Software Design Document Final Draft for Team “Fish Out of Water”
2/16/2024

Project Sponsor: John Fennell
Faculty Mentor: Saisri Muttineni

Team Members:
Jack Shanley
Corey Moreno
Nicholas Robishaw
Jaron Bauers

Table of Contents

Introduction

Deep within a canyon on the border of Northern Arizona and Southern Utah resides the rainbow trout. Anglers from all over come to Lee's Ferry, a well-known name in the trout fishing world, to board a boat onto the Colorado River with hopes of catching these trout. The Arizona Game and Fish Department has to keep a watchful eye over the fishing that happens at Lee's Ferry to ensure that the trout population maintains a healthy number. The rainbow trout is an integral element in the ecosystem of the Colorado River, thus it is important to support the conservation of their species as well as other species of fish that are there.

Implementation of a monitoring program has been a bit challenging. Arizona Game and Fish (AZGFD) sends representatives out a few times a month to interview anglers in person about their experience on the water. They ask them many questions, including how long they were out and how many fish they caught. But the reality is that there is not enough time nor resources for the AZGFD to send representatives out there for them to get an accurate depiction of the environment that would allow them to gain insight into the status of the trout. In order to promote conservation for the trout, they need more data. In an effort to solve this problem, Arizona Game and Fish installed a game camera not too far from the dock to capture images of the dock every thirty seconds. This results in over 1,800 photos taken every day. The addition of the camera has helped our client, John Fennell, and his team procure data for their research and analysis. The pictures contain boats and kayaks, which can give them a better idea of how much traffic the river is seeing and how much fishing is going on. But there is still a problem, John and his team have to manually look at all of those photos to pick out the ones they need. Naturally, this is a very tedious task, as it can take over five hours just to get through one day's worth of photos.

Main Issues:

- Processing their photos manually takes too much time due to the large amount of photos.
- Manually Categorizing these photos also contributes to why this is such a time consuming process for their team.

Our Solution:

- Automate the sorting process of the thousands of images that they go through on a monthly basis by utilizing a Python library called OpenCV2 and an algorithm called YOLO, an object detection model, that will allow us to implement image recognition software within their system.
- Significantly cut down the amount of photos that need to be manually processed by John and his team by pushing the photos through the sorting process that will deposit them into specific folders structured on the requirements of John's team.
- User has access to photos in case there are instances in which manual review is required or desired.
- All of this will run through a simple and effective GUI system as a desktop application.

Implementation Overview

Big Picture - Make a program to sort through and document information on a massive amount of input photos. In this case the program will sort through a folder of 1800+ pictures and identify if there is a boat in the water of the picture. The program will document the date, time, etc from the photo and add it to an excel spreadsheet for the customer to input into their database. Other functions inside the program will identify if the photo is valid meaning it has a boat or if its junk/redundant. This way the customer can save time and energy when studying the boat data at the Lee Ferry sight.

Technology planned

- Python will be the formal programming language used to achieve success in this program due to its vast libraries and versatility in image processing/recognition.
- OpenCV2 will identify and box out pictures of boats.
- YOLO library extension will aid OpenCV2 to become a bit more autonomous so the user does not have to force feed all the photos of boats to teach the program. The following video link demonstrates the process of how to input training data for YOLOV8: https://www.youtube.com/watch?v=m9fH9OWn8YM

Solution vision - Cut down the time it takes to analyze and collect data drastically. Create a program that can navigate through pictures and judge whether conditions are met to either pass

