Finding Hay in the Needle Stack: Doing the Right Thing at the Right Time

Project plan

Team Number: sddec19-09
Client: Collins Aerospace
Adviser: Srikanta Tirthapura
Team Members/Roles
Austin Konsor: Software Architect/Chief Engineer
Mireille Mwiza Iradukunda: Test Engineer/Report Manager
Gooi Yin Hong: Lead Designer/Web Master
Yealim Sung: Project Leader/Meeting Scribe
Team Email: sddec19-09@iastate.edu
Team Website: sddec19-09.sd.ece.iastate.edu

Revised: 2/21/2019/Version1

Table of Contents

List of Figures

List of Tables

1 Introductory Material

1.1 Acknowledgement

We are thankful and excited to be assigned this project with Collins Aerospace. Specifically we would like to thank Branden Lange, Kirsten Dawes, and Andy Zobro for their continued help, resources and guidance. We hope to fulfill their needs and deliver a final product to help with their lasting success.

We would also like to thank Srikanta Tirthapura. His support and guidance will be very valuable in the coming months as we prepare to deliver an outstanding product to Collins Aerospace. Finally, we would like to thank all teaching assistants in this process, as much of it is volunteered time, it is greatly appreciated.

1.2 Problem Statement

Collins Aerospace has about 8000 common test files that are spread across many directories within a SVN repository. These are python files used for display in airplanes. However, of those 8000 files, there are may be many versions of the same test program.

These will have the same file name with the same functions, yet some may work better than others. As of right now, the Collins engineers have no way of knowing which file works the best unless they were to manually search through all of the programs themselves. So the engineers must fix each functions everytime they find a mistake, when that mistake may have been fixed in another version.

In a database called STARWARS, Collins has a collection of test result data regarding each of the programs. This contains data including a time stamp and if a specific function either passed, failed (no errors, however did not run correctly), or crashed (did not compile or run all way through). So, given a user input of a filename + function name, they would like a desktop application to give a suggestion on which version of that specific file to use, relating to the STARWARS test result data.

1.3 Operating Environment

The final product will be a desktop application (written in Python) that will strictly be run only on Collins Aerospace systems, or computers. This is to ensure that no data is seen outside of Collins Aerospace. Their systems just use Windows OS, so there isn't any difficulties with the desktop application. The Python version would be 3 since Python version 2 will stop operating soon. For the database, MongoDB seemed to fit our requirement the most which takes in key-value pairs.

1.4 Intended Users and Intended Uses

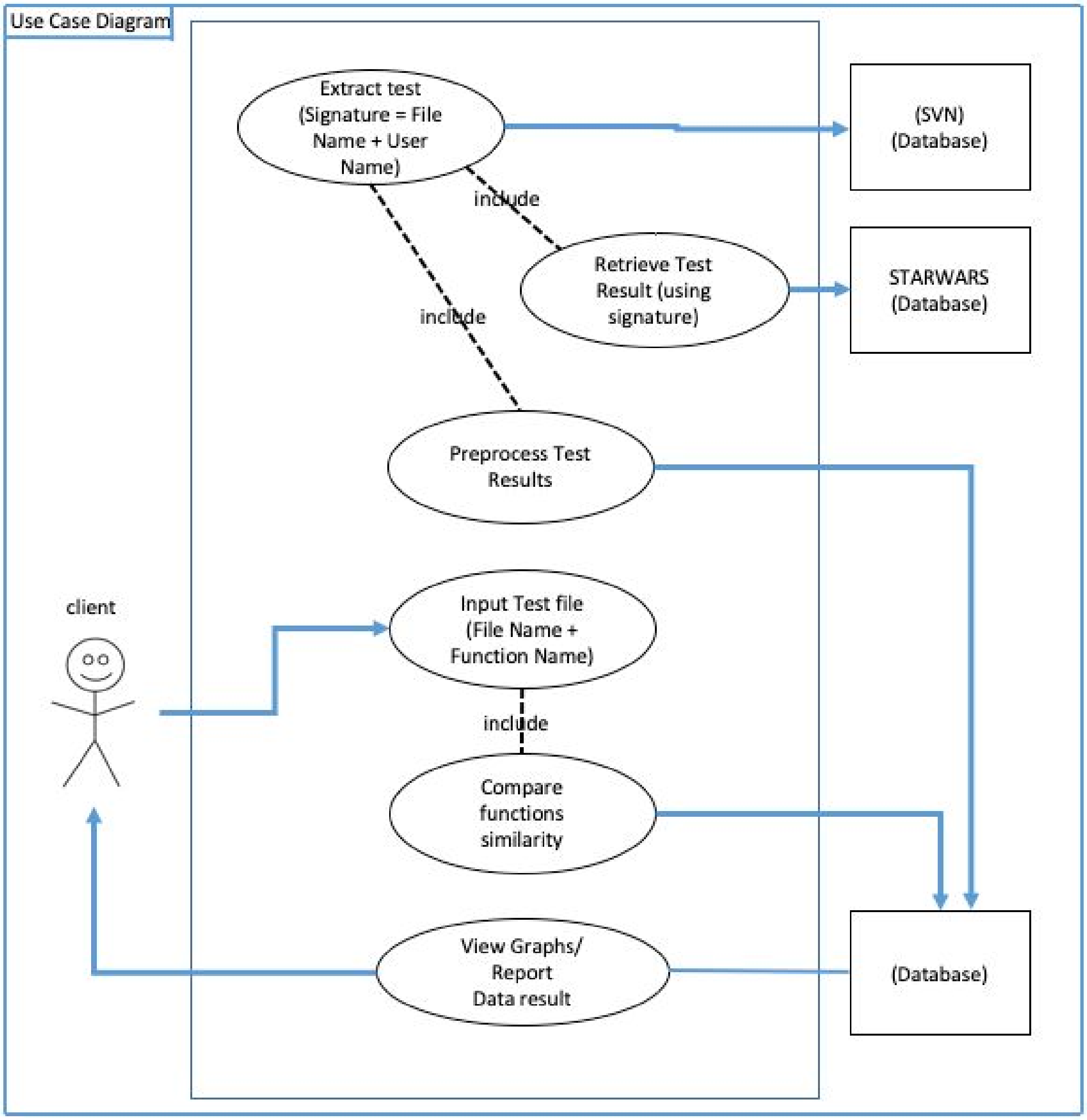

Use Case Diagram
Extract test (Signature = File Name + User Name)
(SVN) (Database)
include
Retrieve Test Result (using signature)
STARWARS (Database)
include
Preprocess Test Results
client
Input Test file (File Name + Function Name)
include
Compare functions similarity
View Graphs/ Report Data result
(Database)

Figure 1: Use-case diagram of our project.

The only users of this application will be the developers at Collins Aerospace. There are one intended uses for this application, that is to get a good data representation of a test file's historical performance after inputting the test's name into the user interface.

The main uses of this system is to help the engineers in Collin's Aerospace to give better recommendation of which test file should be used, in order to improve their work efficiency, to help drive intelligent decisions, and drive down the cost of testing.

1.5 Assumptions and Limitations

Assumptions

- Mock Test Data will be sufficient to build application - Collins Aerospace is unable to give us the true data from their SQL servers to keep it private, so they will give us a mock data set to work with which will imitate a real set.
- This application will only run on a Collins Aerospace system, even for development and testing.

Limitations

- As stated above, we are only able to get a sample data set to work with, however it should be sufficient enough to build the application and work with the true data.
- We must work on the development at the ISU Research Park, there they will provide a Collins Aerospace laptop to work off of. We may code on our own devices, but all data will be stored on that laptop.
- We need a Collins Aerospace employee present in order to gain access to the laptop, so we will need to work around all of our schedules to do so.
- Only one of our team members have previous experience with big data and Python.

