

Understanding user sentiments by observing their software usage pattern

BITS ZG628T: Dissertation

by

Manish Singh

2017HT12447

Dissertation work carried out at

Adobe Systems, Noida



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)**

April 2019

Understanding user sentiments by observing their software usage pattern

BITS ZG628T: Dissertation

by

Manish Singh

2017HT12447

Dissertation work carried out at

Adobe Systems, Noida

Submitted in partial fulfilment of M.Tech. Software Systems degree
programme

Under the Supervision of
Dr Mohit Mahajan, Director, Adobe Systems, Noida



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)**

April, 2019

CERTIFICATE

This is to certify that the Dissertation entitled **Understanding user sentiments by observing their software usage pattern** and submitted by **Manish Singh** having ID-No. **2017HT12447** for the partial fulfillment of the requirements of **M.Tech. Software Systems degree** of BITS, embodies the bonafide work done by him/her under my supervision.



Signature of the Supervisor

Place : NOIDA

Date : 08/04/2019

MOHIT MAHAJAN,
DIRECTOR CUSTOMER EXPERIENCE,
ADOBE SYSTEMS INDIA PVT LTD.
SEC-25A, NOIDA.

Name, Designation & Organization & Location

Birla Institute of Technology & Science, Pilani

Work-Integrated Learning Programmes Division

Second Semester 2018-2019

BITS ZG628T: Dissertation

ABSTRACT

BITS ID No. : 2017HT12447

NAME OF THE STUDENT : Manish Singh

EMAIL ADDRESS : singhmanish.personal@gmail.com
manising@adobe.com

STUDENT'S EMPLOYING ORGANIZATION & LOCATION : Adobe Systems, Noida

SUPERVISOR'S NAME : Mohit Mahajan

SUPERVISOR'S EMPLOYING ORGANIZATION & LOCATION : Adobe Systems, Noida

SUPERVISOR'S EMAIL ADDRESS: momahaja@adobe.com

DISSERTATION TITLE : Understanding user sentiments
by observing their software usage pattern

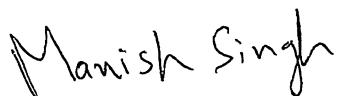
ABSTRACT

Adobe is a product based company specializing in making software related to digital media and digital marketing. Some of the flagship products Adobe has is Photoshop, Premiere Pro, Analytics and Experience Manager. Adobe focuses a lot on customers. For Adobe, the customer comes first and customer experience is of utmost importance. To ensure the highest level of customer experience, the company employs various mechanism which includes analysing the support tickets a customer opens, measuring customer happiness using various KPI's, customer surveys, sentiment analysis of support cases, amount spent on new products. The company also employs Account managers who's job is to advocate for the customer. It also does various summits, customer conferences so that the best customer experience can be delivered.

One of the missing areas is to understand the customer sentiment's while they are still working with the product. The current methods only let the company be aware of the customer sentiment when the customer reaches out with their problem. This project is the step forward where we can capture the customer sentiment's even before they start having a bad experience with Adobe's products by monitoring their software usage pattern, recommending help articles instantly and calculating the customer's sentiment score based how they are using the product. It will ensure a superior customer experience.

Broad Academic Area of Work: Artificial Intelligence and Machine Learning

Key words: Convolution Neural Network, GUI, Help articles, Recommendations, Sentiment Score, Training Data, Validation Data



Signature of the Student

Name: MANISH SINGH

Date: 08/04/2019

Place: NOIDA



Signature of the Supervisor

Name: MOHIT MAHAJAN

Date: 08/04/2019

Place: NOIDA

Acknowledgement

I would like to acknowledge and express my gratitude to Adobe Systems for giving me this opportunity to participate in the M.Tech program for Software Systems of BITS and providing all support as needed.

I would also like to express my sincere gratitude to my advisor Dr Mohit Mahajan for guiding me in this work and providing his valuable input from time to time and encouraging me to take up this topic.

I want to place my gratitude to BITS faculty and administration for providing full support whenever required.

Last but not least, I would like to thank my family for their continuous support while I was working on this project.

Table of content

Chapter 1: Introduction	1
1.1 Overview.....	1
1.2 Problem Statement	3
1.3 Scope of work	5
1.4 Plan and status of work	6
 Chapter 2: Approach.....	 7
2.1 Overview.....	7
2.2 Understanding the user.....	8
2.3 Features to be monitored	10
2.3.1 The way the user reaches the intended option in the menu system	10
2.3.2 The time taken by the operation to complete.....	11
2.3.3 The outcome of the operation of the intended action.....	11
2.4 High level overview of the system.....	12
 Chapter 3: UI and Building the agent into the application.....	 15
 Chapter 4: Processing the RAW data to determine the user experience	 21
4.1 Overview.....	21
4.2 Determining if the user took an optimal path or not for selecting performed operation	21
4.2.1 Designing the Convolutional Neural Network	22
4.2.2 Processing the outcome and delay in doing operation	32
 Chapter 5: The backend storage.....	 33
5.1 Overview.....	33
5.2 Storing product information.....	33
5.2.1 Features collection.....	33
5.2.2 Helpdocs collection	34
5.3 Storing user usage information	35
5.3.1 Users collection	36
5.3.2 Usage collection	36

5.3.3	overallscore collection	37
Chapter 6: Calculating the customer sentiment score.....		39
6.1	Overview.....	39
6.2	Calculating the customer sentiment score	39
Chapter 7: Integrating the system		42
7.1	Overview.....	42
7.2	Flow diagram for the server which analyses the user actions	44
7.3	Flow diagram of training server	45
7.4	Flow diagram of report generator	46
7.5	Screenshot from the working code	47
Summary		51
Conclusions and Recommendations		52
Directions for future work		53
Bibliography and References		54
Appendices.....		55

List of figures

Figure 1: A high-level representation of how Adobe can interact with the customer who is using Adobe product	2
Figure 2: Showing Adobe Photoshop screen layout	3
Figure 3: Dialog box for completing action in Adobe Photoshop.....	4
Figure 4: High level overview of the system	12
Figure 5: The initial load of the UI	16
Figure 6: The UI with the menu system.....	16
Figure 7: Dialog box showing the result of end operation	17
Figure 8: Sending training data to the server.....	19
Figure 9: Sending validation data to the server	19
Figure 10: Sending user usage data to the server.....	20
Figure 11: A good mouse path for File close operation	23
Figure 12: A good mouse path for Saving to google drive operation	24
Figure 13: A good mouse path for applying the diffuse filter operation	24
Figure 14: A good mouse path for applying fill operation.....	25
Figure 15: Graphical representation of ReLU function.....	27
Figure 16: Model summary of the CNN	29
Figure 17: Training the model	30
Figure 18: Showing a user pattern received.....	31
Figure 19: The input mouse path being analyzed	31
Figure 20: Showing the feature list in MongoDB.....	34
Figure 21: Showing the helpdocs in MongoDB.....	35
Figure 22: Showing the users of a customer in MongoDB.....	36
Figure 23: Showing the usage data of users of a customer in MongoDB.....	37
Figure 24: Showing the experience score in overallscore collection stored in MongoDB	38
Figure 25: Sentiment score of open project operation	41
Figure 26: System flow diagram.....	42
Figure 27: UI showing the help article	47
Figure 28: flask serving all end points	48
Figure 29: agent sending raw user usage data for analysis.....	48
Figure 30: Model being trained.....	49
Figure 31: Score being calculated	49
Figure 32: Overall Score for a customer	50

List of tables

Table 1: Status report of the tasks.....	6
Table 2: Showing the relationship between score and customer sentiment.....	14
Table 3: Schema for features collection	34
Table 4: Schema for helpdocs collection	35
Table 5: Schema for user collection	36
Table 6: Schema for usage collection	37
Table 7: Schema for overallscore collection	38
Table 8: Score to customer sentiments mapping.....	41

Chapter 1: Introduction

1.1 Overview

Adobe is a market leader in digital media and digital marketing products. It has some famous products like Photoshop, Illustrator, Dreamweaver, Experience Manager, Analytics. To make the product a success, Adobe invests heavily on the customer's experience. It envisions that the customer should never face any sort of difficulties. The company ensures that the customer succeeds in whatever they do using its products.

To ensure the customer is always happy, Adobe takes up various initiatives like Office Hours, Summits, Symposium to interact with the customer, measuring the pulse of the customer and pitch in new sales deed. Apart for that, the company has employed Account Manager, Technical account managers who advocate for the customer.

The company also monitors its support channels and keeps an eye on what issue the customer is facing and how are we helping them. The company operates its support channel via Phone, Web, Email and Chat. It sends a survey link at the end of each incident to understand how the customer felt about the experience they had with Adobe and the experience with the product. Internally, all this is tracked via some KPI. While the user uses the software, Adobe occasionally sends a small survey to the customer which pops up in the software itself. The purpose is to understand the experience the customer is having using the product. However, this approach is seen as an interruption and is not perceived as good by the customer.

The current methods employed by the company is unable to capture the customers experience while they are using the product. It is after the customer reports the issue; we are able to help them. Adobe wants to capture the customer sentiments while they are using the product so that the company can reach out to the customer even before the customer experience starts going bad. This would ensure that the customer has a positive experience always and the company can also know when to reach out to the customer well in advance. The idea here is to capture the user data in raw format and send it to Adobe so that it can be analyzed and appropriate action can be taken like sending some help document to the customer in real time and notifying the Account manager of the problems the customer is facing.

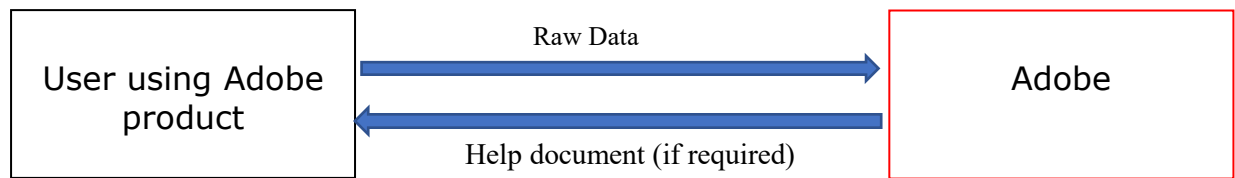


Figure 1: A high-level representation of how Adobe can interact with the customer who is using Adobe product

1.2 Problem statement

To determine the user experience, we would like to capture real-time data on how the user is using the Adobe product and based on that we would like to determine if a user requires help or not. Also, we would like to capture the user sentiment and calculate a score which gives a hint on how the user experience of a given product is. Adobe has standardized its user interface for its products. The standard Adobe UI is desktop based and follows a Menu Based approach. Figure 2 is a snapshot taken from Adobe's leading software, Adobe Photoshop

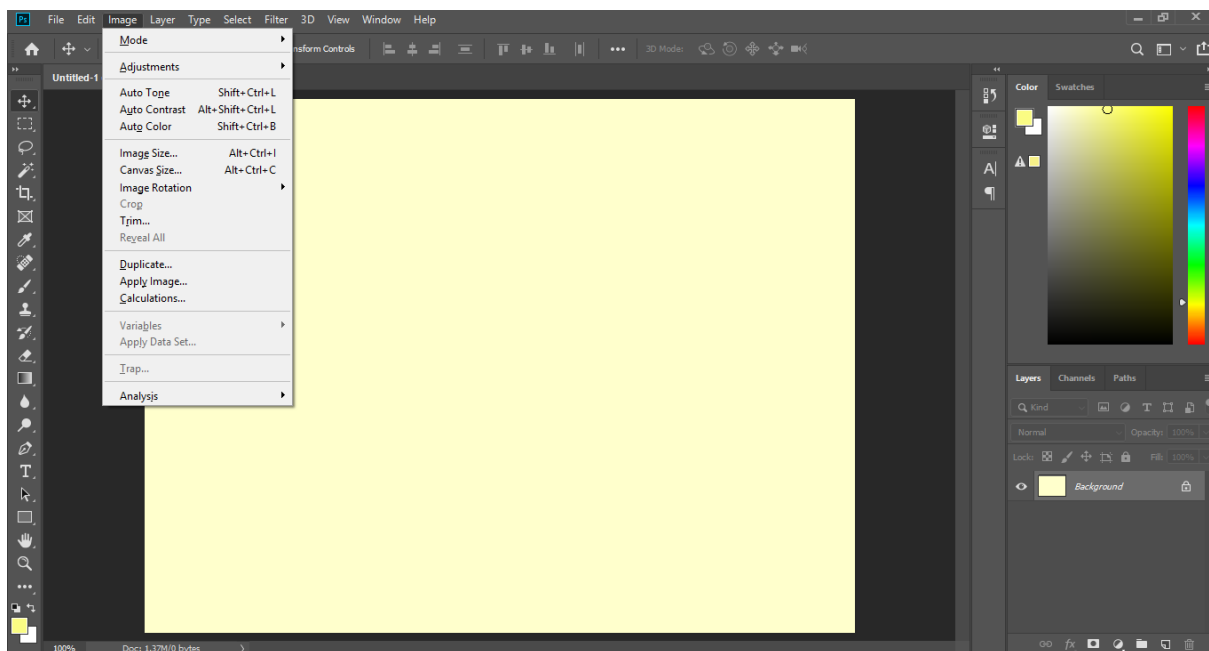


Figure 2: Showing Adobe Photoshop screen layout

Based on the option a user selects, a dialogue gets open and on pressing OK, the operation is applied. Figure 3 shows a screenshot of adding a new layer.