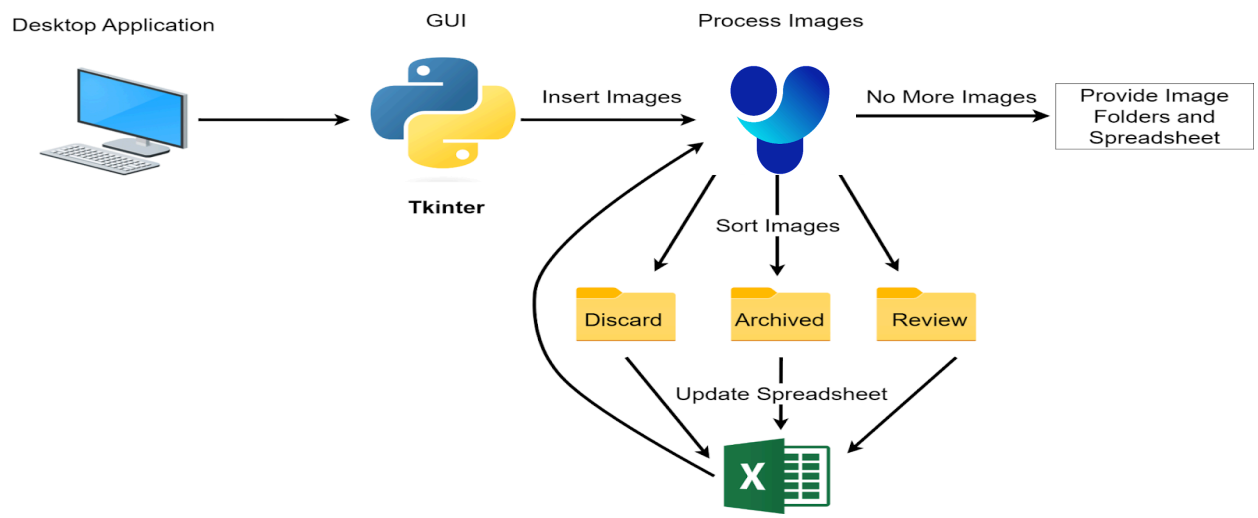
or fail a picture. Make John's life easier and allow him to spend more time doing other more important things at work.

Overall approach - Attempting to implement a divide and conquer approach to first divide all pictures down essentially filtering out the junk and redundant photos. This way we can simplify the functions used and make them smaller and easier to maintain. Possibly making multiple functions call themselves recursively to save time and resources. Then once all the junk photos are taken out the program can go through and combine the information from all the leftover photos.

Roles of the team members

- Jack (Team lead/coder) - The team lead is responsible for ensuring that all team members have been assigned tasks and are completing them as we have planned. The team lead will ensure that all team members have the support they need to successfully complete high-quality work. Will be coding specific parts of the program.
- Corey (Architect/coder) - The architect is responsible for ensuring that the core architectural decisions that have been made are followed. The architect will also be in charge of overseeing the design sequence and logic for the project, and make necessary adjustments as needed. Will be coding specific parts of the program.
- Jaron (Customer Communicator/coder) - The customer communicator is in charge of contacting the customer with any questions, meeting times, etc. If there are any immediate questions involving key implementation concerns, the customer communicator will reach out to get clarification. Will also contribute code to the project as well as review teammates' pull requests.
- Nick (Release manager/coder) - The release manager is responsible for reviewing all major changes prior to releasing to the main branch. Will be coding specific parts of the program.

Architecture Overview



(Figure 1 - Control Flow, this figure demonstrates the flow of how this program will execute)

The system for this project will primarily be built using Python, the OpenCV2 Python module, the YOLO Python module, Tkinter, and the Pandas Python module. Each of these modules will provide structure for the process behind the image sorting methods.

OpenCV2 and YOLO are Python modules that are used for image processing and handling as well as object detection. Both of these modules will have to be trained with test data in order for them to understand the objects we are looking for in each image (boat, kayak, raft, etc.). In order to do this, we will take multiple days worth of images provided by Lee's Ferry and cut out example images of boats, kayaks, and rafts to feed them into these libraries. OpenCV2 and YOLO will translate these images to machine code for the program to refer to when looking for specific images.

The Pandas module is important for exporting our data to an Excel spreadsheet format. Pandas is a tool that can take any datatype from the Python code to format and then export that data into the desired spreadsheet format. We will use this module to export important metadata from the photos such as boat count, time, date, error flags, etc.

Tkinter is a Python module that aids in creating the GUI of a program. Instead of having our client learn how to run a Python program in the command line of their terminal, we want to design a GUI that is user friendly and can display the correct and desired outputs.