1.6 Expected End Product and Other Deliverables

The delivered product will be a desktop application that the developers at Collins Aerospace will run on their systems, and only their systems. The application will mine through the directories in relation to the STARWARS test result data in order to feed our MongoDB by creating a key for the specific file/function name combination and pairing it with the results. The key will be unique to the file version by the difference in the implementation of the function. The user (Collins Engineer) will then input a filename + function name, and the application will return the suggested version of the given file. The criteria will be based off of version date, test results, and overall compatibility. The clients will also receive all related documents such as design, code and architecture documents in order for them to build off of this application in the future if they wish.

2 Proposed Approach and Statement of Work

2.1 Objective of the Task

The goal of this project is to deliver a software product that visualizes data and provides insights to Collins Aerospace engineers to help them in developing and testing the software. This tool should be able to assimilate and visualize data and develop signatures for tests. The resultant end product would be an application that will take several files as an input and possibly output a report and a visual.

2.2 Functional Requirements

Analyze input files Given the file and key, the program should find what the key represents. It should also identify how similar one program is to others and how many of them are using the same version.

Visualize end result After comparing the input file with other files, it should be able to output the result in a tabular form to help the user to clearly see the result.

Generate final report The summary of analyzation and visualization should be included in this final report that will be generated when the program ends.

Quick, accurate, and reliable output The result should help the engineers to quickly and easily determine what to do and provide deeper understanding.

Offer suggestions Given a file the program should see the trends and offer suggestions that will improve the result.

2.3 Constraints Considerations

The main constraint would be the security of the program itself and the input datas. As our client is in a corporate it is not easy to freely access their datas and resources. Team members are required to physically be in the facility in order to test the program and the input datas will be fabricated.

The team members are not familiar with Python language. Therefore, the members need to figure out how they are going to learn the language and who to find when there is a problem. The standard protocol would be to discuss internally if there is any problem, then to discuss it with the faculty advisor, and finally discuss it with the client.

2.4 Previous Work And Literature

dupeGuru is a duplicate file finder for Windows. Although it is normally used to delete the extra copies of a file, it can also be used to just identify the duplicates and where they may be found. It is one of the best of it's kind. That is the extent of its usage though, it is unable to parse through a file to see if it has the same functions or any of the other necessary requirements for our project. (Lifehacker.com, 2019)

Code Compare is a more complex software which takes the duplicate finding to the next steps. It is able to compare two files to see if certain lines of code match, with the ability to merge them together if needed. This could be used to make sure they have the same functions within the python files, and if so, are the functions implemented the same or will they produce different results. Again, this is as far as the usage goes as far as our problem is concerned. It would be useful to produce keys for the database. However, this product costs $49.95. (Devart Software, 2019)

2.5 Proposed Design

The block diagram shows the architecture of the program. It will take in the file directories in SVN(Apache Subversion) and aggregate the test data via our API and store the tests in the database. Then it will fetch the data from STARWARS database to aggregate the test result data and store it in the database as well. The API will fetch the test results and return it to the user by using the data visualization tools to output the report of the results.

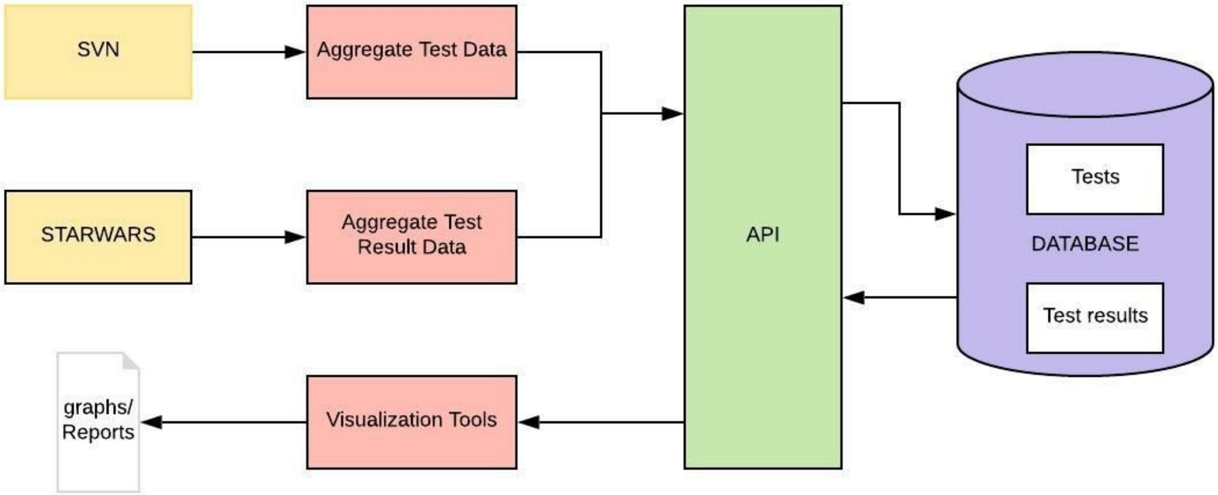


Figure 2: Block diagram of the basic architecture.

The table below is the possible design for the visualization of the output. C represents that the program crashed, P represents that it passed, and F represents that it failed. Regarding the information given, the user can tell which File is similar to which.

Table 1: Proposed design of the visualization output.

			Week															
Sig	%	Program	1	2	3	4	1	2	7	8	9	10	11	12	13	14	15	16

A1XHD	100	A							P	P			P					
		B		P	P			P		P			P		P		P	
		C	P	F	P				P									
		D														P		
BD4AO	95	A	C	C			F		C									
		F	C	P														
		G			C								P					
C489H	80	E				P												
		D	P		C			C					C					

The final report will show the path to the suggested files and show data visualization using the library 'matplotlib' from Python. The type of graphs would be determined once the testing is done with the actual data to see the relation between dataset but suggested graphs for current situation would be Scatter Masked plot, Vertical Bar chart, and Pairplot. The following visualization was chosen because they are the main plot/charts to compare datasets in one glance.

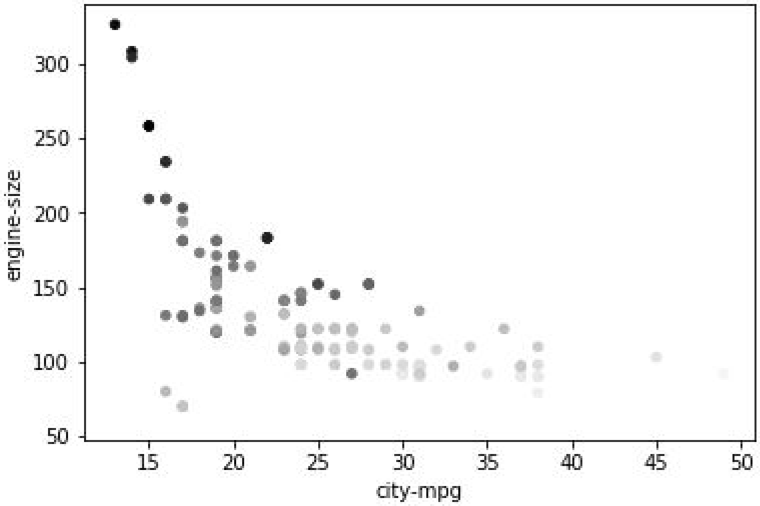


Figure 3: Scatter Masked plot example using matplotlib.