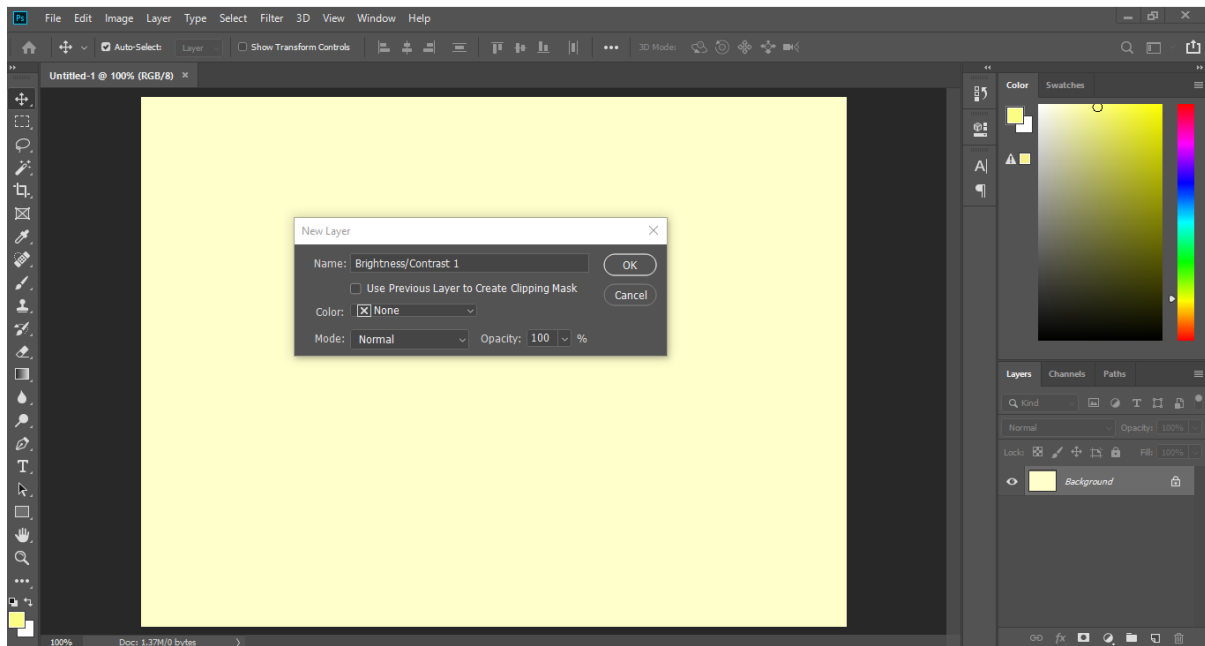


Figure 3: Dialog box for completing action in Adobe Photoshop

So basically, if we observe the user interface, a user will majorly interact with the products using the menus and the dialog boxes presented for completing the actions and this drives a major part of the user experience.

If we could understand the user behaviour based on how the user is using the software and how the outcomes are, we can categorize the user sentiments and can help the user even before they approach Adobe.

Adobe is now moving its solution into the cloud. Though the design principle of the UI remains the same, the company is now building the same experience on the web and intends to shift to cloud-based offering of its application.

1.3 Scope of work:

1. To identify the good usage pattern of the product.
2. Understanding the customer usage of the product and identify if the customer is facing an issue or not.
3. Suggesting help articles to the user in case they face an issue using the product.
4. Calculating the customer sentiment score and categorizing the customer as Sad, Disappointed, Neutral, Happy, Extremely Happy

1.4 Plan and Status of work

Task	Status	Remarks
Making a UI prototype	Done	Built in HTML, CSS and JavaScript
Building an agent for capturing user usage data	Done	Built in JavaScript
Designing the approach on identifying the customer usage experience	Done	Used CNN and operation outcome to determine user sentiment
Building a model for predicting the outcome of the mouse path	Done	Built in python using TensorFlow
Training the model	Done	Captured various mouse paths
Building the backend to store user usage raw and processed data	Done	Data is stored in MongoDB
Integrating all the components and running it end to end	Done	Used Flask to integrate the system

Table 1: Status report of the tasks

Chapter 2: Approach

2.1 Overview

To understand the user sentiments while they are using the product, it is quite important that we capture various parameters in raw form and process it in such a way that it can tell us the sentiment of the customer who is using our product. Deciding on what parameters need to be captured is quite essential as capturing the wrong parameters can lead to incorrect calculation of the customer sentiment score.

As we are taking up Adobe's UI format, we figured out that we would need to emphasize on the menu system and the outcome of the operation a user is doing. To build a holistic approach, it was important that we look into the following aspects:

1. What is the proficiency of the user using our product i.e. a user can just be a beginner who has very limited to no knowledge on how to use our product, all the way up to a professional user.
2. How the user is selecting the options for the operation they want to perform. i.e. what path did the mouse travel. As we use menus a lot, it is very important for us to understand this aspect.
3. What was the outcome of the operation. Did it fail or succeed?
4. How much time did the operation take to complete i.e. did an operation take more time than what it should take.

If we capture data around these points then we can understand the sentiments of the user while they are using the product. As we capture all of this data in real-time, we can compute the raw data received in a 24-hour cycle and get deep insights into the experience the customer is having with the product. Based on the captured data, we can also determine which feature of the product is prone to have a negative effect on the user experience. This is quite handful as this data can be shared with the product design team and also with the customer experience team so that we can support the customer even before they feel the heat while we think on redesigning the feature in parallel.

2.2 Understanding the user

As we talk about customer sentiment, it is very important that we first understand the user. We can generally categorize a user into three categories:

1. **Beginner:** This type of user is the one who is starting out with the product or has very limited knowledge of the product. This type of user needs the maximum handholding as they are likely to feel negative about the product during the initial phases. Though Adobe tries to make its UI simple, given the functionalities Adobe provides in the product, it becomes very confusing for a beginner to understand where to find the required option and how to use it successfully.
2. **Intermediate:** This type of user is the one who has gained some knowledge using the product. They are aware of the majority of the features and can understand the product terminology. However, these users may not know the best practices for doing an operation or may not be aware of an efficient way of doing another task. Such users are still in the process of learning. This type of user forms the majority of the userbase.
3. **Professional:** This type of user knows the in and out of the product. They are the people whom other users look up to for help or to do complex tasks. They have a higher say in the organization they work in. If this type of user faces any type of issue, it can have a high impact on the overall perception of the product.

If the user faces any issue while working on the product, it would be helpful if we can present a help article so that the user can rectify the problem then and there instead of reaching out to the product support team and spend extra time explaining the problem and understanding the solution. It has been observed that even for a simple issue, a customer has to wait up to 3 days before they get a proper answer given the fact that the solution was already documented on the product documentation page. So, serving a help document the moment we detect that a user is facing an issue saves not only time for the user but also elevates the user experience. This also saves the company some resources as the company would not be required to spend additional resources on dealing with customer issues and can

measure the pulse of the customer mood even before the customer start forming a bad perception about the product.

For Adobe, a customer can be of two types:

1. **Individual:** This type of customer is an individual user who purchases the product directly and use it themselves.
2. **Enterprise:** This type of customer generally range from small teams to big enterprises and have multiple users.

When calculating the customer's sentiment for a given product, we need to account for all the users using the product for a given customer.

We can assume that if a professional user is facing issues, the customer sentiment about the product would not be good as professional users are generally at key positions like Architect, Lead Developer, CTO, influencers etc. and their opinions matters the most. It can be a deal maker or deal breaker. A bad customer experience may even lead to loss of business which in turn leads to loss of revenue. So, whenever a professional user faces a bad experience, it should have a higher impact on the customer sentiment score.

As intermediate users form the majority of the userbase in any given team, we also need to monitor their experience closely. It reflects how the majority of the users perceive the product. This user segment should have a good impact on the customer sentiment score. However, the effect should not be equal to professional users as intermediate users are still learning the product. Though they are large in numbers, their opinion about the product is only taken as a feedback and may not have much impact on the customer's opinion about the product. However, their opinion should not be downscaled completely. Adobe can use the data generated by this layer to strengthen the product experience further and hold short seminars for users in this layer.

For beginners, we can safely assume that they will face difficulties. So, their impact on the sentiment score should not be given much weight as they are just starting out and learning the software. However, after analyzing the raw data from this layer, Adobe can accurately target the areas where it needs to focus on for training and can improve its help recommendations.

In all, the customer sentiment will be calculated based on the raw data we capture from the users. Based on this data, we can categorize the customer

sentiment and can help them bring to a level where they are happy with the product even before they start complaining or decide to end the contract with Adobe.

2.3 Features to be monitored

In the last section we talked about the importance of user types and how can they influence the customer sentiment score. Traditional methods have focused more on analyzing the customer tone on support forums and social media using Natural language processing. The disadvantage here is that this analysis is only possible when the customer have raised their complaints. If we have to monitor the customer sentiment's as they use the product, we can't rely on natural language processing as all of Adobe products are fully GUI based. Therefore, to calculate the customer's sentiment score and to recommend appropriate help documents we have to monitor the following features:

1. The way the user reaches the intended option in the menu system
2. The time the application takes to complete the operation
3. The outcome of the operation of the intended action

If we monitor these three key parameters, we can capture enough data around them which can help us understand the customer's sentiment while using the product. Let us see each parameter in details.

2.3.1 The way the user reaches the intended option in the menu system

In a GUI based system, a user interacts with the menu system using a pointing device like Mouse. Ideally, the designers try to place each option categorically and logically so that the user can find the desired options without having a cognitive overload. If we monitor the path of this pointing device and compute the path taken by the user to select the operation as a good path or not, we can understand if the user faced any difficulties. For example, to select Option C, the ideal path would have been

A->B->C

and we observed that the user selected

F->H->J->A->E->B->C

We can say that the user took a longer path which is indeed not good user experience. It suggests that the user had difficulties in finding the option or he was not well aware where the option is placed. If we can educate the user about the menu system by suggesting a help article on where to find what, instantly, we can certainly improve the user experience of the user. In case, we see a lot of bad user experiences finding a given option we can certainly look into restructuring the position of the options as well.

2.3.2 The time taken by the operation to complete

Every company wants their application to work instantly. For the same, a company invests a lot of resources to ensure that the application execution time is minimal. A long processing time means that the user has to wait for a longer time for an operation to complete. An application can take a long processing time due to various reasons like installation of a plugin, incorrect previous shutdown, CPU is busy in some other process etc. If an application takes a long time i.e. more time than what is expected, it can affect the user experience. This can be an indicator of an upcoming problem which can degrade the performance of the application or a sign of an already occurring issue. Thus, measuring this aspect is also essential and should be used in calculating the user experience.

2.3.3 The outcome of the operation of the intended action

What matters the most is if the user was able to complete their operation successfully or not. If a user was not able to complete the operation they intended, it leads to an awful experience and entices a wrong perception of the product at a fast pace. Therefore, we should not only immediately present a help document that can help the user troubleshoot the issue but also give more weight to operation failures when calculating the customer sentiment score

2.4 High level overview of the system

After considering all the aspects of the problem, it is understood that we need different components to design the algorithm which can help us in determining the customer's sentiment score as well as suggest help documents to the user the moment we detect a bad user experience. The figure below gives a high-level flow of this system.

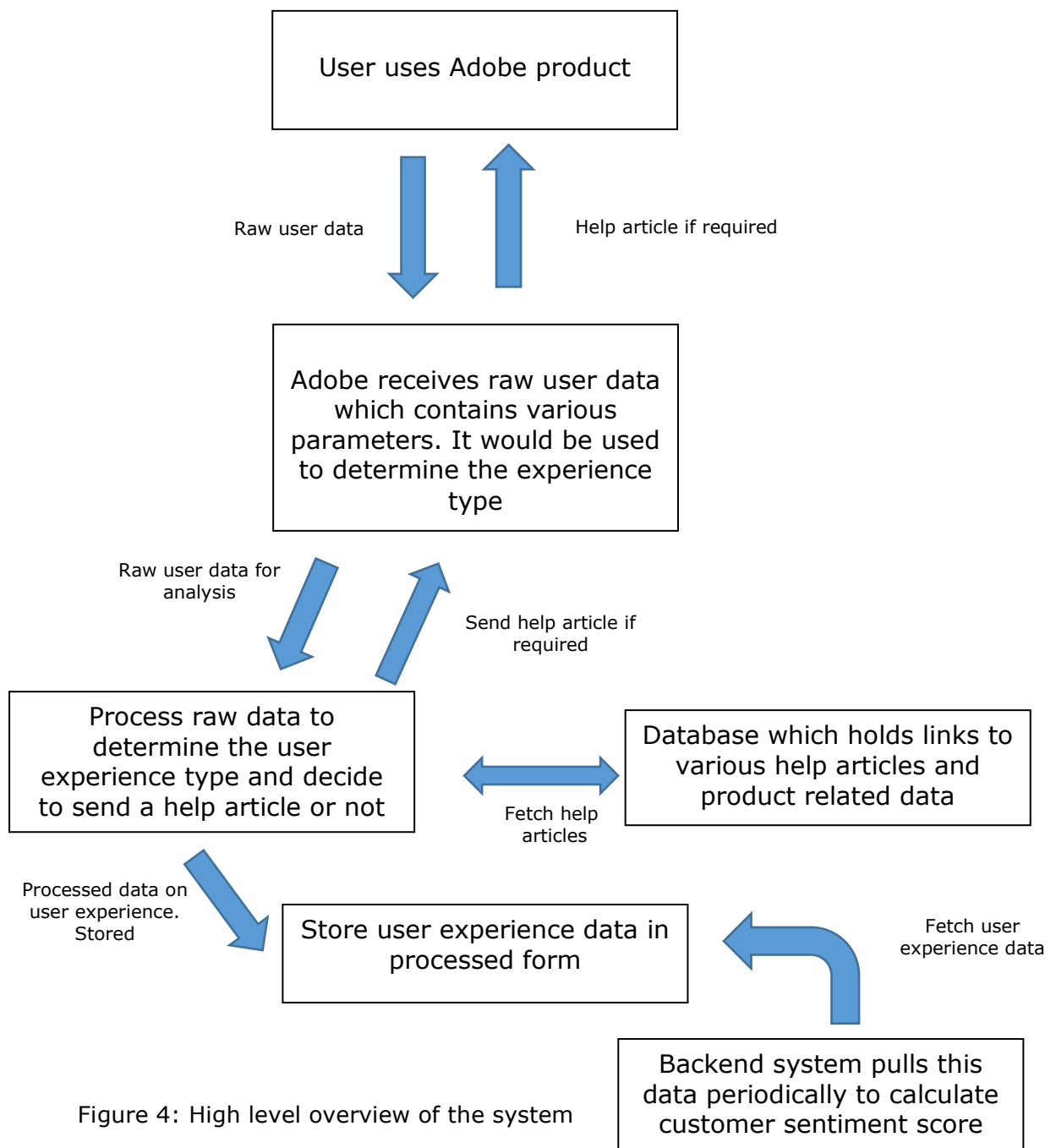


Figure 4: High level overview of the system

The user uses Adobe's product as usual. In the background, we capture the user usage data silently and send it to our publicly exposed endpoint. We do this by designing an agent into the product. This agent would send the data to the publicly exposed endpoint over the internet using REST calls.

Once we get the data, we will extract all the relevant data and send it to a backend server for analysis. This server will look over the sent data and determine if the operation done by the user in a given scenario can be categorized as good user experience or bad user experience. This layer will mark a good user experience as +1 and a bad user experience as -1 for a given feature of an operation. For eg: If a user is performing operation C, how was the experience of the user over various features i.e. Did the user choose the optimal path to reach the option, was the outcome of the operation a success or failure.

If the user experience is determined to be bad, this server will send a request to another system which will return the relevant help article. On receiving the help articles, this server will send its response back to the server which had sent the customer raw data, with the help articles. In case a good experience is detected, the server will respond to the calling system stating that no action is required.

After determining the type of experience i.e. good or bad for a given event, the backend system will send this processed data to another system which will store this information for customer sentiment score calculation. This data is not fully processed. It just indicates if a user had a good or bad experience using a feature for the given operation. It can be possible that the user did not have a good experience reaching the option but had a successful end operation. So, the data will be stored for each feature we are monitoring.

