3>C15 × 33.9</option>
            <option id="id_Channelsize"
value=4>C12 × 30</option>
            <option id="id_Channelsize"
value=5>C12 × 25</option>
            <option id="id_Channelsize"
value=6>C12 × 20.7</option>
            <option id="id_Channelsize"
value=7>C10 × 30</option>
            <option id="id_Channelsize"
value=8>C10 × 25</option>
            <option id="id_Channelsize"
value=9>C10 × 20</option>
            <option id="id_Channelsize"
value=10>C10 × 15.3</option>

```

| | |
|-------------------------------|-----------------------------|
| value=11>C9 × 20</option> | <option id="id_Channelsize" |
| value=12>C9 × 15</option> | <option id="id_Channelsize" |
| value=13>C9 × 13.4</option> | <option id="id_Channelsize" |
| value=14>C8 × 18.75</option> | <option id="id_Channelsize" |
| value=15>C8 × 13.75</option> | <option id="id_Channelsize" |
| value=16>C8 × 11.5</option> | <option id="id_Channelsize" |
| value=17>C7 × 14.75</option> | <option id="id_Channelsize" |
| value=18>C7 × 12.25</option> | <option id="id_Channelsize" |
| value=19>C7 × 9.8</option> | <option id="id_Channelsize" |
| value=20>C6 × 13</option> | <option id="id_Channelsize" |
| value=21>C6 × 10.5</option> | <option id="id_Channelsize" |
| value=22>C6 × 8.2</option> | <option id="id_Channelsize" |
| value=23>C5 × 9</option> | <option id="id_Channelsize" |
| value=24>C5 × 6.7</option> | <option id="id_Channelsize" |
| value=25>C4 × 7.25</option> | <option id="id_Channelsize" |
| value=26>C4 × 5.4</option> | <option id="id_Channelsize" |
| value=27>C3 × 6</option> | <option id="id_Channelsize" |
| value=28>C3 × 5</option> | <option id="id_Channelsize" |
| value=29>C3 × 4.1</option> | <option id="id_Channelsize" |
| | <optgroup label="Aluminum"> |
| value=30>12 × 11.822</option> | <option id="id_Channelsize" |
| value=31>12 × 8.274</option> | <option id="id_Channelsize" |
| value=32>10 × 8.36</option> | <option id="id_Channelsize" |
| value=33>10 × 6.136</option> | <option id="id_Channelsize" |
| value=34>9 × 6.97</option> | <option id="id_Channelsize" |


```

value=35>9 × 4.983</option>
value=36>8 × 5.789</option>
value=37>8 × 4.147</option>
value=38>7 × 4.715</option>
value=39>7 × 3.205</option>
value=40>6 × 4.03</option>
value=41>6 × 2.834</option>
value=42>5 × 3.089</option>
value=43>5 × 2.212</option>
value=44>4 × 2.331</option>
value=45>4 × 1.738</option>
value=46>3 × 1.597</option>
value=47>3 × 1.135</option>
value=48>2 × 1.071</option>
value=49>2 × 0.577</option>
</select>
<label for="id_flowpaths"><h4>Number of
Flow Paths: </h4> </label>
<input id="id_flowpaths" type="number" name="flowpaths"
min="1" max="20" value="3" style="height:30px" />
<label for="id_kclength"><h4>Installed Keel Cooler Length:
</h4> </label>
<input id="id_kclength" type="float" name="kclength" min="0"
value="100" style="height:30px" />
<label for="id_vessel_speed"><h4>Vessel Speed (knots): </h4>
</label>
<input id="id_vessel_speed" type="float" name="vessel_speed"
min="0" value="3" style="height:30px" />
<br><br><input type='submit' value='SUBMIT' class
="btn' />
<div="blank" style="color:red;">{{message}}</div>
<br><h3>RESULTS: </h3>
<p> Qout = {{ Qout }} </p>
<p> Area = {{area}} </p>
<p> Depth = {{depth}} </p>
<p> bf = {{bf}} </p>
<p> tf = {{tf}} </p>

```