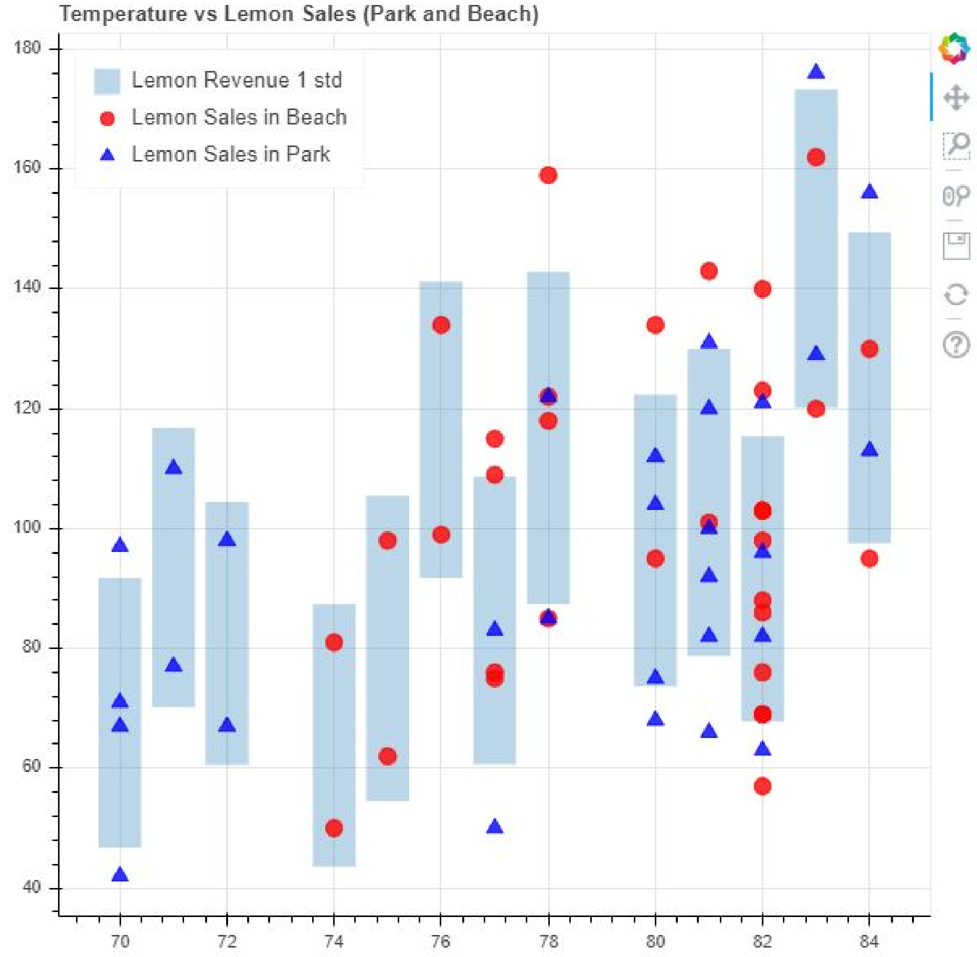


Figure 4: Vertical Bar chart example using matplotlib.

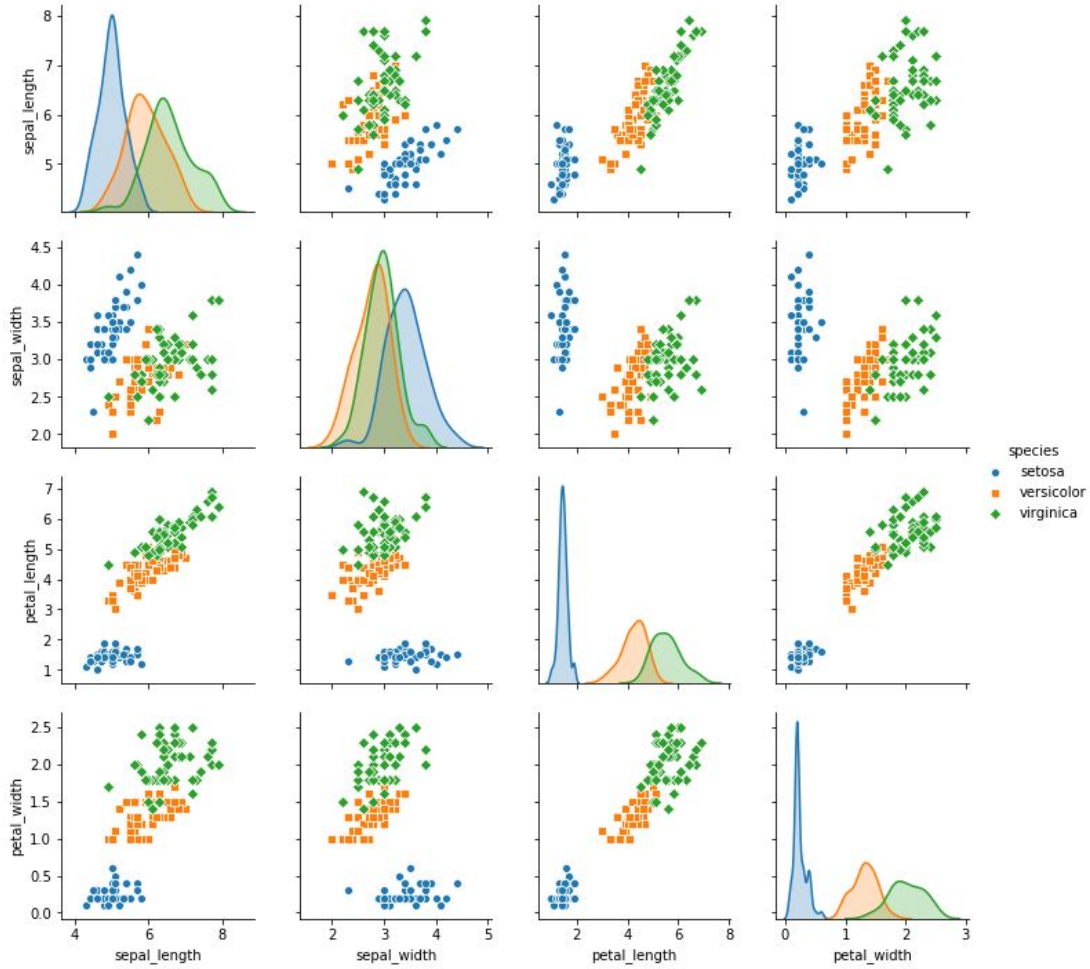


Figure 5: Pairplot example using seaborn.

2.6 Technology Considerations

As the project is delivered as a desktop application, if the zip file gets corrupted or has problem delivering, it would lead to a problem. In order to prevent any incidents, our team will keep a tight communication throughout the whole process.

2.7 Safety Considerations

Due to the nature of the project, there will not be any physical harm or risk in the process of developing our project or using our product.

2.8 Task Approach

Figure 1 as shown below is our current approach to our task at hand based on the design thinking model. In the Requirement Analysis phase, we will try to understand and identify the client's problem. We will list important requirements and specifications for the project. We will try gain all the requirements before developing the applications. Most of the work are done in documentation and meeting with the client in this phase. The development phase would begin in the next semester.