Periodically, a system will fetch the user experience data and would calculate the customer sentiment score based on the following:

1. Feature-wise based on user level
2. Operation wise based on user level
3. Overall customer sentiment score

Here we are calculating the sentiment of each option built into the product based features and customer level. This helps us in understanding how the customer feels using our product at a very granular level.

The following table categorizes scores to customer sentiments:

Score	Customer Sentiment
Above 1.0	Extremely Happy
Between 0.7 to 1.0	Happy
Between -0.7 to 0.7	Neutral
Between -1.0 to -0.7	Disappointed
Below -1.0	Sad

Table 2: Showing the relationship between score and customer sentiment

As we calculate the score, it is finally stored into another system which is used for reporting the customer sentiments. Using this data, Adobe can know well in advance if any customer is facing difficulties using its product and can reach out to them promptly. This will help in delivering a positive customer experience which leads to more business.

Details on each part of the system will be covered in subsequent chapters.

Chapter 3: UI and Building the agent into the application

To capture user data, it is essential that we build an agent into our application with the sole purpose of monitoring user activities and communicating the same to the servers responsible for collecting this data. The agent should capture this data in raw format and should not apply any sort of processing on it. If the agent does some processing on the data, it will impact the application performance which is not desired. The agent should be designed in such a way that it does not consume much resources and the user should not be able to notice it. It should run quietly and efficiently.

Following the modern trend, Adobe plans to provide all of its solutions over the web and run the algorithms on its server. The advantage of this approach is that the company would have tighter control on the experience of its products, can push updates frequently and can provide other services. Given the scenario, for this project, a mock UI was built using HTML, CSS and JS. Therefore, the agent was built in JS.

The mock ui is based on Adobe's UI standard which is:

1. Menu based system for navigation
2. Dialog boxes for operations.

The mock ui provides 39 functions which we will call as operations. These are the end operations a user can perform with the application. As this is a mock ui, few features were built in, like the ability to provide a username, mode of application, end operation status, end operation delay i.e. time taken for processing and capturing the user experience level. These features help in demonstrating various scenarios a user may face while working with the application. The below figure is the snapshot of the mock ui built for this project. This UI uses bootstrap for styling and JQuery for constructing the menu system and performing other operations.

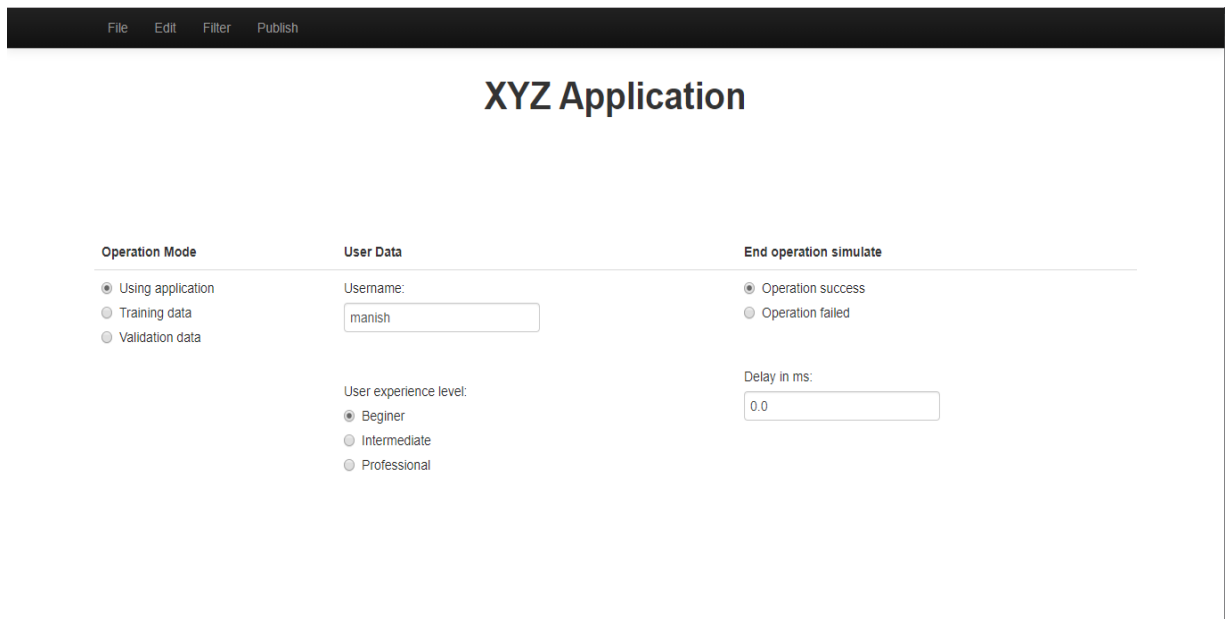


Figure 5: The initial load of the UI

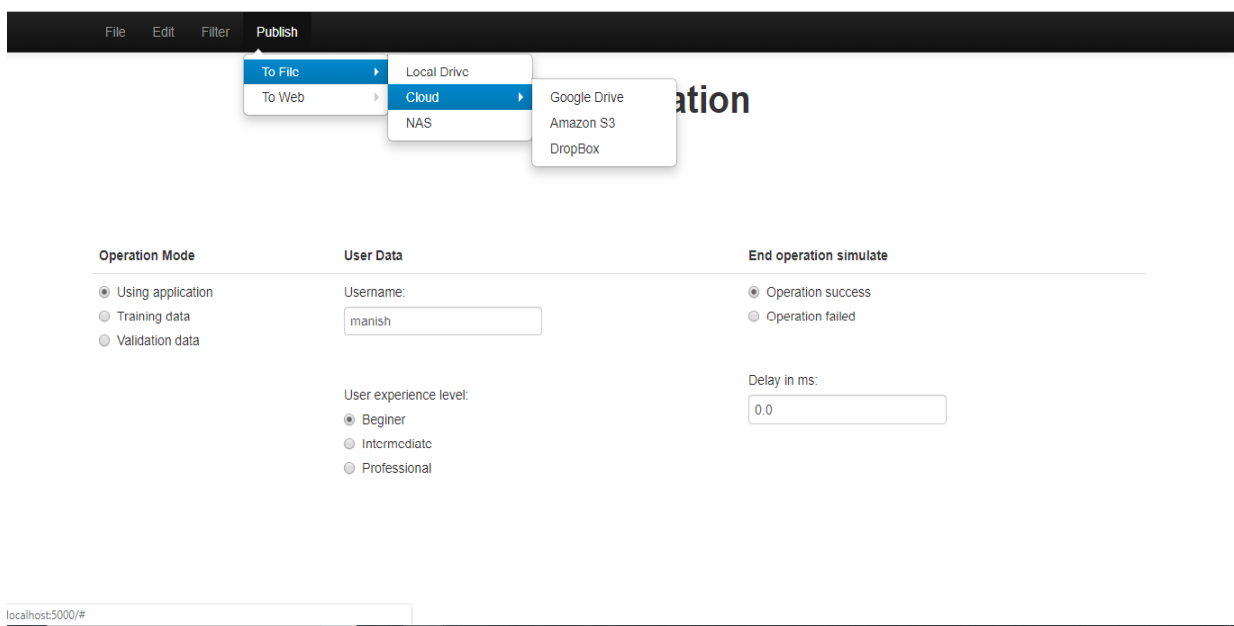


Figure 6: The UI with the menu system

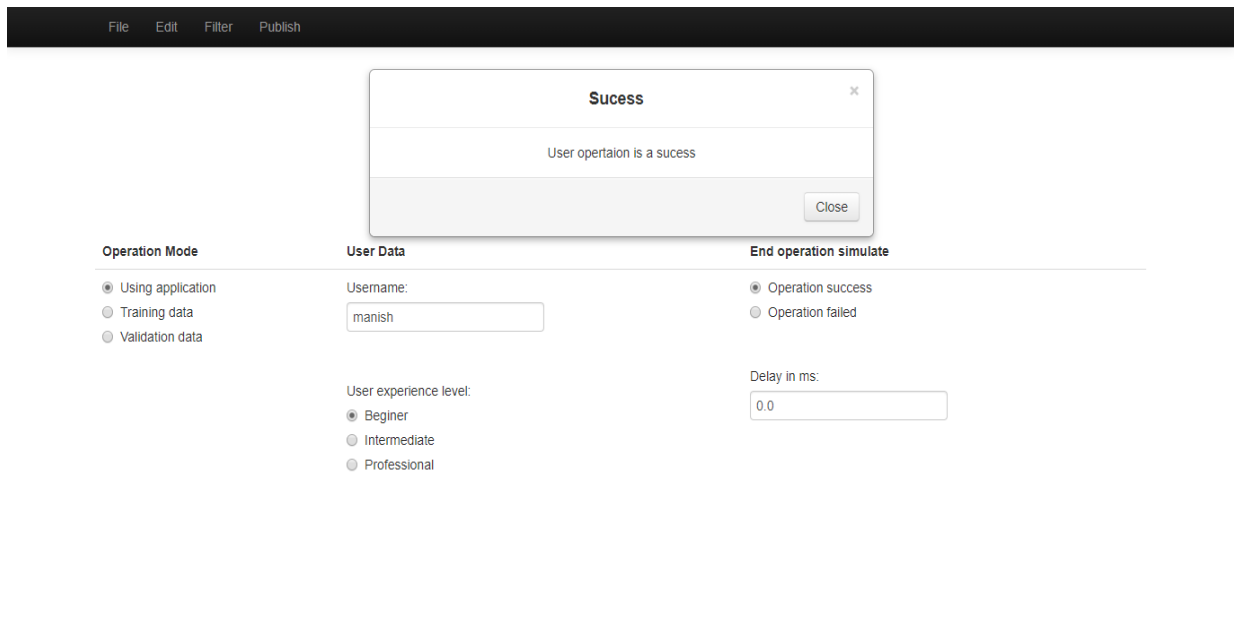


Figure 7: Dialog box showing the result of end operation

The UI covers all actions a user can perform i.e. selecting an operation, operation end result, delay in user operation once OK was clicked.

In this UI, we also capture the username so that we can store individual user experience when they are using the application.

In the UI, we have inbuilt our agent. It has 3 jobs as listed below:

1. Monitor user usage of the application and send it to the server
2. Enable collecting of training data for the model
3. Enable collecting of validation data for the model

The agent is written in JS and would capture the following in any mode i.e. monitoring user usage data, collecting training data or collecting validation data:

1. Capture the pointing device path from start to end when selecting any operation
2. The selected operation

In the Monitoring user usage data mode, the agent will capture the following additional data:

1. User experience level
2. End operation status
3. Time taken for the operation to complete
4. Company Name

To understand when a user is starting an operation, we start capturing the mouse path only when the user clicks the menu items. While capturing the pointing device path, the agent marks the pixel the pointed device touched as White and remaining areas are kept as black. This helps in only capturing the mouse path. The below code snippet shows how it is done.

The moment a user selects a menu:

```
document.addEventListener('mousemove', mouseUpdate);
```

Capturing mouse path as White:

```
function mouseUpdate() {  
    xpos = window.event.screenX;  
    ypos = window.event.screenY;  
    screenMap[xpos][ypos] = 255;  
}
```

The moment a user selects an operation, we stop recording the mouse path:

```
document.removeEventListener('mousemove', mouseUpdate);
```

All of this data is then sent to the backend server as a REST call. Below figures show various REST calls being sent in different modes:

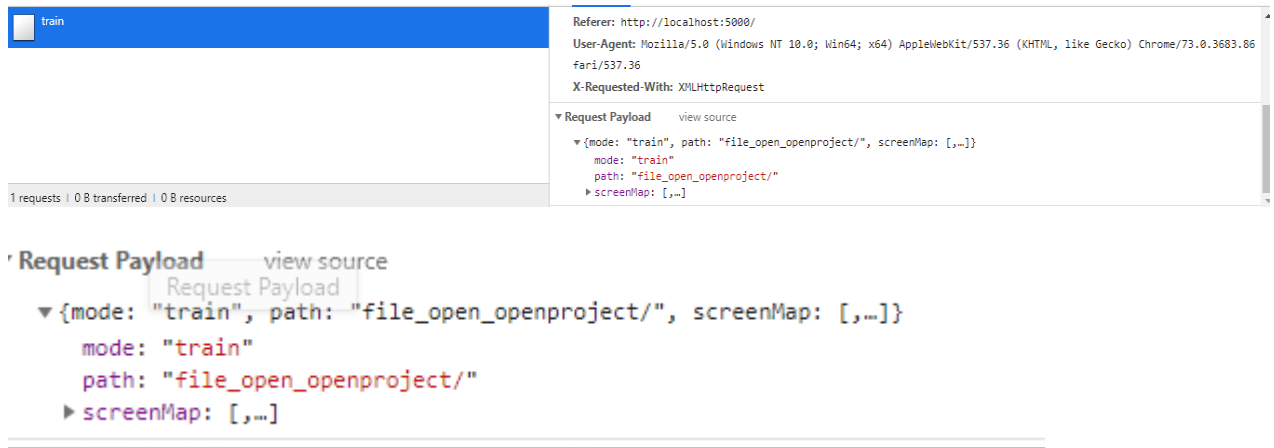


Figure 8: Sending training data to the server

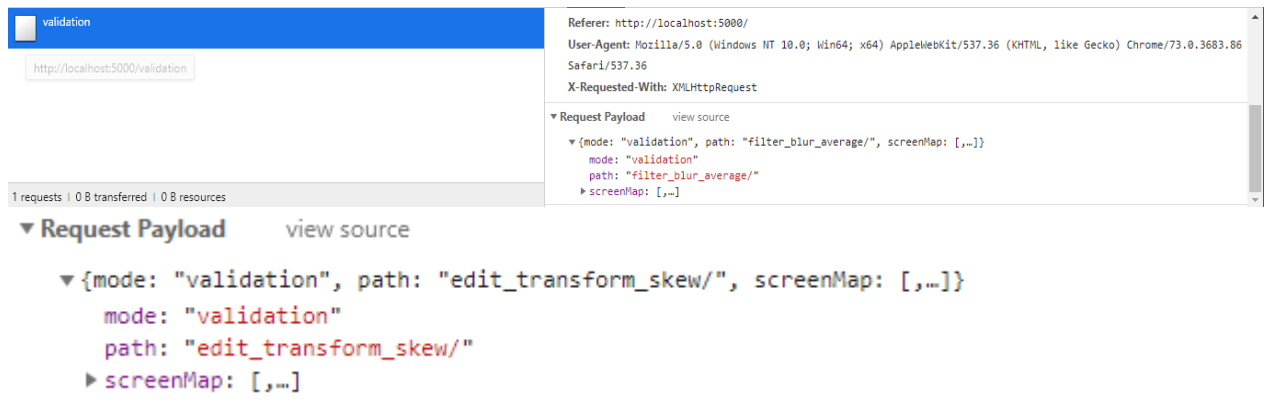


Figure 9: Sending validation data to the server

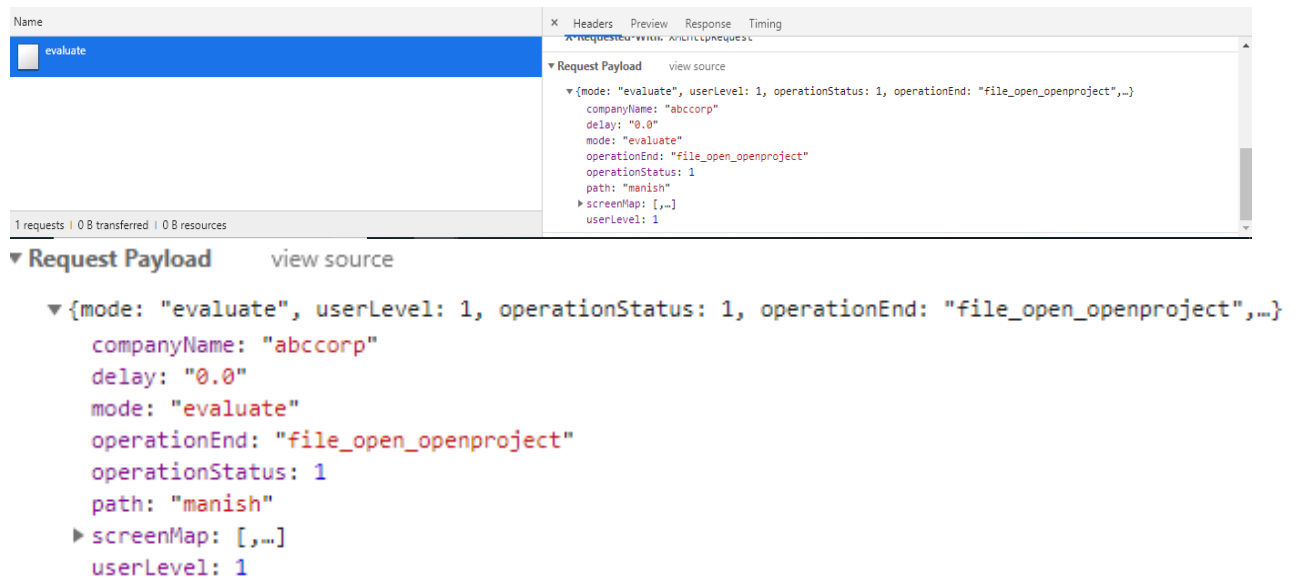


Figure 10: Sending user usage data to the server

For recording the mouse path, the agent determines the screen resolution and then only colors those pixels in white where the mouse passed over. The coloring of white is done by recording the value as 255. Thus, the agent only transfers a 2D array and not an image

The agent does not consume much of the resource and does not interfere with the performance of the application.

Chapter 4: Processing the RAW data to determine the user experience

4.1 Overview

Once the agent sends in the raw data we have to process it to determine the user experience on the features we are monitoring. We are monitoring the following features to determine customer sentiment:

1. The path the user took to select an operation
2. The outcome of the selected operation
3. The time it took to complete the operation

We would need to calculate the customer experience as +1 for good and -1 for bad on these features individually for each user.

4.2 Determining if the user took an optimal path or not for selecting performed operation

When we have to analyze what path the user took to reach the desired operation on the menu system, the conventional way is to simply note down the options which were highlighted and monitor the time from start to finish. Though this method is good it has 2 shortcomings:

1. It does not record the actual path of the pointing device. Its entirely possible that though the user selected the options correctly, the path of the pointing device swayed a lot which can suggest that the user is not sure if they are following the correct path. The traditional method will miss on this part.
2. Sometimes a user selected the right options and right path but due to some external impedances, they leave the mouse in between and return after some time to complete the action. The traditional method would record this as a delay in reaching out to the desired operation.

For eg: A user selects the path

A->B->C

to reach option C. The path chosen is correct but while the user was at Option B, they had to answer a colleague. So, classifying this case as a bad user experience may not be correct.

However, if we can train the computer to tell us if the path followed by the user was the optimal one for the given operation, we can very quickly and accurately predict the customer experience.

To achieve this, we design a Convolutional Neural Network (CNN). The idea is to train the network on identifying what a good path would look like for a given operation and then present the usage pattern of the user and ask it to predict which operation did the user opt for. If the prediction matches with what the user had chosen, we categorize it as a good experience else we categorize it as a bad experience. A similar concept was used with text-based sentiment analysis using different methods as described in [4]

4.2.1 Designing the Convolutional Neural Network

To determine if the path followed by the user is a good one or not, we turn the mouse path into a black and white image where white represents the mouse path. To do so, we process the raw data received in python. We retrieve the 2D array sent by the agent and convert it into a Numpy array. We then use Pillow and convert this array into an image. The code snippet presented below shows how we convert a 2D array into an image.

```
array=content['screenMap']
np_array=np.array(array).astype(np.uint8)
path=os.getcwd()+'/data/training/'+content['path']
img = Image.fromarray(np_array.T)
os.makedirs(os.path.dirname(path), exist_ok=True)
img.save(path+'/'+str(uuid.uuid1())+'.bmp')
```

It is very important to collect data for training the neural network. Therefore, we need to collect good mouse paths for training the model. We collect images as a training set and validation set individually for each operation. The ratio for collecting such images is 70% of images for training and 30% for validation. Once the 2D array is converted into an image, it is stored under the folder which is named as per the operation for which this image was sent. For eg, If the mouse path was for sent for File Close operation as a training image, it would store the image under

data\training\file_close. Similarly, validation images are stored under validation folder. Eg: data\validation\file_close. Before storing an image, each image is given a UUID name so that we can store as many images as possible. Below are some sample images stored as good paths for each operation:

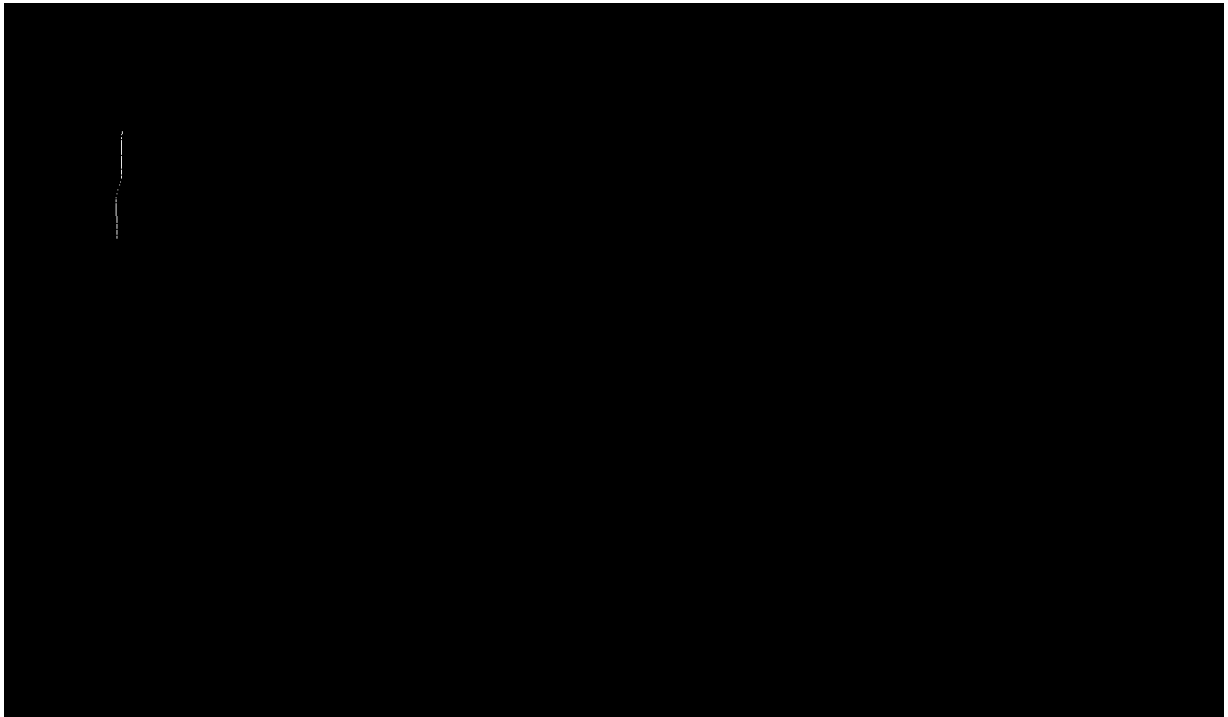


Figure 11: A good mouse path for File close operation

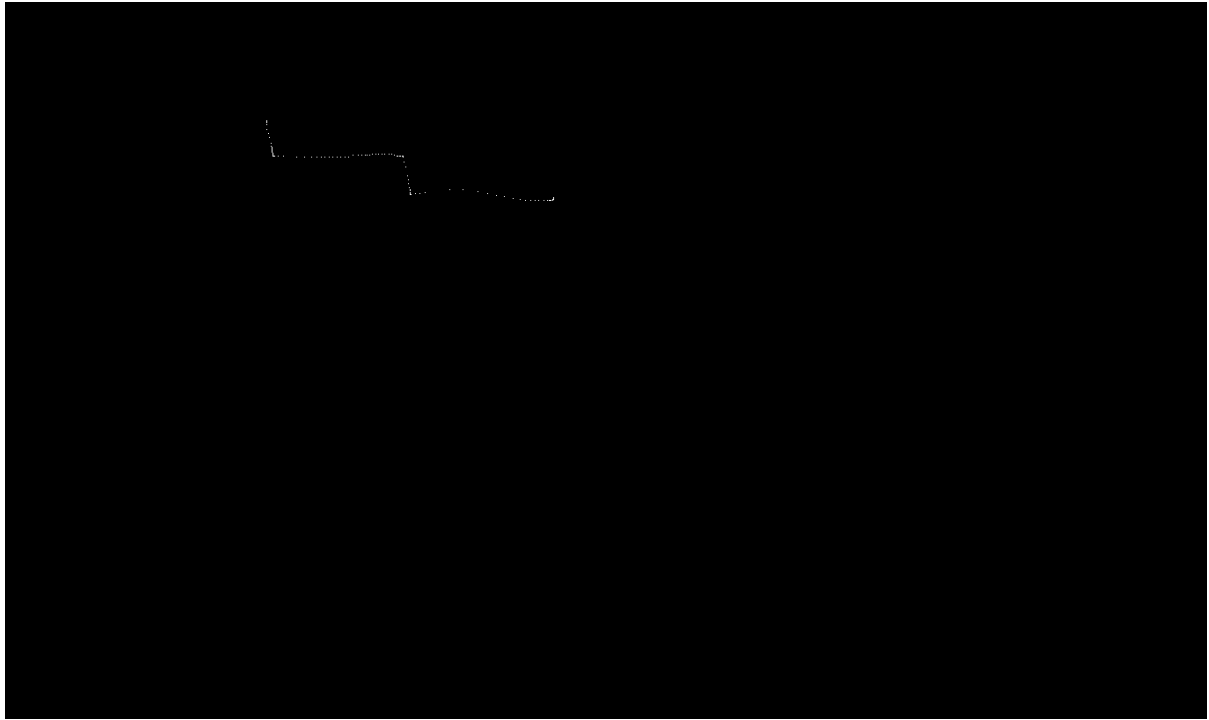


Figure 12: A good mouse path for Saving to google drive operation

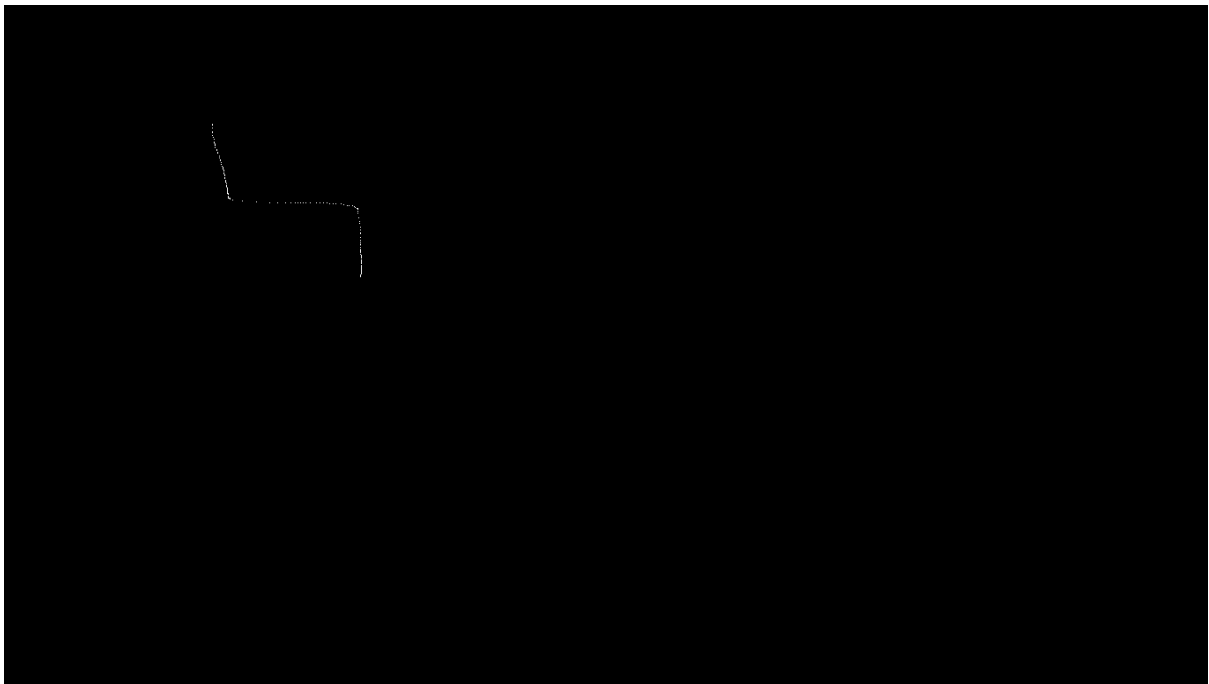


Figure 13: A good mouse path for applying the diffuse filter operation

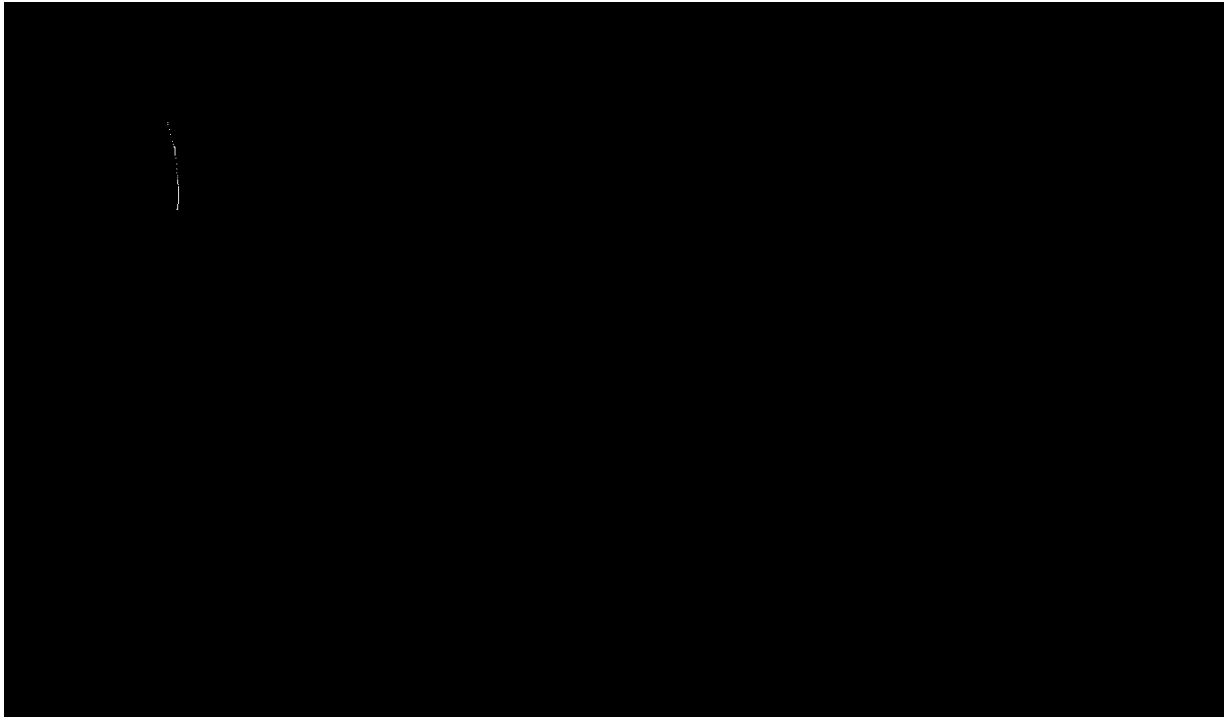


Figure 14: A good mouse path for applying fill operation

We can have different test users send us training and validation images which can help us build a big database of good patterns. The test users used are from Adobe itself and these tests are conducted under a controlled environment. We don't expect our customers to send us the test and validation images.

We use the TensorFlow library to build our CNN [3]. TensorFlow is a highly accepted library and has good support for everything required to build a CNN. After collecting around 20 training images and 6 validation images for each operation, a training and validation generator is created. As our UI contains 39 operations, we have 39 folders created in the training data and validation data directory. These 39 folders would act as 39 classes. Below is the code for the training generator. The validation generator has a similar code.

```
train_generator = train_datagen.flow_from_directory(  
    dataPath+'training',  
    target_size=(300, 300),  
    color_mode='grayscale',  
    shuffle=False,  
    batch_size=4,  
    class_mode='categorical')
```

In the training generator, we convert all images to 300x300. This is done because when we get the images from the agent; they are either in HD resolution or Full HD resolution. Our input images have a majority of black pixels with whites representing the mouse oath. Converting all images into 300x300 ensures that the neural network sees all images in the same size. The value of 300x300 was achieved by running various tests with different sizes. Higher dimensions images were consuming a lot of memory while training and anything lower than this was compromising on the accuracy of the model. We experimented with the following sizes : 600x600, 500x500, 400x400, 200x200, 150x150

As our input images are black and white, we set the color mode of the generator as grayscale. We are taking 20 images for training so the batch size was chosen as 4. This would ensure that our network takes 4 images at a time and train itself. These values can be further optimized as we get more data and higher hardware. The purpose is to get a model with higher accuracy.

Once the training and validation generator are configured, we designed a CNN. As recommended in [7], we took 2 levels of convolutions and max pooling. A single level of convolution and maxpooling resulted in a larger output shape and the accuracy was not good. 3 or more layers of convolution and max-pooling did not help much. Even before passing the images to the generators, we had rescaled each pixel to have a value in between 0 and 1. As we only have black or white, each pixel representation changed to 0 (for black) or 1 (for white). We did this using Keras image preprocessing. Below line of code shows the same:

```
train_datagen = ImageDataGenerator(rescale=1/255)
```

As all pixels are either 0 or 1, we took the activation function as ReLU for the convolution. ReLU is helpful in this scenario as relu function returns 0 as output if the input is 0 or less else it returns X if X is positive. So we get 1 if a white is encountered else a 0. The mathematical function of ReLU is:

$$A(x) = \max(0, x)$$

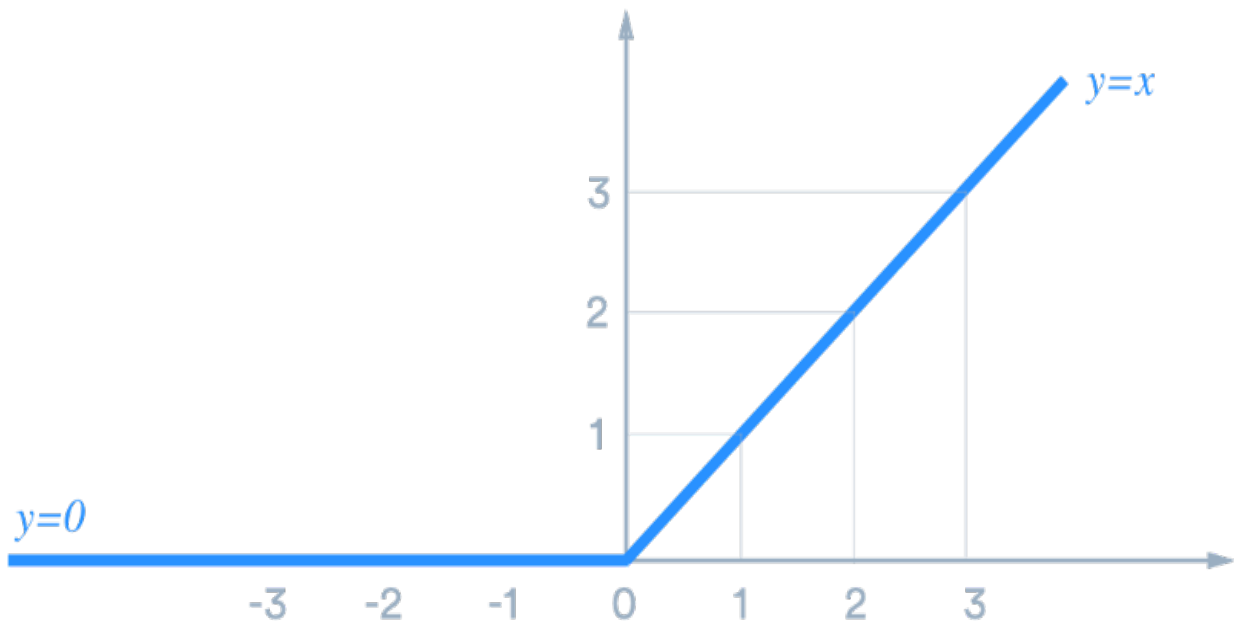


Figure 15: Graphical representation of ReLU function

The other advantages of choosing ReLU is that it is cheap to compute, no special hardware is required and is better than sigmoid & tanh as presented by Glorot, Bordes and Bengio in [6].

As shown in [7], applying random filters improves feature extraction. In our convolution stage, we apply 16 random filters of size 3x3. With the input set we had; 16 filters seemed sufficient. After the convolution, we applied Max pooling of size 2x2, this helped in downsampling the input images while not affecting the image feature. Downsampling the image helps in reducing the computational efforts in training the neural network.

As recommended in [7], we apply 2 layers of convolutions and max-pooling and then the images are passed to the neural network of 512 neurons. Here also we use ReLU as the activation function. 512 neurons act as a hidden layer between the input and output layers. These neurons are connected to 39 neurons which represent 39 classes or 39 operations we support in our UI. At the output layer, we use softmax as the activation function. Softmax function gives us a normalized probability distribution consisting of K probabilities. The mathematical representation of softmax is:

$$S(y_i) = (e^{y_i}) / \sum e^{y_j}$$

Softmax helps in predicting which class does our input belongs to. Below is the code of the CNN written in Tensorflow:

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(300, 300,
1)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(train_generator.num_classes, activation='softmax')
])
```

For our CNN, we use **Adam optimizer algorithm**. It is computationally efficient and has lesser memory requirements as presented by Kingma, J. and Jimmy in [5]. For loss calculation, we use **categorical_crossentropy** as we have multiple classes. Below is the code where we define the loss and optimizer:

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

To train the model, we have set 15 epochs (iterations) with 28 steps per epoch. Same parameters are used for validation. These values can be optimized based on the input size and by trying various values. Below is the code which shows the fit generator:

```
history = model.fit_generator(
    train_generator,
    steps_per_epoch=28,
    epochs=15,
    validation_data = validation_generator,
    validation_steps=28,
    callbacks=[callbacks])
```

Once we touch 90% accuracy, we stop the training. This is done by implementing a callback which checks the accuracy of the model after each epoch. The code for the callback is mentioned below:

```
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('acc')>0.90):
            print("\nReached 90.0% accuracy so cancelling training!")
            self.model.stop_training = True
```

Based on this configuration, our model summary comes out to be:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 298, 298, 16)	160
max_pooling2d (MaxPooling2D)	(None, 149, 149, 16)	0
conv2d_1 (Conv2D)	(None, 147, 147, 16)	2320
max_pooling2d_1 (MaxPooling2D)	(None, 73, 73, 16)	0
flatten (Flatten)	(None, 85264)	0
dense (Dense)	(None, 512)	43655680
dense_1 (Dense)	(None, 39)	20007
Total params: 43,678,167		
Trainable params: 43,678,167		
Non-trainable params: 0		

Figure 16: Model summary of the CNN

While training the model with the above-mentioned parameters, we get 90% accuracy in 5-9 epochs. The figure below shows the model reaching 90% accuracy

```

Epoch 1/15
17/17 [=====] - 3s 172ms/step - loss: 3.5216 - acc: 0.1642
31/31 [=====] - 41s 1s/step - loss: 3.4973 - acc: 0.0984 - val_loss: 3.5216 - val_acc: 0.1642
Epoch 2/15
17/17 [=====] - 2s 144ms/step - loss: 2.7513 - acc: 0.2537
31/31 [=====] - 43s 1s/step - loss: 2.4021 - acc: 0.4016 - val_loss: 2.7513 - val_acc: 0.2537
Epoch 3/15
17/17 [=====] - 3s 160ms/step - loss: 2.5476 - acc: 0.2836
31/31 [=====] - 38s 1s/step - loss: 1.3576 - acc: 0.6148 - val_loss: 2.5476 - val_acc: 0.2836
Epoch 4/15
17/17 [=====] - 2s 132ms/step - loss: 2.3509 - acc: 0.3582
31/31 [=====] - 38s 1s/step - loss: 0.7056 - acc: 0.8115 - val_loss: 2.3509 - val_acc: 0.3582
Epoch 5/15
17/17 [=====] - 2s 137ms/step - loss: 2.7704 - acc: 0.2836

Reached 90.0% accuracy so cancelling training!
31/31 [=====] - 38s 1s/step - loss: 0.3525 - acc: 0.9098 - val_loss: 2.7704 - val_acc: 0.2836
Trained model saved

```

Figure 17: Training the model

Once the model is trained, we save it. Below is the code for saving the model:

```
tf.keras.models.save_model(model,"D://MS//BITSPilani//Sem4//project//main//
ui//data//model/myModel.h5", overwrite=True, include_optimizer=True)
```

When the agent from the UI sends the mouse path data, we evaluate it using the saved model and observe the class predicted. If the class predicted matches with the action taken by the users, it is concluded that the user had a good experience, else a bad experience is recorded. The figures below show the mouse pattern received from the user and its evaluation.

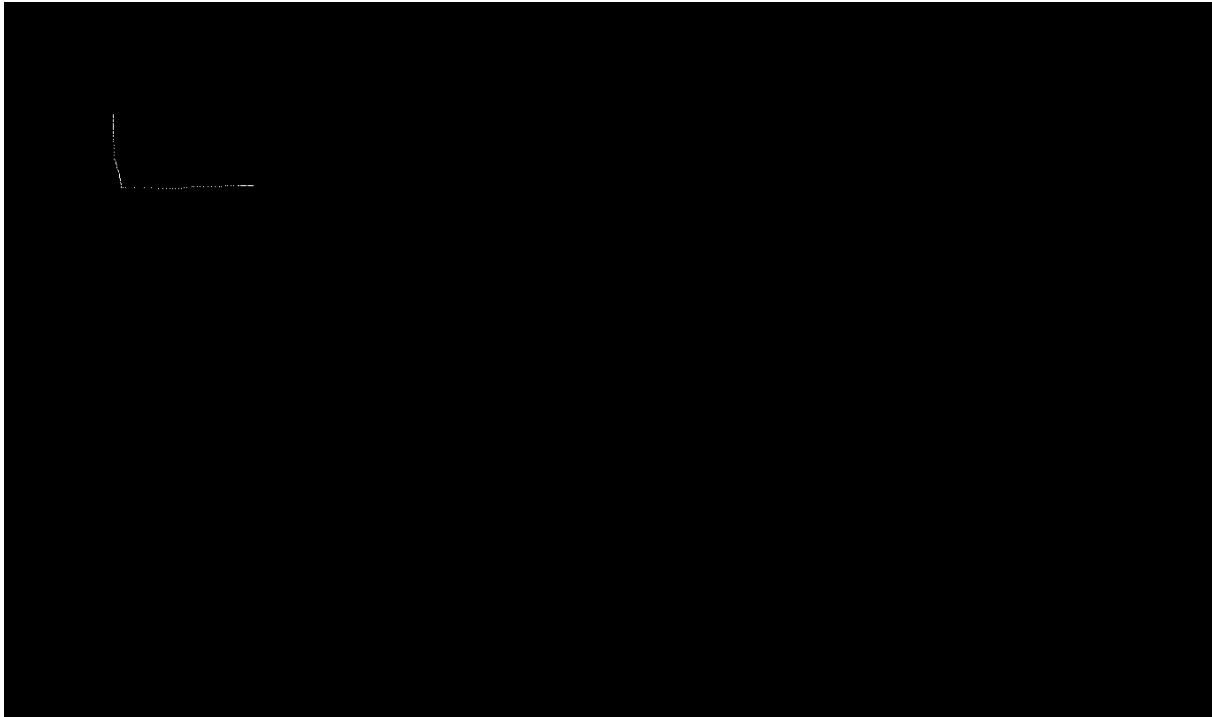


Figure 18: Showing a user pattern received