Key Functions and Description

find_boat(haystack, needle):

This function takes in the photo that will be searched for objects (aka the Haystack Photo) and will search the photo for the desired boat, kayak, or raft using YOLO (aka the Needle). If there is no needle found in the haystack, then the photo will be sent to the discard folder. If needles are found in the haystack image, the image will first be sent to the archive folder, then the object count will be incremented, and then the metadata will be stored to later output onto the spreadsheet. This process will continue until there are no photos left.

compare_haystacks(photo, compareTo):

Sometimes, there are photos that are very similar in consecutive instances. To shorten the image processing time, we will discard photos that seem similar to other photos. This function will take the current photo and compare it to the next photo in line. If the photos are too similar, the next photo in line will be sent to the discard folder. Otherwise, the photos will be put into the archive folder for the find_boat function. This process will continue until the next incoming photo is significantly different.

write_to_excel(filename, data):

Our client asked us to output any significant data into the form of an Excel spreadsheet. Using the Pandas module, once all of the data from a day's worth of photos are gathered, we will have Pandas export this data into the desired format in an Excel file format for our client to store.

create_gui():

Using Tkinter, we will create a simple GUI for our client to use. Our GUI will have the option for the user to drop in the folder with all of the images or the option to type in their folder’s file path. Once they place their desired folder, all they will have to do is press the ‘Run’ button, and the program will take care of the rest.

get_image_time(photo):

In every photo, there is a timestamp and date on the top. Using OpenCV2, we can have the module read the top of the photo and save the date and time to later export to the spreadsheet.

sort_images(photo):

Instead of giving our client back every single photo, we found that it would be best to send each photo to a different folder depending on its usability. If the photo is a duplicate photo, it will get sent to a discard folder. If the program is unsure if there is a boat in frame, the photo will be sent to the review folder for manual inspection. Otherwise, the photo will be put in the archive folder for it to be tested for needles.

Fortunately, there will be no need for the client to have any sort of connection or interaction with the main functions of the program. The user only has to upload the desired folder, run the program, and get the results after the program has run successfully. The program has no need for a database or a server to be connected to. As long as the computer is functioning, the program will operate as described.

Module and Interface descriptions

Desktop application plan - This plan will make it simple and very general to operate our program at the AZGFD. The operators might not be the most experienced with running terminal programs via windows powershell. Compiling the application into a simple desktop application will allow the operator to be much more confident running the program since it will be similar to what they are already accustomed to in their current system.

GUI - For the graphical user interface, the team decided to use a similar look and feel of a canvas turn in. Students, mentors and instructors are all familiar with the canvas system for turning in files. This will include a simple drag and drop system for the input files or the user can manually select the file through a file explorer. Making the application have an easy input method will drastically reduce the learning curve for the operator. Talking and working with the customer we know that the existing software they are currently using (timelapse) has a similar

approach to accepting a file of pictures. Making this UI similar and updated will allow the customer to almost feel like they are doing the same task steps but the system is taking a lot of the hard work out of the picture that the operator used to manually do.

Process images - This is the most important and robust part of the program. Think of this as the brain of the whole process which will do all the sorting and comparing. The process images function will compare the current input images in the queue and will test it against its own database of knowledge to come up with a solution. The team will need to feed thousands of pictures outlined with the wanted objects so the program has a solid foundation to build its conclusion off of. It will conclude whether or not to keep the image based off of the presence or absence of an identifiable object within the image. Boats, Rafts, and kayaks will be the primary objects the brain of the program will be looking at. For the computer to correctly determine a boat from a raft will be very simple. When training the machine learning algorithm, the developers will feed in a large amount of pictures of rafts and boats. This way the program does not make the mistake of counting a commercial raft and wasting valuable time in which the operator would have to go back and delete this unimportant object. Machine learning depends highly on the training process at the beginning stage of development. Using the team's valuable time at the beginning of the semester to manually box out and identify objects present in the pictures will ensure a reliable program to use and build off of.