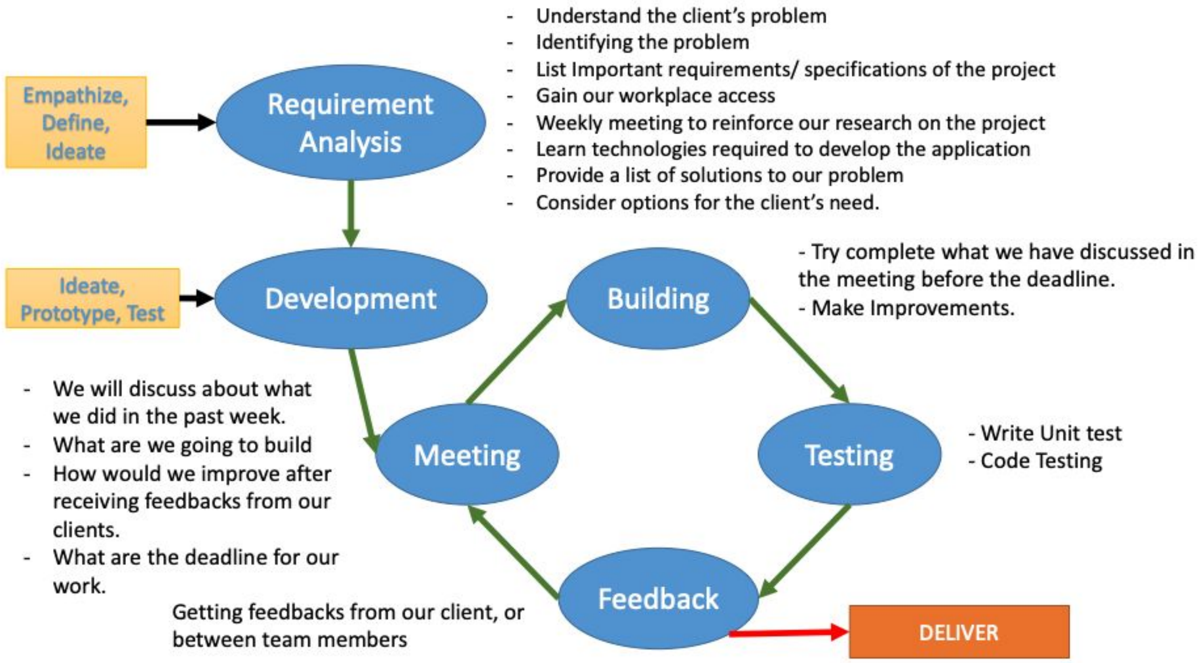

Empathize, Define, Ideate
Requirement Analysis
- Understand the client's problem
- Identifying the problem
- List Important requirements/ specifications of the project
- Gain our workplace access
- Weekly meeting to reinforce our research on the project
- Learn technologies required to develop the application
- Provide a list of solutions to our problem
- Consider options for the client's need.
Ideate, Prototype, Test
Development
Building
- Try complete what we have discussed in the meeting before the deadline.
- Make Improvements.
- We will discuss about what we did in the past week.
- What are we going to build
- How would we improve after receiving feedbacks from our clients.
- What are the deadline for our work.
Meeting
Testing
- Write Unit test
- Code Testing
Getting feedbacks from our client, or between team members
Feedback
DELIVER

Figure 6: Project Workflow

2.9 Possible Risks And Risk Management

Cost: After discussing with our clients, there are currently zero cost for the project as they will provide us equipment, laptops for us to develop the application.

Materials: No materials are needed for our project as most of the part can be done using computer.

Equipment: Computers are provided by our clients. The possible risk is that we might not be able to develop the application remotely which might also hinder our progress due to their company's policy and security measure.

Knowledge of area:
There is a risk that we might be developing the system's architecture in the wrong direction, but to mitigate this problem, we have been asking a lot of questions regarding our goals and requirement for this project.

We might also lack of python experience, because most of our team members only have a basic knowledge about using python, we have been starting to learn and revise the technologies that we are going to use. Risk of our end product algorithm are not as efficient and could consume time to process because we are probably going to deal with a huge amount of data.

Accuracy Issue: Since we are dealing with a large data and also comparing signatures between them, we must be precise with our results and do testing for different factors that might affect.

2.10 Project Proposed Milestones and Evaluation Criteria

Key Milestones: classifying data, collect relationship between the test, develop an application for data visualization, use those test's relationships to visualize the data.

Testing: We will be testing each of the milestones.

- First we will test if the classified data are being classified correctly by using the fabricated data set that will be provided by our client.
- We would also try to build known relationships as a start to test our program.
- Develop an application that will be able to retrieve data and show the data in an easy way to understand, and we will test if the visualizations are being output correctly.

2.11 Project Tracking Procedures

To keep track of our progress throughout the course. In this semester we will be using Trello, writing the "Weekly Status Report", and conduct internal weekly meeting, and meeting with our clients.

In the next semester, we will continue using Trello, writing "Weekly Status Report" and probably using a private repository (company's security policy) for developing the application.

2.12 Expected Results and Validation

Expected Results:
Building a wrapper: The wrapper should be able to classify each of the test functions and build a mechanism that is able compute a variety of relationships between each data sets. After collecting those relationships.

Visualization:

- The application should be able to show those data accurately.
- The visualization should be able to suggest the engineers on what test they should be using.
- How the tests have been performed historically.

Validation:

- We will test each of parts separately to ensure each unit works as expected.
- We will perform integration testing, system testing, and acceptance testing.

2.13 Test Plan

String comparison testing

The code that concatenates the string of a function will be tested with fabricated code that has the same code implementation but with different line spacing, and etc. The test pass when it is able to tell both are the same code.

Preprocess Testing

During the preprocess phase of getting extracted test from the SVN database and extracted test result from STARWARS database. We will test if the computed signature from a test is able to extract the same test file from STARWARS and SVN.

Interface Testing

In the interface development, we will feed in expected inputs into the interface application for an expected visualization. We will also test if the interface is able to communicate with the back-end server, and retrieve data from the database.

We will also use unit, integration and acceptance testing necessary until we have a complete system. For acceptance testing, we will have the client to test the end product of the system and receive feedback if more improvements should be done, otherwise it will be ready to deliver.

2.14 Standards

IEEE 14764-2006 - ISO/IEC/IEEE International Standard for Software Engineering - Software Life Cycle Processes - Maintenance

This standard provides efficient way to plan, execute, control, review, and evaluate maintenance process. Since most of the softwares are sold to the client, it is important to maintain its functionality in order to succeed. The code should be reusable for at least several years. As this project needs to be used by Collins engineers, it is important to keep up and maintain the code. This standard was approved by IEEE.

1012-2012 - IEEE Standard for System and Software Verification and Validation

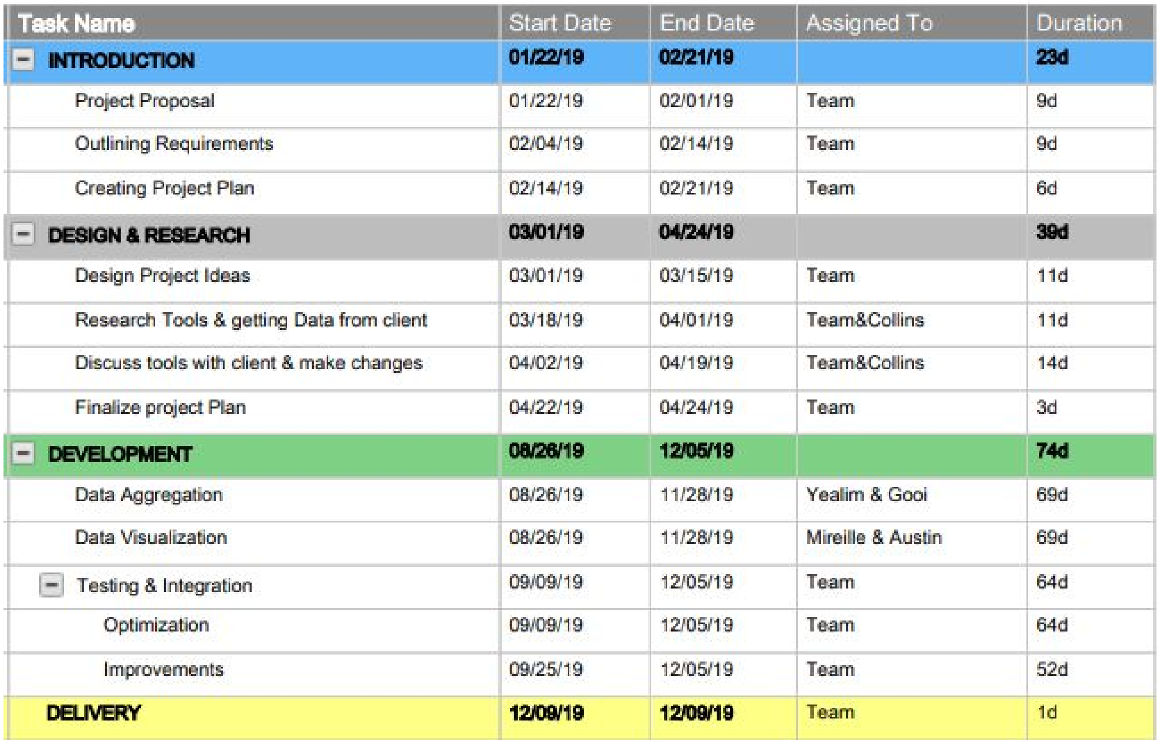
Verification and validation processes are used to determine whether the development products of a given activity conform to the requirements of that activity and whether the product satisfies its intended use and user needs. Life cycle process requirements are specified for different integrity levels. The scope of the processes encompasses systems, software, and hardware, and it includes their interfaces. It is important as it will help in developing, maintaining, or reusing the project. This standard was approved by IEEE.