```

Action detected: edit_assignprofile
{'edit_assignprofile': 0, 'edit_fill': 1, 'edit_preferences_cameraraw': 2, 'edit_preferences_general': 3, 'edit_preferences_tools': 4, 'edit_transform_distort': 5, 'edit_transform_rotate': 6, 'edit_transform_scale': 7, 'edit_transform_skew': 8, 'file_close': 9, 'file_new_newfile': 10, 'file_new_newproject': 11, 'file_open_openfile': 12, 'file_open_openproject': 13, 'file_properties': 14, 'file_save': 15, 'filter_blur_average': 16, 'filter_blur_gaussianblur': 17, 'filter_blur_lensblur': 18, 'filter_blur_radialblur': 19, 'filter_distort_displace': 20, 'filter_distort_pinch': 21, 'filter_distort_twirl': 22, 'filter_noise_addnoise': 23, 'filter_noise_diffusenoise': 24, 'filter_noise_reducenoise': 25, 'filter_stylize_diffuse': 26, 'filter_stylize_emboss': 27, 'filter_stylize_oilpaint': 28, 'filter_stylize_tiles': 29, 'filter_stylize_wind': 30, 'publish_tofile_cloud_amazons3': 31, 'publish_tofile_cloud_dropbox': 32, 'publish_tofile_cloud_googledrive': 33, 'publish_tofile_localdrive': 34, 'publish_tofile_nas': 35, 'publish_toweb_facebook': 36, 'publish_toweb_instagram': 37, 'publish_toweb_youtube': 38}
[0]
[[1.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 1.1686238e-22 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00]]

```

Figure 19: The input mouse path being analyzed

If the action detected is the one the user opted for then we grade the experience as +1 (good) else we mark it -1 (bad)

4.2.2 Processing the outcome and delay in doing operation

The agent also sends us the outcome of the operation and the time the application took in processing the operation. If the operation is a success then we enter the experience as +1 for good else enter the experience as -1 for bad. The agent simply sends the outcome as 1 or 2 where 1 means the operation is a success and 2 means that the operation is a failure.

The agent also sends us the time the operation takes to complete. It reports the time in milliseconds. While processing the time taken, we look up for the recommended time an operation should take. If it is more than the expected time then we mark it as a bad experience (-1). If it is equal or less we mark it as a good experience (+1). The time taken for each operation is benchmarked in the testing phase against the minimum required hardware specifications.

Chapter 5: The backend storage

5.1 Overview

Once we have evaluated all the features from a user action, we need to store it in a backend system from where we can fetch this data and calculate the customer sentiment score. For this we use MongoDB. MongoDB allows us to use NoSQL and store data in JSON format. The advantage here is that we don't need to follow any schema and can modify our storage structure easily and whenever we want, giving us the flexibility in modifying the storage structure without worrying about the previously stored data.

Also, we need to maintain a list of help articles in a backend database. This database is queried whenever we need to fetch the list of help articles to be sent back to the user. For the same, we again use MongoDB and store this information in a collection.

5.2 Storing product information

To store product related information, we make a database named **product**. In the product database, we have two collections:

1. features
2. helpdocs

5.2.1 Features collection

Features contain the list of operations the UI has. This is useful whenever we want to compute any report and want to know what operations does the application has. The format of the storage schema for features collection is:

Field name	Description
_id	Unique key for the record
event	Stores the operation name

Table 3: Schema for features collection

```
MongoDB Enterprise > db.features.find()
{ "_id" : ObjectId("5c994b43f8fa0615e8b050d9"), "event" : "file_new_newproject" }
{ "_id" : ObjectId("5c994ba1f8fa0615e8b050da"), "event" : "file_new_newfile" }
{ "_id" : ObjectId("5c994baff8fa0615e8b050db"), "event" : "file_open_openproject" }
{ "_id" : ObjectId("5c994bb4f8fa0615e8b050dc"), "event" : "file_open_openfile" }
{ "_id" : ObjectId("5c994bbaf8fa0615e8b050dd"), "event" : "file_save" }
{ "_id" : ObjectId("5c994bbff8fa0615e8b050de"), "event" : "file_close" }
{ "_id" : ObjectId("5c994bc3f8fa0615e8b050df"), "event" : "file_properties" }
{ "_id" : ObjectId("5c994be4f8fa0615e8b050e0"), "event" : "edit_transform_skew" }
{ "_id" : ObjectId("5c994be9f8fa0615e8b050e1"), "event" : "edit_transform_distort" }
{ "_id" : ObjectId("5c994bf1f8fa0615e8b050e2"), "event" : "edit_transform_scale" }
{ "_id" : ObjectId("5c994bf5f8fa0615e8b050e3"), "event" : "edit_transform_rotate" }
{ "_id" : ObjectId("5c994bfd8fa0615e8b050e4"), "event" : "edit_assignprofile" }
{ "_id" : ObjectId("5c994bff8fa0615e8b050e5"), "event" : "edit_fill" }
{ "_id" : ObjectId("5c994c04f8fa0615e8b050e6"), "event" : "edit_preferences_general" }
{ "_id" : ObjectId("5c994c07f8fa0615e8b050e7"), "event" : "edit_preferences_cameraraw" }
{ "_id" : ObjectId("5c994c0af8fa0615e8b050e8"), "event" : "edit_preferences_tools" }
{ "_id" : ObjectId("5c994c0df8fa0615e8b050e9"), "event" : "filter_blur_average" }
{ "_id" : ObjectId("5c994c11f8fa0615e8b050ea"), "event" : "filter_blur_radialblur" }
{ "_id" : ObjectId("5c994c15f8fa0615e8b050eb"), "event" : "filter_blur_gaussianblur" }
{ "_id" : ObjectId("5c994c19f8fa0615e8b050ec"), "event" : "filter_blur_lensblur" }
```

Figure 20: Showing the feature list in MongoDB

5.2.2 Helpdocs collection

Helpdocs contain the list of help articles for each operation the application supports. As we primarily monitor the user mouse path and operation outcome, we store help articles based on individual features we monitor. These articles are sent to the user when required. The format of the storage schema for helpdocs collection is:

Field name	Description
_id	Unique key for the record
event	Stores the operation name
type	Feature we monitor like path, end operation
doclink	The URL for the help article

Table 4: Schema for helpdocs collection

```
MongoDB Enterprise > db.helpdocs.find()
{ "_id" : ObjectId("5c9954d7f8fa060e88075c3f"), "event" : "file_new_newproject", "type" : "end", "doclink" : "http://file_new_newproject_end" }
{ "_id" : ObjectId("5c9954d7f8fa060e88075c40"), "event" : "file_new_newproject", "type" : "path", "doclink" : "http://file_new_newproject_path" }
{ "_id" : ObjectId("5c9954dddf8fa060e88075c41"), "event" : "file_new_newfile", "type" : "end", "doclink" : "http://file_new_newfile_end" }
{ "_id" : ObjectId("5c9954dddf8fa060e88075c42"), "event" : "file_new_newfile", "type" : "path", "doclink" : "http://file_new_newfile_path" }
{ "_id" : ObjectId("5c9954f0f8fa060e88075c43"), "event" : "file_open_openproject", "type" : "end", "doclink" : "http://file_open_openproject_end" }
{ "_id" : ObjectId("5c9954f0f8fa060e88075c44"), "event" : "file_open_openproject", "type" : "path", "doclink" : "http://file_open_openproject_path" }
{ "_id" : ObjectId("5c9954f4f8fa060e88075c45"), "event" : "file_open_openfile", "type" : "end", "doclink" : "http://file_open_openfile_end" }
{ "_id" : ObjectId("5c9954f4f8fa060e88075c46"), "event" : "file_open_openfile", "type" : "path", "doclink" : "http://file_open_openfile_path" }
{ "_id" : ObjectId("5c9954f7f8fa060e88075c47"), "event" : "file_save", "type" : "end", "doclink" : "http://file_save_end" }
{ "_id" : ObjectId("5c9954f7f8fa060e88075c48"), "event" : "file_save", "type" : "path", "doclink" : "http://file_save_path" }
{ "_id" : ObjectId("5c9954faf8fa060e88075c49"), "event" : "file_close", "type" : "end", "doclink" : "http://file_close_end" }
{ "_id" : ObjectId("5c9954faf8fa060e88075c4a"), "event" : "file_close", "type" : "path", "doclink" : "http://file_close_path" }
{ "_id" : ObjectId("5c9954fdf8fa060e88075c4b"), "event" : "file_properties", "type" : "end", "doclink" : "http://file_properties_end" }
{ "_id" : ObjectId("5c9954fdf8fa060e88075c4c"), "event" : "file_properties", "type" : "path", "doclink" : "http://file_properties_path" }
{ "_id" : ObjectId("5c995508f8fa060e88075c4d"), "event" : "edit_transform_skew", "type" : "end", "doclink" : "http://edit_transform_skew_end" }
{ "_id" : ObjectId("5c995508f8fa060e88075c4e"), "event" : "edit_transform_skew", "type" : "path", "doclink" : "http://edit_transform_skew_path" }
{ "_id" : ObjectId("5c99550cf8fa060e88075c4f"), "event" : "edit_transform_distort", "type" : "end", "doclink" : "http://edit_transform_distort_end" }
{ "_id" : ObjectId("5c99550cf8fa060e88075c50"), "event" : "edit_transform_distort", "type" : "path", "doclink" : "http://edit_transform_distort_path" }
{ "_id" : ObjectId("5c995510f8fa060e88075c51"), "event" : "edit_transform_scale", "type" : "end", "doclink" : "http://edit_transform_scale_end" }
{ "_id" : ObjectId("5c995510f8fa060e88075c52"), "event" : "edit_transform_scale", "type" : "path", "doclink" : "http://edit_transform_scale_path" }
```

Figure 21: Showing the helpdocs in MongoDB

5.3 Storing user usage information

To store user related information, we create a database based on the customer name. As we can have multiple customers, each customer has their own database in the system and all data specific to that customer is stored in that database only.

Each customer will have the following collections:

1. Users
2. Usage
3. Overall score

5.3.1 Users collection

The user collection will hold all the username used by the customer. The entry in this collection is done as soon as we get the username from the agent running inside the UI. The schema for user collection is:

Field name	Description
_id	Unique key for the record
User	Username of the user

Table 5: Schema for user collection

```
MongoDB Enterprise > db.users.find()
{ "_id" : ObjectId("5c967783f8fa064fb8aadb6d"), "user" : "manish" }
{ "_id" : ObjectId("5c9677d4f8fa064fb8aadb6e"), "user" : "manish1" }
```

Figure 22: Showing the users of a customer in MongoDB

5.3.2 Usage collection

The usage collection holds all the user experience data for a given customer. The data from this collection is used to calculate the customer sentiment score. The schema for the usage collection is:

Field name	Description
_id	Unique key for the record
event	Stores the operation name
type	Feature we monitor like path, end operation
experience	+1 for good and -1 for bad
userLevel	The level of the user. 1 for beginner, 2 for intermediate and 3 for professional
username	Username of the user

Table 6: Schema for usage collection

```
MongoDB Enterprise > db.usage.find()
{ "_id" : ObjectId("5c9952eff8fa060edcea43dd"), "username" : "manish", "userLevel" : 1, "type" : "path", "experience" : -1, "event" : "file_close" }
{ "_id" : ObjectId("5c9952eff8fa060edcea43de"), "username" : "manish", "userLevel" : 1, "type" : "end", "experience" : 1, "event" : "file_close" }
{ "_id" : ObjectId("5c995348f8fa060edcea43df"), "username" : "manish", "userLevel" : 1, "type" : "path", "experience" : 1, "event" : "file_close" }
{ "_id" : ObjectId("5c995348f8fa060edcea43e0"), "username" : "manish", "userLevel" : 1, "type" : "end", "experience" : 1, "event" : "file_close" }
```

Figure 23: Showing the usage data of users of a customer in MongoDB

5.3.3 overallscore collection

This collection contains all the calculated score which depict the user sentiment of every operation based on the user level i.e. beginner, intermediate, professional and component i.e. mouse path, end operation outcome, time taken for an operation to complete. This also stores the customer overall sentiment score. The schema for overallscore collection is:

Field name	Description
_id	Unique key for the record
event	Stores the operation name
type	Feature we monitor like path, end operation
experience_score	Experience score of the feature
userLevel	The level of the user. 1 for beginner, 2 for intermediate and 3 for professional

Table 7: Schema for overallscore collection

```
{ "_id" : ObjectId("5c99bfc9959c673891dfc498"), "event" : "file_open_openproject", "type" : "weighted", "experience_score" : 4.25 }
{ "_id" : ObjectId("5c99bfc9959c673891dfc49a"), "event" : "file_open_openproject", "type" : "path", "userLevel" : 1, "experience_score" : 0 }
{ "_id" : ObjectId("5c99bfc9959c673891dfc49c"), "event" : "file_open_openproject", "type" : "end", "userLevel" : 1, "experience_score" : 1 }
{ "_id" : ObjectId("5c99bfc9959c673891dfc49e"), "event" : "file_open_openproject", "type" : "combined", "userLevel" : 1, "experience_score" : 1 }
{ "_id" : ObjectId("5c99bfc9959c673891dfc4a5"), "event" : "file_open_openproject", "type" : "path", "userLevel" : 2, "experience_score" : 1 }
{ "_id" : ObjectId("5c99bfc9959c673891dfc4a7"), "event" : "file_open_openproject", "type" : "end", "userLevel" : 2, "experience_score" : 1 }
{ "_id" : ObjectId("5c99bfc9959c673891dfc4a9"), "event" : "file_open_openproject", "type" : "combined", "userLevel" : 2, "experience_score" : 2 }
{ "_id" : ObjectId("5c99bfc9959c673891dfc4b0"), "event" : "file_open_openproject", "type" : "path", "userLevel" : 3, "experience_score" : 1 }
{ "_id" : ObjectId("5c99bfc9959c673891dfc4b2"), "event" : "file_open_openproject", "type" : "end", "userLevel" : 3, "experience_score" : 1 }
```

Figure 24: Showing the experience score in overallscore collection stored in MongoDB

Chapter 6: Calculating the customer sentiment score

6.1 Overview

To calculate the user sentiment score, we need to analyze the data inside the usage collection. This collection contains the user experience of each operation, feature wise in terms of good or bad. The methodology applied to calculate the customer sentiment score is as follows:

1. Calculate the score of each operation, feature wise, individually for each user level
2. Multiply the weights for each feature of the operation, user level wise, with the score calculated previously for the feature
3. Add the calculated weighted score to get the operation score.
4. Add all the operation scores to get the customer sentiment score

6.2 Calculating the customer sentiment score

To start with, we pick up each operation the product has to offer and calculate the score on each feature we are monitoring. This is done individually for each user level. For eg, if we want to calculate the sentiment score of file close operation, we would first calculate the sentiment score on how the user reached to the option (using a pointing device), then the operation outcome and how much time the operation took. We would calculate this for beginner users, then for intermediate user and then professional users.

To calculate the sentiment score of each feature we use the following formula

$$S_{(feature, userlevel)} = \frac{\text{sum of user experience of that feature for the given user level}}{\text{Number of occurrences of the usage of that feature for the given user level}}$$

We use MongoDB aggregation pipeline to calculate the sentiment score of each feature per user level as shown below:

```
pipeline=[{"$match":{"event":feature,"type":"path","userLevel":x}},{"$group":{"_id":"$event", "sumscore":{"$sum":"$experience"}, "count":{"$sum":1}}},{"$project":{"score":{"$divide":["$sumscore","$count"]}}}]
```

After we get the sentiment score of each feature for every user level, we multiple the feature with the weight of that feature at a given user level. This weight is configurable. The purpose of weights is to influence the overall customer sentiment score by giving higher weights to important features and user levels and lower weights to others. Eg: Operation outcome is more important than navigating to the operation The formula to calculate the feature score is

$$S_{feature} = \sum \text{weight}_{(feature, userlevel)} \times S_{(feature, userlevel)}$$

Once we get the weighted score of each feature, we add them to calculate the sentiment score for each operation. We use the following formula to calculate the operation sentiment score

$$S_{operation} = \sum S_{feature}$$

```
{ "_id" : ObjectId("5c99bfc9959c673891dfc498"), "event" : "file_open_openproject", "type" : "weighted", "experience_score" : 4.25 }
```

Figure 25: Sentiment score of open project operation

To calculate the overall customer sentiment score, we add all the operation scores. The formula for the same is

$$S_{\text{customer sentiment}} = \sum S_{\text{operation}}$$

Based on the score received, we map it to the customer sentiment using the below table.

Score	Customer Sentiment
Above 1.0	Extremely Happy
Between 0.7 to 1.0	Happy
Between -0.7 to 0.7	Neutral
Between -1.0 to -0.7	Disappointed
Below -1.0	Sad

Table 8: Score to customer sentiments mapping

Chapter 7: Integrating the system

7.1 Overview

Till now we have seen the individual components we can use to calculate the customer sentiment score. Now, all components would be integrated to make it into one system which works to update the customer sentiment score constantly.

Below is a system flow diagram which shows how each component would work with each other.

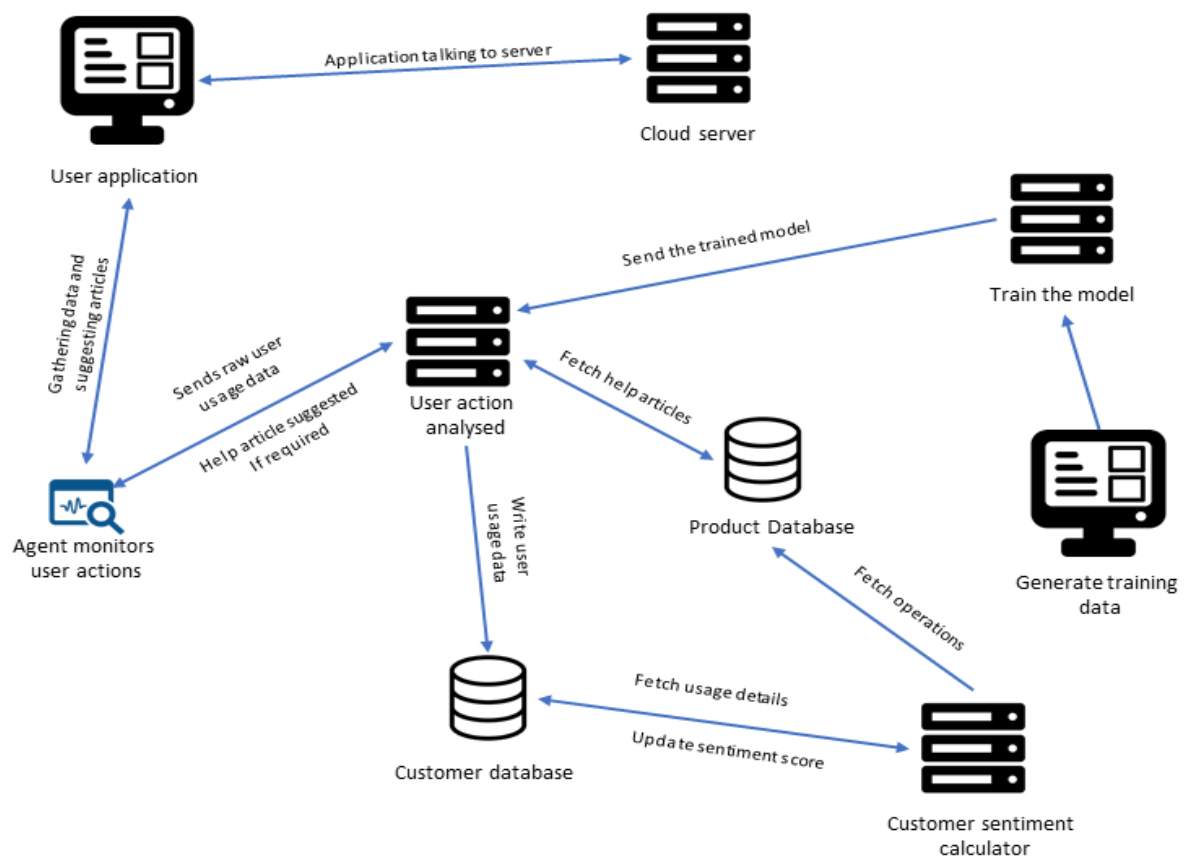


Figure 26: System flow diagram

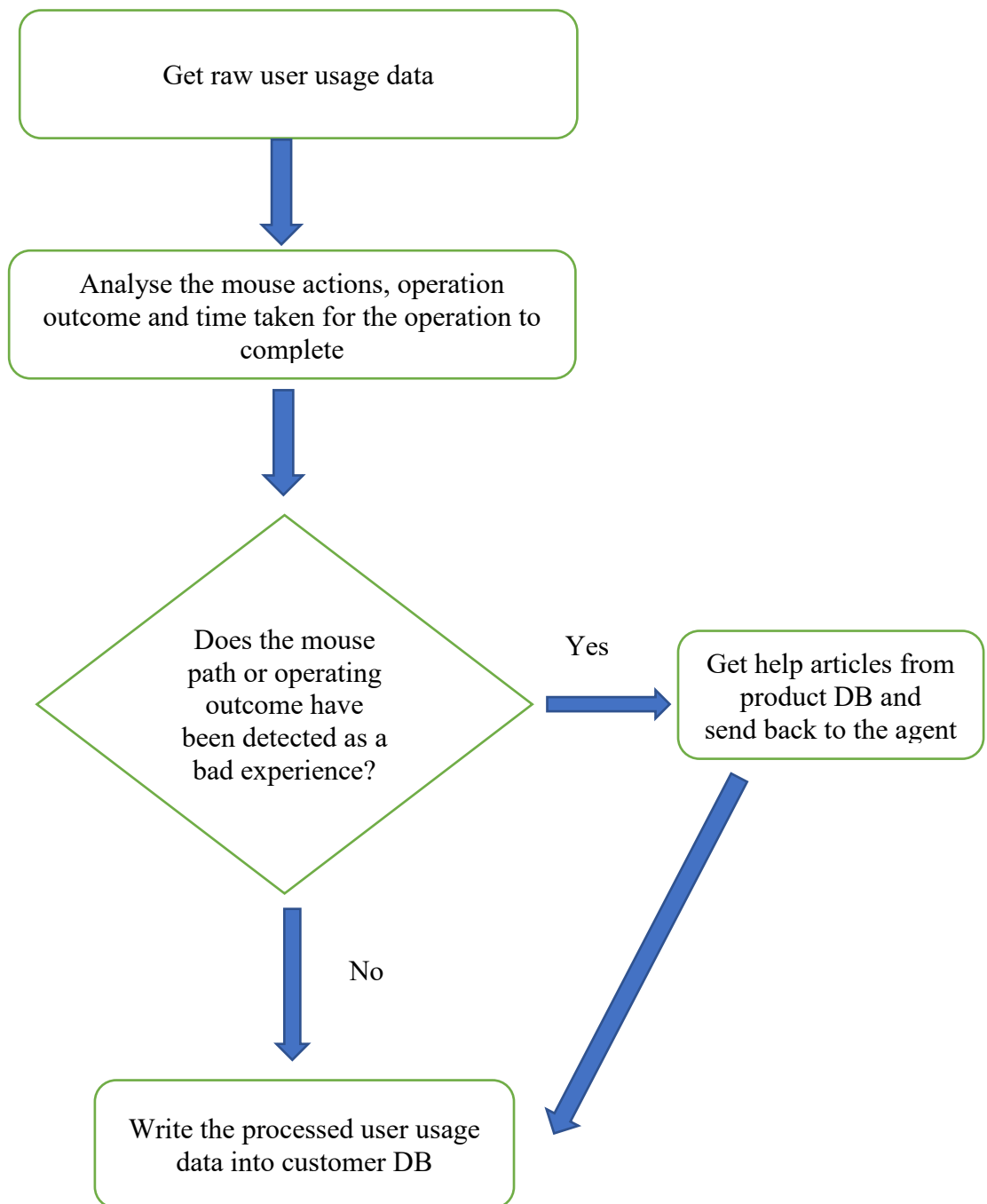
The above diagram clearly shows how each component works in the system to help the company calculate the customer sentiment score while suggesting help articles to the user. The application can be loaded from a cloud server on the computer. An agent keeps monitoring the user actions and sends raw user usage data to a server which analyses the user actions and decides if a user needs to be suggested help articles or not. If help articles are suggested, the server retrieves them from the products database and sends the relevant articles back to the agent which in turns suggest them to the user. The server also writes the user usage processed data into the customer database.

Another server keeps on training the model with new data and when the model is trained, it is transferred to the server which analyses user action. This action can be run every day or on a periodic schedule.

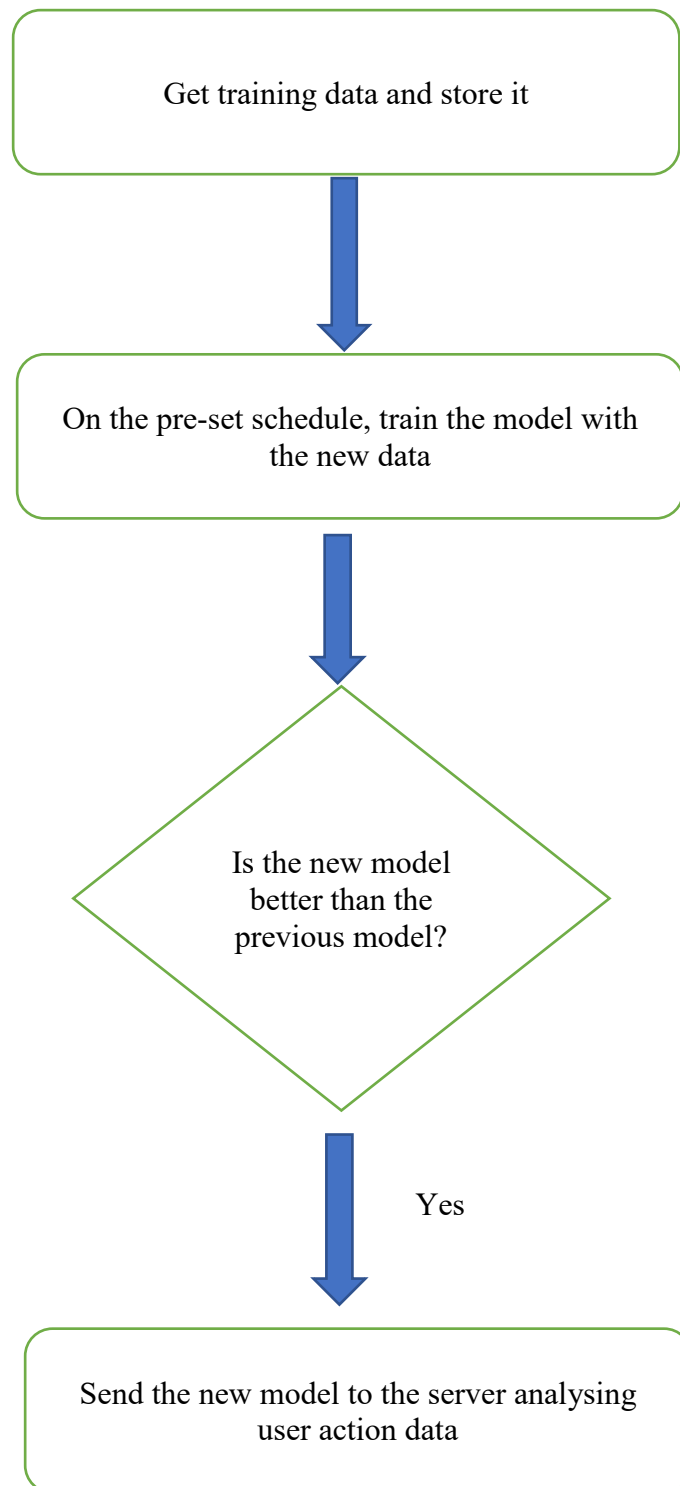
To calculate the customer sentiment score, another system runs a script periodically. It fetches the list of operations from the product database, fetches the user usage processed data from the customer database and calculates the customer sentiment score. Once the score is calculated, it writes back the score in the customer database.

A reporting application can retrieve this data periodically and build a complete report for a given customer.

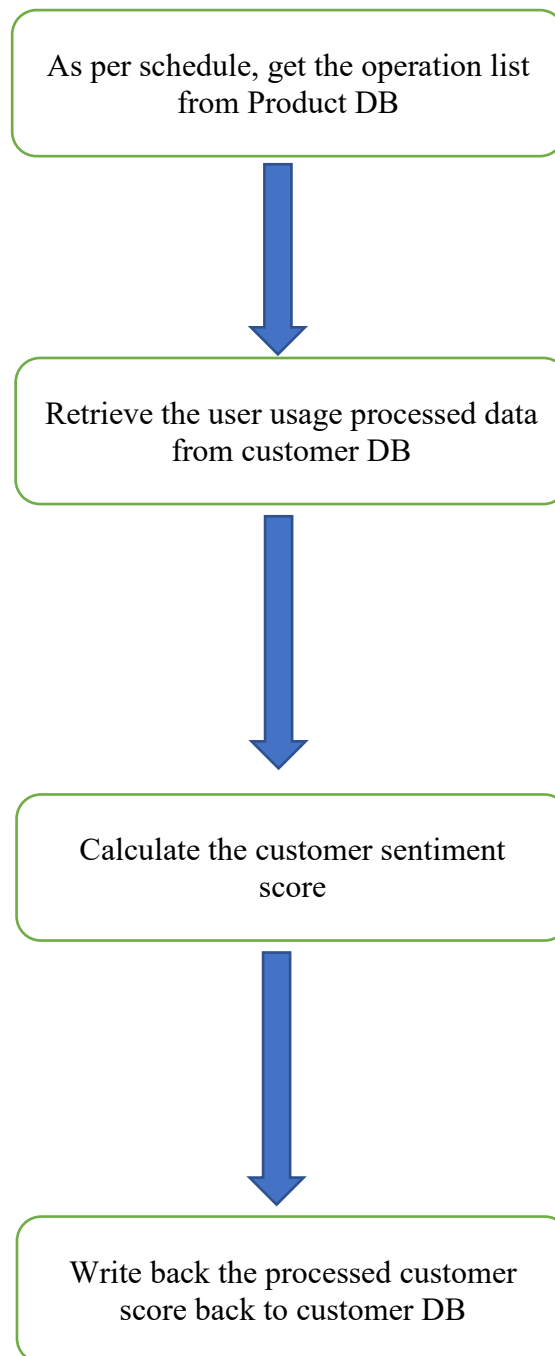
7.2 Flow diagram for the server which analyses the user actions



7.3 Flow diagram of training server



7.4 Flow diagram of report generator



7.5 Screenshot from the working code

To achieve the objective, we have done the following:

1. **UI:** Written in HTML, CSS and JS for simulating Adobe's UI menu system
2. **Agent:** Written in JS and embedded in UI
3. **User action analysis:** Written in python using tensorflow, numpy, pillow and hosted on flask (app.py)
4. **Model training:** Receiving of data written in python and hosted on flask. For model training, Tensorflow is used. (app.py, train.py)
5. **Customer score calculator:** Approach written in Python (score.py)
6. **Data storage:** All data is stored in MongoDB

The screenshot displays the 'XYZ Application' interface. At the top, there is a dark navigation bar with the menu items 'File', 'Edit', 'Filter', and 'Publish'. Below the navigation bar, the title 'XYZ Application' is centered. A message states: 'You can read the following document: http://file_open_openproject_path'. The main content area is divided into three columns: 'Operation Mode', 'User Data', and 'End operation simulate'. Under 'Operation Mode', there are three radio buttons: 'Using application' (selected), 'Training data', and 'Validation data'. Under 'User Data', there is a 'Username:' label with a text input field containing 'manish', and a 'User experience level:' label with three radio buttons: 'Beginner' (selected), 'Intermediate', and 'Professional'. Under 'End operation simulate', there are two radio buttons: 'Operation success' (selected) and 'Operation failed'. Below these, there is a 'Delay in ms:' label with a text input field containing '0.0'.

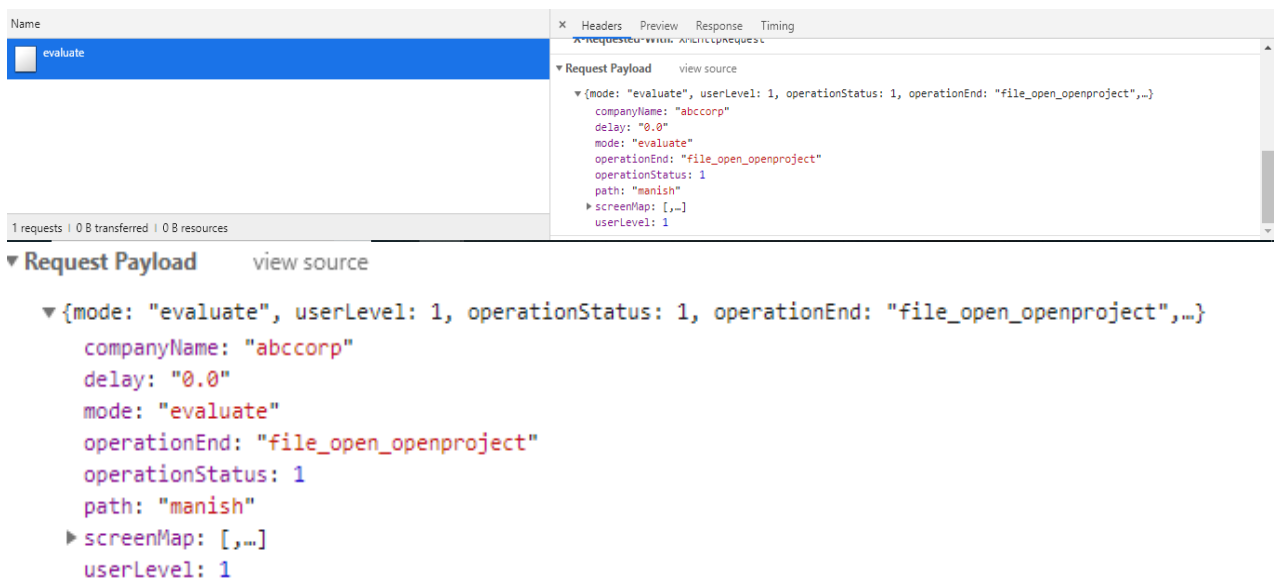
Figure 27: UI showing the help article