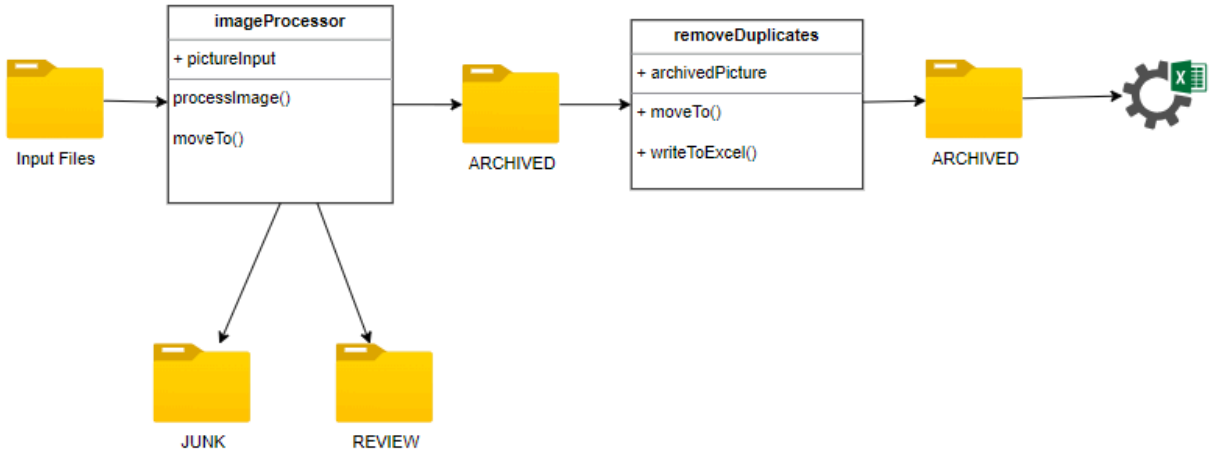
Sorting images - This will be a helper function to the process image main function. Producing a profitable output lies largely upon the quality of the input data. For our program, we intend to only pass inputs that have pertinent data, thus removing duplicate images and images that have no relevant objects that can be identified. When the input picture is first passed in it will go through a series of checks to determine if the picture will be optimal/profitable to pursue to the next step. This way we mitigate waste being passed to later, more crucial tests of the main function. Think of this as the sifting area in a slush box when looking for gold. There is a set of mechanisms at the start of the machine that will filter out a large amount of junk that will slow down the process later down the line. Sorting the images early on in the program will allow the main decision making process to compare against the best images to get the most optimal outputs. Outputting duplicate images is a huge risk in this program since it will negate the whole automated process that the customer is looking for. The main 3 outputs of the sorting function will be the Junk file which will include duplicates and pictures with irrelevant data, review file

where the image was close to the confidence threshold to pass the image as a boat and will need human revision, and finally the archived file which holds all outputs that the main function determined and a correct output to identify. Finally the archived file will always have duplicate counts of boats since it will not be able to determine if a boat is returning or leaving the boat ramp. This will be where another supporting function will run the output against the other archived pictures to determine if the boat is a duplicate and if it is returning or leaving to refine the output of the excel file.

Testing for duplicate photos - As stated previously this function will test archived pictures inputs against other archived photos. Depending on the angle and photo quality this will narrow down the duplicate counts that will appear in the output excel file. Before program completion and writing to excel file, this function will run and will essentially pass cut out images of the wanted objects to the comparison photo which will calculate the similarity of the object. Comparing 2 different photos with different angles, lighting and quality will most definitely have its errors but it will help out the operator by discarding the most blatant duplicates which can save revision time by the operator.

Write to excel - Finally the last step of the process will be the output file. The output file will include a status of the photo (leaving/returning), time of the photo, date, and name of the photo which is located in the archived file. Our customer uses this excel file currently with their time lapse software to document for their database. The customer is looking for our team to follow the same template which is currently being used so the database will not need to be altered. This will be very simple with the help of the Pandas library, and with some hard coded structure.justification of the cells. Ensuring the structure is clear and commented the maintenance down the road will be easy to change if the customer ever changes the database.