The Role of Method Chains and Comments in Software Readability and Comprehension—An Experiment

This standard discusses about efficient way to comment so that everyone in the group will be able to understand each others code. I think it is an important matter since group project tend to trail off when one has to ask every line of code to another. The code should be readable since the author will not be the only person to use it. It should be easy to implement for the customers and other engineers. It will be helpful to follow this standard in order to make the transition between our team and the Collins engineers smooth. This standard was approved by IEEE.

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline



Task Name	Start Date	End Date	Assigned To	Duration
INTRODUCTION	**01/22/19**	**02/21/19**		**23d**
Project Proposal	01/22/19	02/01/19	Team	9d
Outlining Requirements	02/04/19	02/14/19	Team	9d
Creating Project Plan	02/14/19	02/21/19	Team	6d
DESIGN & RESEARCH	**03/01/19**	**04/24/19**		**39d**
Design Project Ideas	03/01/19	03/15/19	Team	11d
Research Tools & getting Data from client	03/18/19	04/01/19	Team&Collins	11d
Discuss tools with client & make changes	04/02/19	04/19/19	Team&Collins	14d
Finalize project Plan	04/22/19	04/24/19	Team	3d
DEVELOPMENT	**08/26/19**	**12/05/19**		**74d**
Data Aggregation	08/26/19	11/28/19	Yealim & Gooi	69d
Data Visualization	08/26/19	11/28/19	Mireille & Austin	69d
Testing & Integration	09/09/19	12/05/19	Team	64d
Optimization	09/09/19	12/05/19	Team	64d
Improvements	09/25/19	12/05/19	Team	52d
DELIVERY	**12/09/19**	**12/09/19**	Team	1d

Figure 7: Senior Design Grid View Schedule over the course of 2 semesters.

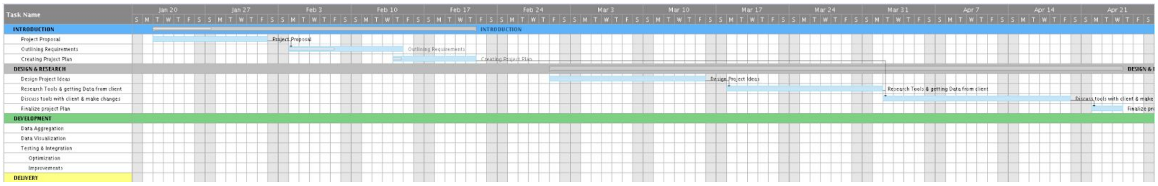


Figure 8: Semester 1 Gantt Chart

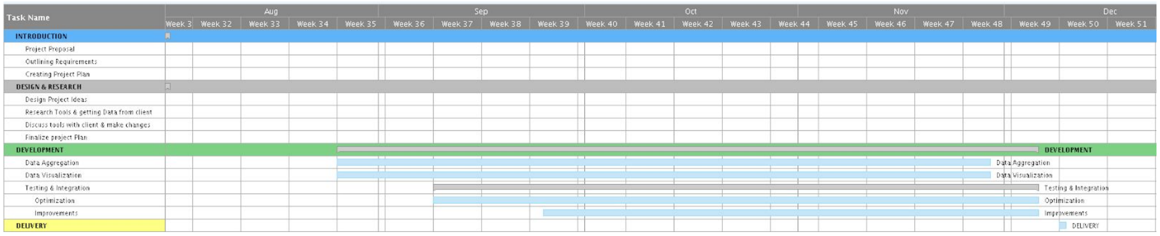


Figure 9: Semester 2 Gantt Chart

As shown above, The project will be divided into 3 phases and Delivery. 2 phases will be part of semester 1 and the last phase will be in semester 2. Currently, the project is almost at the end of phase 2 meaning phase 1 has already been completed. Phase 1 was an introduction to the team, client and the problem statement. Over the course of the first month the team has empathized with the client's problem to better understand their problem statement. The proposed timeline for this phase was 1.5 months but overall took 1 month.

Phase 2 started on march 1st, After the empathizing step, work was started on defining and ideating for the proposed solution for the problem statement. Research was conducted in regards of how we can achieve the proposed solution. The conducted research was both on the data to be used and the best tools to be used in the implementation of the project. The proposed timeline for this phase was 2 months but took 1.5 months.

Phase 3 will be the last phase that will last a whole semester. It is going to be comprised of development, testing and release. Development will be mostly made of two main parts that are Data Aggregation and Data Visualization. The whole process of phase 3 will be a cycle made of Development, Prototyping and Testing and once the product is good enough for the client and team, The project will be released. The proposed timeline for this phase is 4 months.

The phases are divided in this manner because although development and testing is very important, it is equally important for the team to draft and design the project, get a deep understanding of the problem before tackling it. This will ensure less mistakes that could be costly in the long run.

3.2 Feasibility Assessment

The challenge encountered by the team and that are still ongoing are access and clearance to data from Collins Aerospace. The issue is still being worked on at the moment as the client is still finding a way in which we can access their data to be able to develop the tool. This is part of the overall big challenge that will be a security challenge on their side. From the team's meeting with Collins Aerospace, it seems to be a big challenge for them to get security clearance for the team. The team will use Python which has good documentation on both data aggregation and data visualization. With this the project development process will be feasible.

3.3 Personnel Effort Requirements

Each member of the team currently has their roles. However when it comes to research and development, everyone is expected to do their part and cooperate to ensure that everyone is on the same page. All members are expected to give 100% of their effort in a timely manner.

Weekly Meetings where expected attendance is 100%

- Meeting Every Mondays without Client
- Meeting Every Tuesdays with Client

Research Park visit when needed to go to Collins Aerospace.

Clear communication between the team members

Each member is expected to report their weekly effort.

3.4 Other Resource Requirements

Collins Aerospace computers will be used in the process of testing.

3.5 Financial Requirements

This project will incur no financial expenses. The team and client decided to work on the Iowa State provided repository GitLab and other tools will be open source and free.

4 Closure Materials

4.1 Conclusion

Using the right test from a huge data set can be a tiring task to do. It can be a waste of time finding the right test, and sometimes wasting resources and cost for the company.

Our main purpose of this project is to help our client Andy Zobro and his team from Collins Aerospace which has more than 80 directories of test files to develop an application that is able to help their engineers to work more efficiently. The application that we will be developing will be able to give suggestions on what test should be used, shows how the tests have been performed historically, and the relationships between them.

We conduct meetings with our client weekly and we are open to receive any feedbacks to improve the client's need, each of our team members works closely with each other and help each others' back in order to succeed.

4.2 References

[1] *Smartsheet*, www.smartsheet.com/template-gallery/popular-templates.

[2] *Devart Software. (2019). Code Compare - free file comparison tool. [online] Available at: https://www.devart.com/codecompare/ [Accessed 24 Apr. 2019].*

[3] *Lifehacker.com. (2019). [online] Available at: https://lifehacker.com/the-best-duplicate-file-finder-for-windows-1696492476 [Accessed 24 Apr. 2019].*

[4] *The Role of Method Chains and Comments in Software Readability and Comprehension-An Experiment* - IEEE Journals & Magazine, ieeexplore.ieee.org/document/7404062.

[5] *"14764-2006 - ISO/IEC/IEEE International Standard for Software Engineering - Software Life Cycle Processes - Maintenance."* IEEE, standards.ieee.org/standard/14764-2006.html.

[6] "*1012-2012 - IEEE Standard for System and Software Verification and Validation." - IEEE Standard, ieeexplore.ieee.org/document/6204026.*

4.3 Appendices

Table 1: Proposed design of the visualization output.