```

D:\MS\BITSPilani\Sem4\project\main\ui>python app.py
Using TensorFlow backend.
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
Using TensorFlow backend.
* Debugger is active!
* Debugger PIN: 216-155-486
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [08/Apr/2019 04:20:18] "[37mGET / HTTP/1.1[0m" 200 -
127.0.0.1 - - [08/Apr/2019 04:20:18] "[37mGET /static/css/bootstrap-responsive.css HTTP/1.1[0m" 200 -
127.0.0.1 - - [08/Apr/2019 04:20:18] "[37mGET /static/css/bootstrap.css HTTP/1.1[0m" 200 -
127.0.0.1 - - [08/Apr/2019 04:20:21] "[37mGET /static/js/jquery.js HTTP/1.1[0m" 200 -
127.0.0.1 - - [08/Apr/2019 04:20:21] "[37mGET /static/js/bootstrap-dropdown.js HTTP/1.1[0m" 200 -
127.0.0.1 - - [08/Apr/2019 04:20:21] "[37mGET /static/js/bootstrap-modal.js HTTP/1.1[0m" 200 -
manish

```

Figure 28: flask serving all end points



The screenshot shows a web browser's developer tools interface. On the left, a list of requests is shown, with the first one selected. The main pane on the right displays the 'Request Payload' for the selected request. The payload is a JSON object with the following structure:

```

{
  "mode": "evaluate",
  "userLevel": 1,
  "operationStatus": 1,
  "operationEnd": "file_open_openproject",
  "companyName": "abccorp",
  "delay": "0.0",
  "mode": "evaluate",
  "operationEnd": "file_open_openproject",
  "operationStatus": 1,
  "path": "manish",
  "screenMap": [
    ...
  ],
  "userLevel": 1
}

```

Figure 29: agent sending raw user usage data for analysis

```

Number of classes: 39
Layer (type)                Output Shape                Param #
-----
conv2d (Conv2D)              (None, 298, 298, 16)       160
max_pooling2d (MaxPooling2D) (None, 149, 149, 16)       0
conv2d_1 (Conv2D)             (None, 147, 147, 16)      2320
max_pooling2d_1 (MaxPooling2 (None, 73, 73, 16)        0
flatten (Flatten)             (None, 85264)              0
dense (Dense)                 (None, 512)                43655680
dense_1 (Dense)               (None, 39)                 20007
=====
Total params: 43,678,167
Trainable params: 43,678,167
Non-trainable params: 0

WARNING:tensorflow:From C:\Program Files\Python36\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
2019-04-08 04:29:48.609933: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
Epoch 1/15
3/31 [=>.....] - ETA: 56s - loss: 3.6677 - acc: 0.0000e+00

```

Figure 30: Model being trained

```

score path1:-1.0
score end1:1.0
score path-1.0score end1.0
combined score:1:0.0
{'n': 1, 'nModified': 0, 'ok': 1.0, 'updatedExisting': True}
score path0.0score end0.0
combined score:2:0.0
{'n': 1, 'nModified': 0, 'ok': 1.0, 'updatedExisting': True}
score path0.0score end0.0
combined score:3:0.0
{'n': 1, 'nModified': 0, 'ok': 1.0, 'updatedExisting': True}
file_open_openproject
score path1:-0.3333333333333333
score end1:1.0
score path-0.3333333333333333score end1.0
combined score:1:0.6666666666666667
{'n': 1, 'nModified': 1, 'ok': 1.0, 'updatedExisting': True}
score path2:1.0
score end2:1.0
score path1.0score end1.0
combined score:2:2.0
{'n': 1, 'nModified': 1, 'ok': 1.0, 'updatedExisting': True}
score path3:1.0
score end3:1.0
score path1.0score end1.0
combined score:3:2.0
{'n': 1, 'nModified': 1, 'ok': 1.0, 'updatedExisting': True}
file_open_openfile
score path1:1.0
score end1:1.0

```

Figure 31: Score being calculated

```
MongoDB Enterprise > db.overallscore.find({"type":"overallscore"})
{ "_id" : ObjectId("5c99c44f959c673891dfe623"), "type" : "overallscore", "experience_score" : 0.5833333333333335 }
MongoDB Enterprise >
```

Figure 32: Overall Score for a customer

Summary

Understanding the customer's sentiment is one of the highest priorities for Adobe. The existing methods used by the company helps it to learn about the customer sentiment only after they reach out to the company.

As part of this project, we have seen that capturing user software usage pattern and predicting a customer's sentiment is quite essential and helps us in reaching out to the customer even before the customer reaches out to us. This has many benefits which include having greater customer connect, increased revenues as a happy customer is more likely to renew or purchase new subscriptions and also reduces the effort of after sales team as well. We have also seen that by suggesting the user a help article, we now reach out to the customer in a matter of seconds which is a big win over the earlier ways where we expected a customer to search our help documents.

In this project, we have used the power of Machine learning and Artificial intelligence to analyse customer software usage behaviour. Our model uses some of the widely available libraries. The model we built has given us some good results. The way the system is designed, we can always improve the model by adding more training data and make it accurate further. We can also include more features to monitor which can also help us predict the customer sentiment score with more details.

Overall, it has been a great experience while working on this project where we solve one of the biggest challenges of the day i.e. to measure the customer mood in real-time and also helping the customer the moment, they face an issue.

Conclusions and Recommendations

In this project, we have designed a system which can capture user's application usage data over multiple aspects and can analyse them in real-time. Based on the real-time analysis, we can recommend the user some help articles if they are facing difficulties using the application. We can also calculate the customer's sentiment score and can take affirmative action's well in advance.

This approach can be applied to any product Adobe has. As we use Machine learning and Artificial intelligence, we can train the model to recognize patterns for each application individually thus making it easily portable to any of the products.

The amount of data we now gather about the user helps us in making more informed decisions and we can make our customers more successful in what they are doing.

Directions for future work

In this project, we have taken 39 operations and monitor the usage on 3 parameters. We can further add more features to monitor like the user usage data of the same application on different devices, how the application is rendered, what values the user inputs for a given operation etc. We can also cover more UI types like mobile interfaces.

For the features where we are using some pre-determined data like weights applied to calculate feature score or measuring the time the application takes to complete an operation against a predefined list, we can use machine learning to come up with a more accurate values to compare with.

Bibliography and References

BOOKS

1. Andreas C. Muller & Sarah Guidol, Introduction to Machine Learning with Python, Sebastopol, O'Reilly, 2016
2. Stuart J. Russell and Peter Norvig, Artificial Intelligence: A modern Approach, Essex, Pearson, 2014
3. Tom Hope, Itay Lieder and Yehezkel S. Resheff, Learning TensorFlow: A Guide to Building Deep Learning Systems, Sebastopol, O'Reilly, 2017

CONFERENCE PROCEEDINGS

4. Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, Michele Lanza, Rocco Olivet; Sentiment analysis for software engineering: how far can we go?; ICSE '18 Proceedings of the 40th International Conference on Software Engineering; Gothenburg, Sweden — May 27 - June 03, 2018; ACM New York, NY, USA
5. D.P. Kingma, J. Ba, Jimmy Lei Ba; Adam: A Method for Stochastic Optimization, ICLR, San Diego, 2015
6. Xavier Glorot, Antoine Bordes, Y. Bengio; Deep Sparse Rectifier Neural Networks; 14th International Conference on Artificial Intelligence and Statistics (AISTATS), April 11-13, 2011, Ft. Lauderdale, FL, USA
7. Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, Yann Lecun; What is the Best Multi-Stage Architecture for Object Recognition?; International Conference on Computer Vision (ICCV'09); Sep 29, 2009 - Oct 2, 2009; Kyoto, Japan

Appendices

C:

CSS: Cascading style sheet

CNN: Convolutional neural network

G:

GUI: Graphical user interface

H:

HTML: Hyertext Markup Language

J:

JS: JavaScript

Checklist of items for the Final Dissertation Report

This checklist is to be attached as the last page of the report.

This checklist is to be duly completed, verified and signed by the student.

1.	Is the final report neatly formatted with all the elements required for a technical Report?	Yes / No
2.	Is the Cover page in proper format as given in Annexure A?	Yes / No
3.	Is the Title page (Inner cover page) in proper format?	Yes / No
4.	(a) Is the Certificate from the Supervisor in proper format? (b) Has it been signed by the Supervisor?	Yes / No Yes / No
5.	Is the Abstract included in the report properly written within one page? Have the technical keywords been specified properly?	Yes / No Yes / No
6.	Is the title of your report appropriate? The title should be adequately descriptive, precise and must reflect scope of the actual work done. Uncommon abbreviations / Acronyms should not be used in the title	Yes / No
7.	Have you included the List of abbreviations / Acronyms?	Yes / No
8.	Does the Report contain a summary of the literature survey?	Yes / No
9.	Does the Table of Contents include page numbers? (i). Are the Pages numbered properly? (Ch. 1 should start on Page # 1) (ii). Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures) (iii). Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables) (iv). Are the Captions for the Figures and Tables proper? (v). Are the Appendices numbered properly? Are their titles appropriate	Yes / No Yes / No Yes / No Yes / No Yes / No Yes / No Yes / No
10.	Is the conclusion of the Report based on discussion of the work?	Yes / No
11.	Are References or Bibliography given at the end of the Report? Have the References been cited properly inside the text of the Report? Are all the references cited in the body of the report	Yes / No Yes / No Yes / No
12.	Is the report format and content according to the guidelines? The report should not be a mere printout of a Power Point Presentation, or a user manual. Source code of software need not be included in the report.	Yes / No

Declaration by Student:

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

Place: NOIDA

Date: 08/04/2019

Manish Singh
Signature of the Student

Name: MANISH SINGH

ID No.: 2017 HT 12447