Class diagram



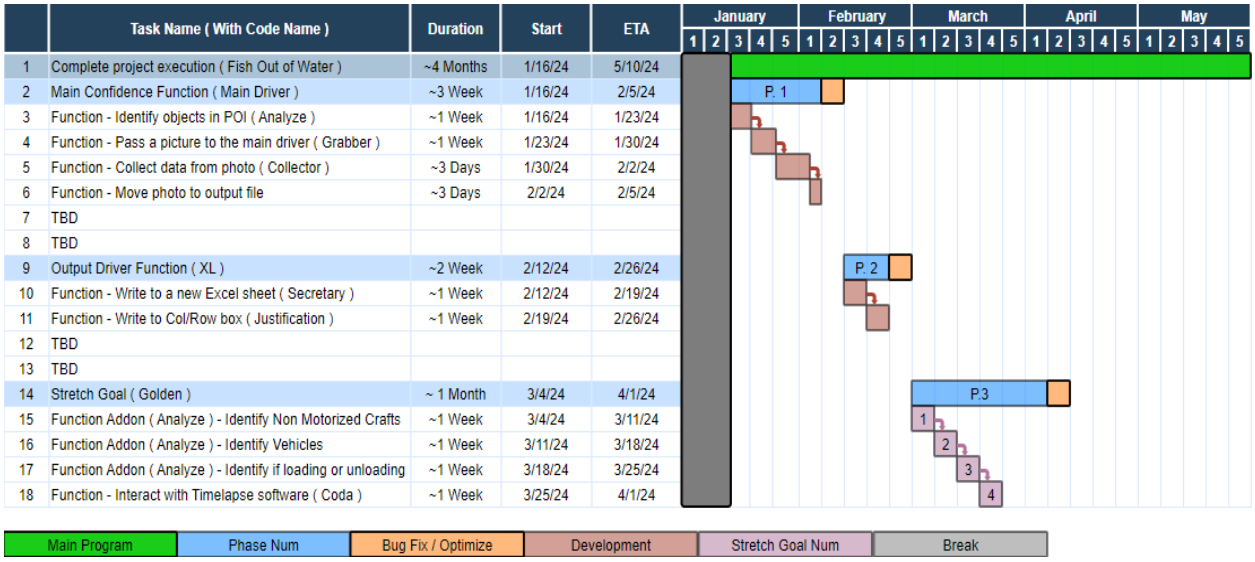
(Figure 2 - UML class diagram of the classes involved in this component of the program)

The imageProcessor class will represent the main driver for the object detection part of the program. This is where the input image folder is passed in and will iterate through all the photos in the file comparing it to the pictures in its database to decipher if it's a boat, raft or kayak. Even though kayaks are a stretch goal it will be easier to train the system to identify them at the same time as the other objects since they could possibly be in the same images. This class will also include helper functions that will box in the object and will move the picture to the desired output folder.

The removeDuplicates class is responsible for refining the archived file before output to excel. This will essentially do a similar thing to the processImage function but instead of deciding whether its a boat or not it will decide if its a duplicate boat that was already in the folder. The function will also test if the boat is returning or leaving the boat ramp. Lastly the main function of the class will be the writeToExcel function which will print the data, time, and photo name to the template given to us by the customer.

Implementation Plan

As you can see in “Figure 3” below is our Gantt chart that maps out our entire project implementation from the start to the end of this semester in May. Our Gantt chart consists of three stages that were specifically broken down to have our team focus on the same main functionalities of our project all at the same time. By working together on each main component at the same time, we will be able to guide each other in the right direction and not run into any implementation confusion as the project progresses. This is what we had in mind when drawing up our Gantt chart.



	Task Name (With Code Name)	Duration	Start	ETA
1	Complete project execution (Fish Out of Water)	~4 Months	1/16/24	5/10/24
2	Main Confidence Function (Main Driver)	~3 Week	1/16/24	2/5/24
3	Function - Identify objects in POI (Analyze)	~1 Week	1/16/24	1/23/24
4	Function - Pass a picture to the main driver (Grabber)	~1 Week	1/23/24	1/30/24
5	Function - Collect data from photo (Collector)	~3 Days	1/30/24	2/2/24
6	Function - Move photo to output file	~3 Days	2/2/24	2/5/24
7	TBD			
8	TBD			
9	Output Driver Function (XL)	~2 Week	2/12/24	2/26/24
10	Function - Write to a new Excel sheet (Secretary)	~1 Week	2/12/24	2/19/24
11	Function - Write to Col/Row box (Justification)	~1 Week	2/19/24	2/26/24
12	TBD			
13	TBD			
14	Stretch Goal (Golden)	~ 1 Month	3/4/24	4/1/24
15	Function Addon (Analyze) - Identify Non Motorized Crafts	~1 Week	3/4/24	3/11/24
16	Function Addon (Analyze) - Identify Vehicles	~1 Week	3/11/24	3/18/24
17	Function Addon (Analyze) - Identify if loading or unloading	~1 Week	3/18/24	3/25/24
18	Function - Interact with Timelapse software (Coda)	~1 Week	3/25/24	4/1/24

(Figure 3 - Gantt Chart, shows our general timeline for completion of project objectives)

Phase 1 (Main Driver) Confidence Functionality:

In the first phase, the entire team will be focusing on the training of our image recognition model. We will all be working together to train this in parallel to speed up the training process time. To train our model to detect boats in our point of interest we will be using the thousands of Lees Ferry Dock Images provided to us by our customer and feed those boats into our data set for our model to use. By training our model directly with the images it will be working with, we can curate the model to detect our exact boat preferences. This task will be an entire team effort so we can train our model in a timely manner.