			Week															
Sig	%	Program	1	2	3	4	1	2	7	8	9	10	11	12	13	14	15	16
		A							P	P			P					
		B		P	P			P		P			P		P		P	

A1XHD	100	C	P	F	P				P									
		D														P		
BD4AO	95	A	C	C			F		C									
		F	C	P														
		G			C								P					
C489H	80	E				P												
		D	P		C			C					C					

Figure 1: Project Workflow

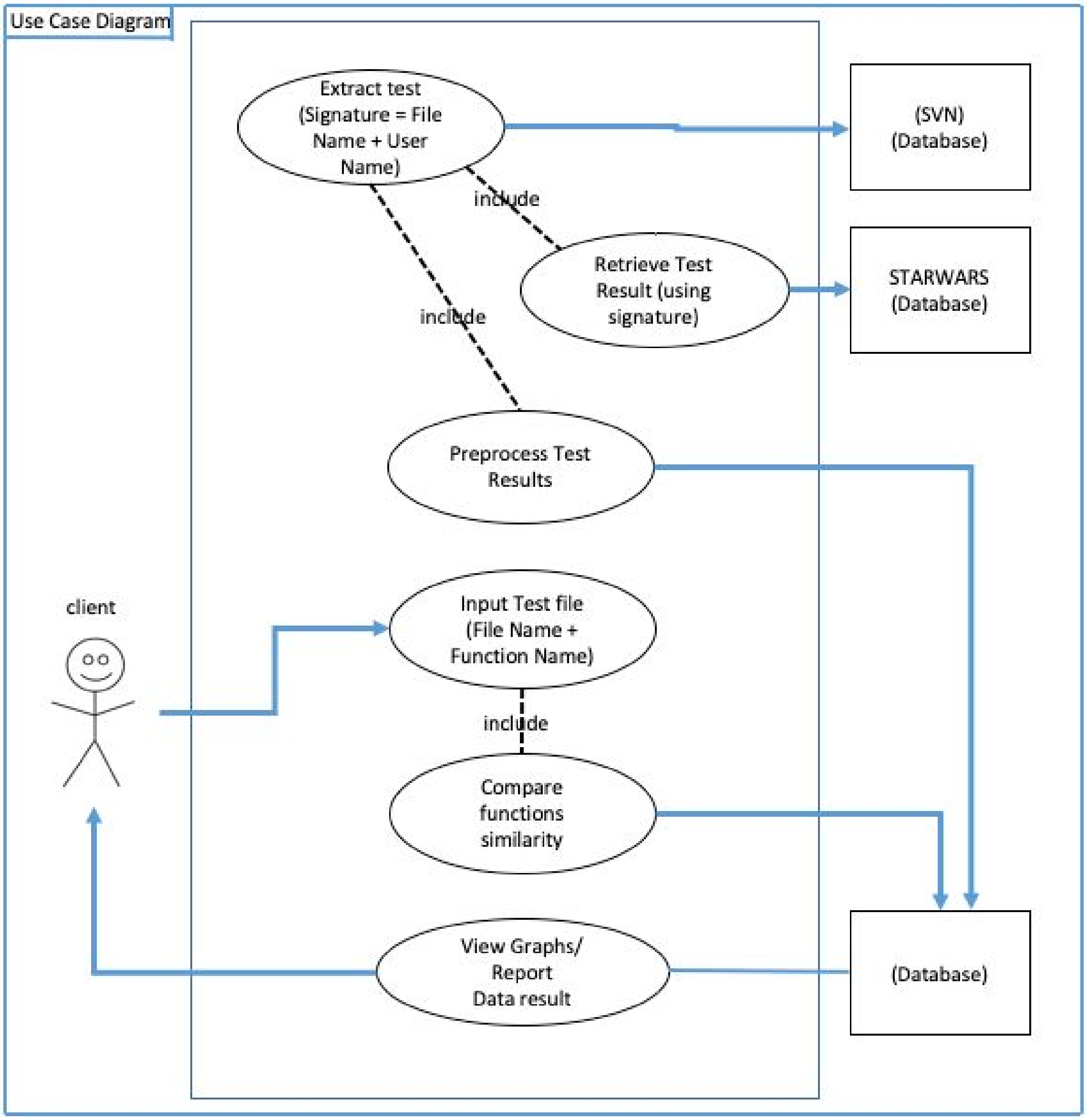

Use Case Diagram
Extract test (Signature = File Name + User Name)
include
Retrieve Test Result (using signature)
include
Preprocess Test Results
(SVN) (Database)
STARWARS (Database)
client
Input Test file (File Name + Function Name)
include
Compare functions similarity
View Graphs/ Report Data result
(Database)

Figure 2: Block diagram of the basic architecture.

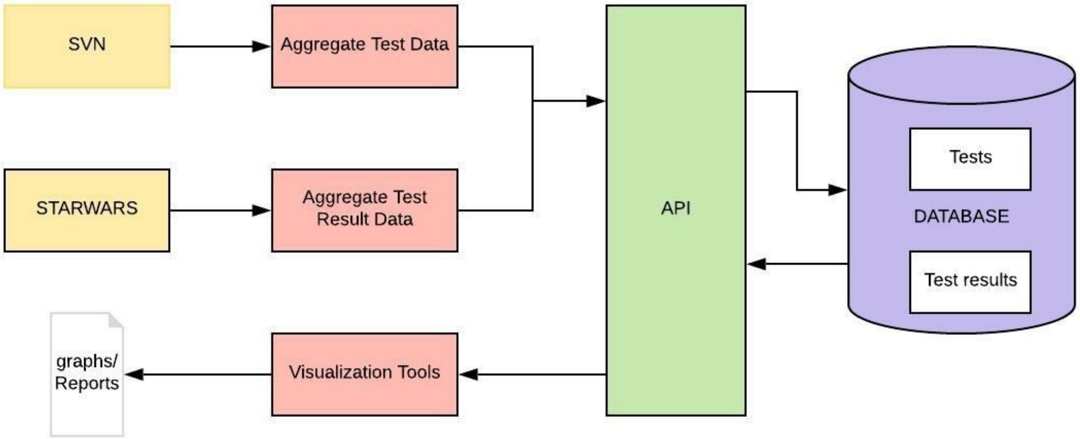


Figure 3: Scatter Masked plot example using matplotlib.

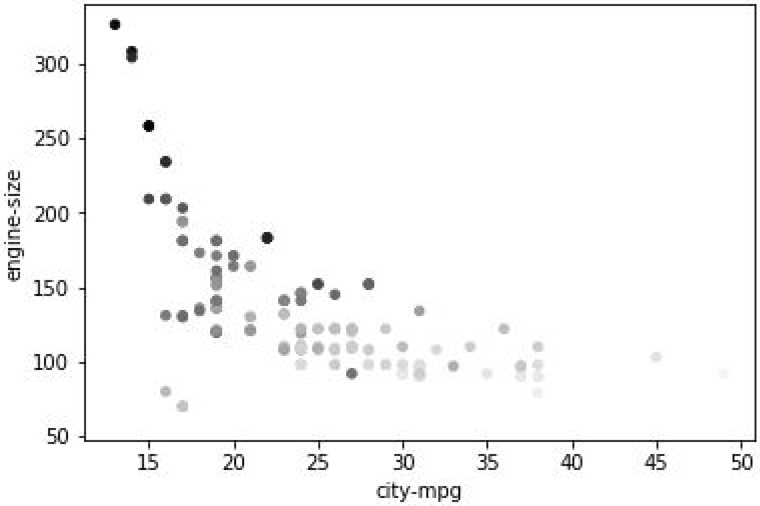


Figure 4: Vertical Bar chart example using matplotlib.