```

        <p> tw = {{tw}} </p>
        <p> k_material = {{k_material}} </p>
        <p> hi = {{hi}} </p>
        <p> R1 = {{R1}} </p>
        <p> kw = {{kw}} </p>
        <p> do_eq = {{do_eq}} </p>
        <p> di_eq = {{di_eq}} </p>
        <p> R2 = {{R2}} </p>
        <p> R3 = {{R3}} </p>
        <p> ho = {{ho}} </p>
        <p> Uoverall = {{U_overall}} </p>
        <p> Temp Increment = {{Temp_increment}} </p>
        <p> LMTD = {{LMTD}} </p>
        <p> Area Required = {{A_required}} </p>
        <p> Area Actual = {{A_actual}} </p>
        <br><h3>(PASS/FAIL): {{ KCRESULT }}</h3>
        <p>{{ NOTE1 }} </p>
        <p>{{ NOTE2 }} </p>
        </div>
    </p>
</form>

</body>
</div>
<!-- /END THE FEATURETTES -->
<!-- FOOTER -->
<footer>
    <p class="pull-right"><a href="#">Back to top</a></p>
    <center>
        <p>&copy; 2016 FSU Senior Design Team 3 | Cummins, Inc.</p>
    </center>
</footer>

</div><!-- /.container -->
<!-- Le javascript
===== -->
<!-- Placed at the end of the document so the pages load faster -->
    <script src="../assets/js/jquery.js"></script>
    <script src="../assets/js/selection.js"></script>
    <script src="../assets/js/bootstrap-transition.js"></script>
    <script src="../assets/js/bootstrap-alert.js"></script>
    <script src="../assets/js/bootstrap-modal.js"></script>
    <script src="../assets/js/bootstrap-dropdown.js"></script>
    <script src="../assets/js/bootstrap-scrollspy.js"></script>
    <script src="../assets/js/bootstrap-tab.js"></script>
    <script src="../assets/js/bootstrap-tooltip.js"></script>
    <script src="../assets/js/bootstrap-popover.js"></script>
    <script src="../assets/js/bootstrap-button.js"></script>
    <script src="../assets/js/bootstrap-collapse.js"></script>
    <script src="../assets/js/bootstrap-carousel.js"></script>
    <script src="../assets/js/bootstrap-typeahead.js"></script>

```

```

        <script>
        !function ($) {
            $(function(){
                // carousel demo
                $('#myCarousel').carousel()
            })
        }(window.jQuery)
        </script>
        <script src="../assets/js/holder/holder.js"></script>
        <script src="http://code.jquery.com/jquery.js"></script>
        <script src="js/bootstrap.min.js"></script>
        <script src="{% static 'blog/bootstrap.min.js' %}"></script>
        </body>
        </html>

```

URLS.PY

"""FirstBlog URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

<https://docs.djangoproject.com/en/1.9/topics/http/urls/>

Examples:

Function views

1. Add an import: from my_app import views
2. Add a URL to urlpatterns: url(r'^\$', views.home, name='home')

Class-based views

1. Add an import: from other_app.views import Home
2. Add a URL to urlpatterns: url(r'^\$', Home.as_view(), name='home')

Including another URLconf

1. Import the include() function: from django.conf.urls import url, include
2. Add a URL to urlpatterns: url(r'^blog/', include('blog.urls'))

"""

```

from django.conf.urls import patterns, include, url
from blog import views

```

```

urlpatterns = patterns("",

```

```

    url(r'^$', 'blog.views.home', name='home'),
    url(r'^index.html', 'blog.views.index', name='index'),
    url(r'^contributors.html', 'blog.views.contributors', name='contributors'),
    url(r'^documents.html', 'blog.views.docs', name='docs'),
    url(r'^contact.html', 'blog.views.contact', name='contact'),
    url(r'^keelcooler.html$', 'blog.views.keelcooler', name='keelcooler'),
    url(r'^result.html$', 'blog.views.formview', name='formview'),
    url(r'^demo_form.asp', 'blog.views.formr', name='formr'),

```