Once we get our model to be able to detect boats confidently, we will implement our main driver where we will be able to pass in pictures iteratively to be processed. After the processing of each image, we will store all of the useful data provided by our model like the date, time, name of the photo, coordinates of the boat found, etc. for later use. Lastly, we will check if this image contains any objects of interest and move the image to its respective folder. This large task will also be spread out among all team members.

Finally, the creation of UI elements and unit-testing frameworks will be happening in parallel with all coding implementation throughout the entire phase. Jaron will be in charge of setting up the unit-testing framework and the whole team is responsible for UI element implementation. This wraps up our first phase.

Phase 2 (XL) Output Driver Functionality:

Our second phase, which is our shortest, will be mainly focused on taking this useful data provided from our model and displaying it neatly in an XL spreadsheet. We will display this data in order of the earliest date and time stamp from top to bottom. Each column in the XL spreadsheet will include the date and time of the photo, the image name, if a boat was found in the image, which folder the image now resides in, coordinates of the boat respective to the image, etc. Will write this data correctly to a raw column justification for each image. Lastly, We will be working on unit-testing and GUI elements in parallel with this implementation.

Phase 3 (Golden) Stretch Goals:

Our third phase will be completely focusing on our stretch goals for this project. We will put all of our focus as a team on training our model to also be able to identify kayaks and vehicles dropping off/picking up boats on the boat ramp. The most important stretch goal would be to identify the same boat leaving and arriving back to the boat ramp, as well as include in the spreadsheet how long this boat was out on the river. The final stretch goal in mind is to be able to have our software interact with the customer’s existing software Timelapse for easier collaboration between the two tools. These stretch goals will be split amongst team members to be worked on in parallel and assigned to team members based on available workload.

Conclusion

The Arizona Game and Fish Department's (AZGFD) mission is to conserve diverse wildlife resources and manage for safe, reusable outdoor recreation opportunities for current and future use. The AZGFD faces many different challenges when it comes to achieving their mission while they work with the unpredictable "mother nature". The environmental constraints pose a legitimate challenge for a successful implementation of our solution. Unfortunately, it is impossible to move the camera closer to the dock because of privacy concerns of the patrons at the river as well as theft and/or tampering concerns. There are many environmental factors that can affect the quality of the pictures that are taken by the game camera. This includes things such as inclement weather or glare. This could make it difficult for our software to correctly analyze whether a photo needs to be reviewed, kept, or discarded. This is why they focus their attention more on the problems they can control rather than the ones they cannot. When it comes to preserving the rainbow trout of the Lee's Ferry Fishery, the AZGFD is able to monitor and control the amount of angler boat traffic coming in and out of their dock.

As you may know, when one new solution comes into action, it can bring multiple problems along with it. The process that the AZGFD has set into place to monitor the amount of angler traffic the Lees Ferry is getting is way too time-consuming of a task for a human. The monitoring system in place takes more than 1,800 pictures a day, and a human needs to fish through each image to tag every boat, kayak, and trailer that enters and leaves the dock. This process takes at least 5 hours to complete only one day's worth of images captured. The new problem the AZGFD has introduced is that they've created more manual work for their workers.

Our team, Fish Out of Water's goal is to cut down the amount of images our customer has to look through and ultimately try to automate this process as best we can. We hope to solve their time-consuming problem by developing a desktop application that will process these images for them, hopefully with little to no human interaction. Our program should create a summary of its findings and display this angler boat traffic data in an Excel spreadsheet to be easily consumed by the AZGFD. We hope this program will be able to save them hours of work so they can shift their attention toward other projects that align with their mission.

Fish Out of Water is very passionate about this project, as all members of the group have a love for fishing. The team only wants to see the fishing environment at Lee's Ferry and other fishing spots in Arizona thrive and flourish. We will be hard at work this upcoming year to

provide the AZGFD with this autonomous program for them to utilize for years to come. We are highly confident that our current proposal above will be sufficient to solve their problem, and we are all eager to get to work on our solution.