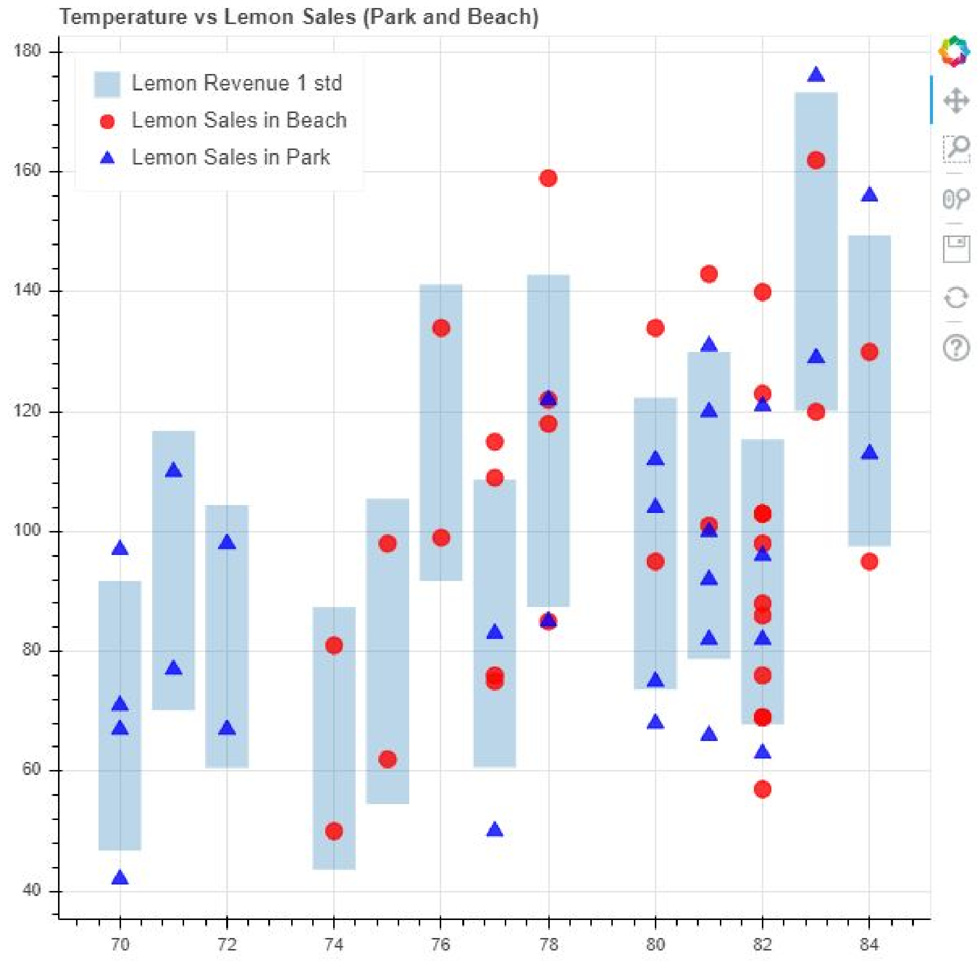


Figure 5: Pairplot example using seaborn.

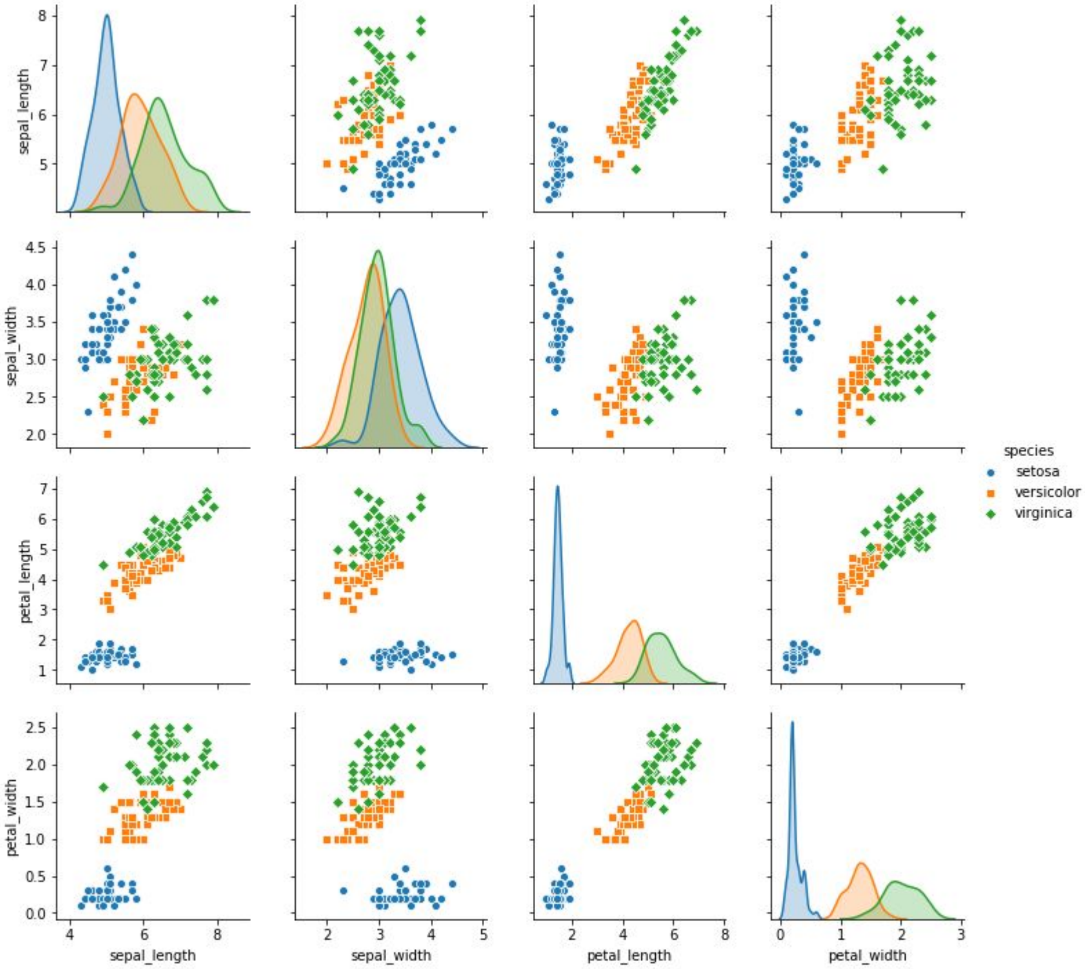


Figure 6: Project Workflow

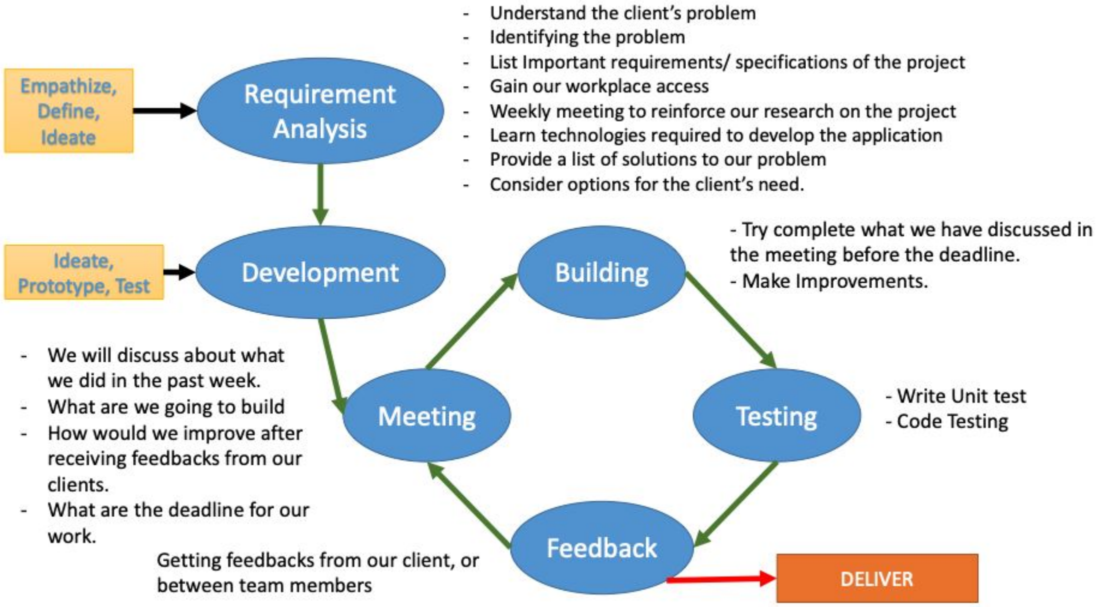
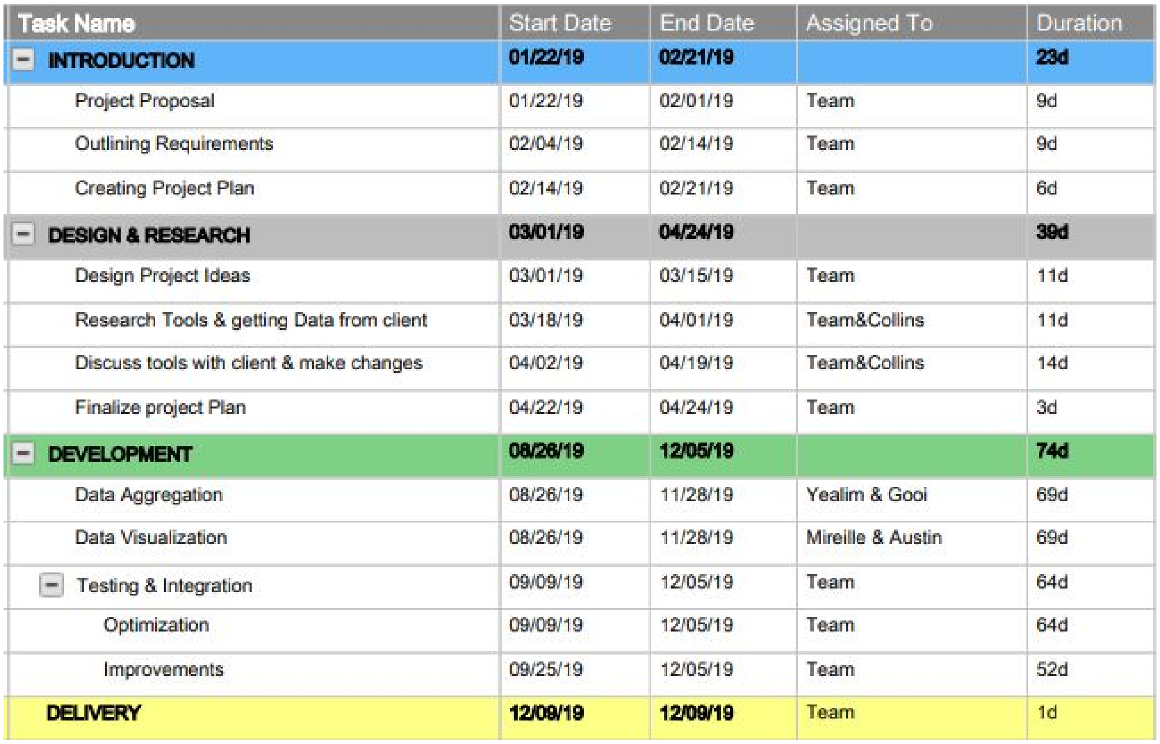

Empathize, Define, Ideate
Requirement Analysis
- Understand the client's problem
- Identifying the problem
- List Important requirements/ specifications of the project
- Gain our workplace access
- Weekly meeting to reinforce our research on the project
- Learn technologies required to develop the application
- Provide a list of solutions to our problem
- Consider options for the client's need.
Ideate, Prototype, Test
Development
Building
- Try complete what we have discussed in the meeting before the deadline.
- Make Improvements.
- We will discuss about what we did in the past week.
- What are we going to build
- How would we improve after receiving feedbacks from our clients.
- What are the deadline for our work.
Meeting
Testing
- Write Unit test
- Code Testing
Feedback
Getting feedbacks from our client, or between team members
DELIVER

Figure 7: Senior Design Grid View Schedule over the course of 2 semesters.



Task Name	Start Date	End Date	Assigned To	Duration
INTRODUCTION	**01/22/19**	**02/21/19**		**23d**
Project Proposal	01/22/19	02/01/19	Team	9d
Outlining Requirements	02/04/19	02/14/19	Team	9d
Creating Project Plan	02/14/19	02/21/19	Team	6d
DESIGN & RESEARCH	**03/01/19**	**04/24/19**		**39d**
Design Project Ideas	03/01/19	03/15/19	Team	11d
Research Tools & getting Data from client	03/18/19	04/01/19	Team&Collins	11d
Discuss tools with client & make changes	04/02/19	04/19/19	Team&Collins	14d
Finalize project Plan	04/22/19	04/24/19	Team	3d
DEVELOPMENT	**08/26/19**	**12/05/19**		**74d**
Data Aggregation	08/26/19	11/28/19	Yealim & Gooi	69d
Data Visualization	08/26/19	11/28/19	Mireille & Austin	69d
Testing & Integration	09/09/19	12/05/19	Team	64d
Optimization	09/09/19	12/05/19	Team	64d
Improvements	09/25/19	12/05/19	Team	52d
DELIVERY	**12/09/19**	**12/09/19**	Team	1d

Figure 8: Semester 1 Gantt Chart

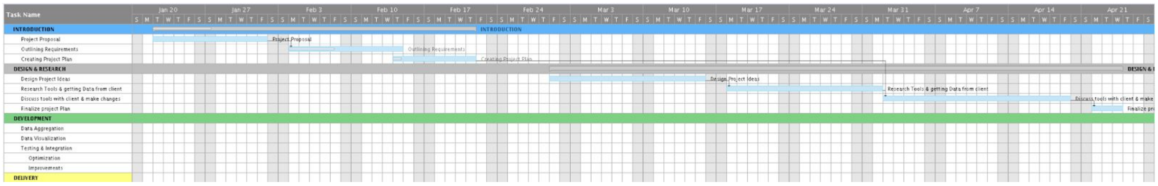
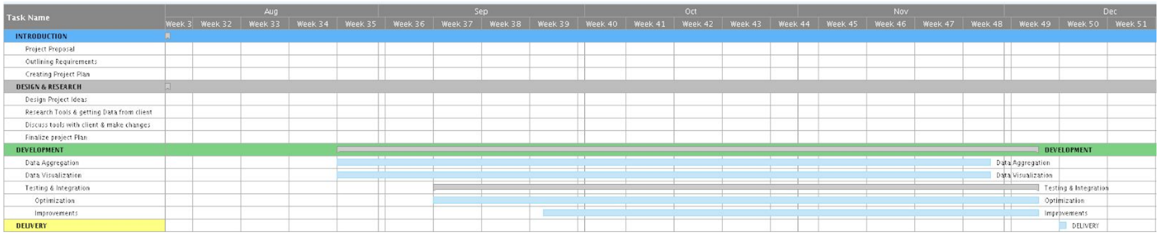


Figure 9: Semester 2 Gantt Chart


Task Name
Aug
Sep
Oct
Nov
Dec
Week 3
Week 32
Week 33
Week 34
Week 35
Week 36
Week 37
Week 38
Week 39
Week 40
Week 41
Week 42
Week 43
Week 44
Week 45
Week 46
Week 47
Week 48
Week 49
Week 50
Week 51
INTRODUCTION
Project Proposal
Outlining Requirements
Creating Project Plan
DESIGN & RESEARCH
Design Project Ideas
Research Tools & getting Data from client
Discuss tools with client & make changes
Finalize project Plan
DEVELOPMENT
Data Aggregation
Data Visualization
Testing & Integration
Optimization
Improvements
DELIVERY
DEVELOPMENT
Data Aggregation
Data Visualization
Testing & Integration
Optimization
Improvements
DELIVERY