



---

# USER ROUTINE PREDICTIVE ANALYSIS

---

BIG DATA ANALYTICS AND APPLICATIONS PROJECT



SPRING 2016

SUBMITTED BY

RAKESH REDDY BANDI (02)

NIHAR DUDAM (06)

SRICHARAN PAMURU (20)

MARK SCHULTZ (26)

## **I. Introduction:**

Big data has become a buzzword in the field of Computer Science. As technology advances, we are finding new ways to collect and analyze data. With the invention of the smart watch, we have been able to expand this collection of data to areas that had not been possible years ago. In addition, it has never been easier for the user to retrieve data from their device when needed. Our project combines these two conveniences into one application. We will use big data to measure and map a User's routine, and use this routine to make predictions and suggestions about the future.

We plan on applying several sources for data collection. First of all, the applications collect data from movements of the watch to classify patterns in certain movements. It will use these movements to guess the activity that the User may be doing at the moment. It will also collect data from the heart rate monitor of the smartwatch to measure the rigor of the User's activity. GPS data on the watch will determine the location of the user. We will also use datetime metrics to determine the day and the time that the user typically does certain activities.

The sources of data above will be collected by the user, but we also want to combine this with several API's to make better predictions, and add suggestions to give back to the user. We will use speech to text API to query the application for information. We will use the Google Location API to develop a richer context of the user's most frequented locations, and suggest possible locations that might interest the user. We will implement the Yelp API, so that when the application predicts when the user eats, it will make suggestions on places to go.

All of the data we will collect will be organized using the Hadoop file system and implemented using Amazon Web Services. We will discuss the technologies more in this documentation. The project is broken down into four phases. For the first Increment, we began the data collection to measure several common movements. We are initially using Naïve Bayes to classify these movements. We have also began research on several of the API's that we plan on implementing.

## **II. Project Goal and Objectives:**

The main implementation of the project will consist of the following specification based objective and subjective analysis along with a sketched perspective of goal:

### **1. Overall Goal:**

The goal of the project is to predict a user's activity at any time during a user's schedule. A large dataset is provided by the smartwatch and the machine is trained on the basis of machine learning algorithms as to how the coordinates will affect the motion of the sensor. Machine is trained and the provided data is analyzed based on the reading from the various sensors of the smartwatch. The location information will provide a set of variable features such as restaurants, parks, etc. near to the naive user. APIs will provide the analysis of information and data regarding the review and further circumspect options are brought forward to help the smartwatch analyze the data and provide a predictive routine. A variable set of sensors will collect the proximity, heart rate, accelerometer based motion readings, etc. to analyze data set. The analysis of this large datasets will return recommended notifications to the user for a viable routine.

### **2. Specific objectives:**

- Predict user routine by analysis of large datasets through machine learning techniques.
- Return geolocation based places in proximity such as restaurants, parks, etc.
- Sensor dataset provides specific activity based readings.
- Provide recommended services by sending notification to user depending on time and routine.

### **3. Specific features:**

Sensors in the watch serve as primary source for providing data which is required for analysis. The accelerometer will provide a three dimensional reading with regard to the position. The heart rate sensor will provide the heart beat rate which can be used to obtain the magnitude of force exerted while performing a particular activity. GPS will provide information about the location and the APIS will return data to the user depending upon the routine's requirements and suggestive

notifications will be sent to the smartwatch while the connection to another remote device or machine such as a smartphone holds true. It will create a complicate alliance and thus the machine is to be trained using various machine learning algorithms. Data clusters are classified using the classification techniques intended to provide more information about the routine prediction. There are other sensors in the watch such as ambient light sensor, gyroscope, magnetic field sensor, step counter, gravity, linear acceleration, significant motion, etc. which are used collaboratively to obtain a dataset.

#### **4. Significance:**

The most important aspect of the project is that an analysis of big data is done in order to predict the routine of a specific user. It will generate a pattern based on the different readings to generate a more sophisticated clustered activity set. This will help in obtaining a clear cut view of a user's routine plus a more generative dimension to a time schedule. Sensors present in the smartwatch will be crucial in collection of data, which in turn is fed into the machine through various machine learning techniques. A stable schedule would prompt the user to receive a collaborative approach to obtain the output as a predictive routine analysis.

### III. Project Plan:

#### 1. Schedule for entire project:

The entire schedule for the project is shown as follows where the respective use case and process step up with all the modules and the time frame is categorized specifically to illustrate the step by step fulfilment of the entire processes done such as feature design and feature implementation.

**Screenshot 1:** The following screenshot demonstrates the overall perspective of implementation in which the project is being done. The To do and In Progress have a significantly high number of modules and are continued in the following images.

The screenshot displays the GitHub interface for the repository 'rakeshreddybandi / Robo-smart-Secure-System'. The 'Boards' tab is selected, showing a Kanban-style view of project issues. The interface includes a navigation bar with links to Code, Issues (25), Pull requests (0), Boards, Burndown, Wiki, Pulse, Graphs, and Settings. The repository has 4 stars and 0 forks. The 'Boards' section is divided into five columns: 'New Issues' (2), 'To Do' (17), 'In Progress' (8), 'Done' (9), and 'Closed' (3). Each column contains a list of issues with their titles, numbers, and status labels (e.g., 'enhancement', 'bug', 'help wanted').

Column	Count	Issue #	Title	Labels
New Issues (2)		#23	Robo-smart-Secure-System Cognitive analysis	
		#26	Robo-smart-Secure-System Connection between SmartWatch and Mobile Phone	bug, help wanted
To Do (17)		#29	Robo-smart-Secure-System Sensor data collection	enhancement, help wanted
		#30	Robo-smart-Secure-System Training the machine	enhancement, help wanted
		#33	Robo-smart-Secure-System Collecting json from Yelp API	enhancement, help wanted
		#34	Robo-smart-Secure-System Collecting json from Eventful API	
In Progress (8)			Application of machine learning algorithm to data set	enhancement, help wanted
		#19	Robo-smart-Secure-System Work on Event full API	
		#11	Robo-smart-Secure-System Sensor data collection	enhancement
Done (9)			Robo-smart-Secure-System Documenting project proposal	enhancement, help wanted
		#16	Robo-smart-Secure-System Requirement analysis	enhancement, help wanted
		#8	Robo-smart-Secure-System Establishment of connection between remote device and smartwatch	enhancement, help wanted
Closed (3)		#27	Robo-smart-Secure-System converting data collected into csv format	bug, help wanted
		#25	Robo-smart-Secure-System Test Smartwatch functionality	bug, duplicate, question
		#1	Robo-smart-Secure-System Getting more done in GitHub with ZenHub	

## Screenshot 2:

The following screenshots shows the second listing of the To Do phases and In Progress where there is an increase in the number of elements and tasks to be done. There is an incentive in populating the respective tasks and assign them to the members as per core requirement and generative design.

The screenshot displays a GitHub repository page for 'Robo-smart-Secure-System' by user 'rakeshreddybandi'. The URL is <https://github.com/rakeshreddybandi/Robo-smart-Secure-System/issues#boards?repos=50711311>. The repository has 4 Unwatch, 0 Stars, and 0 Forks. The 'Boards' tab is selected, showing a Kanban board with five columns: New Issues (2), To Do (17), In Progress (8), Done (9), and Closed (3). Each column contains a list of issues with their titles, IDs, and labels.

Column	Issue ID	Issue Title	Labels
New Issues (2)	#23	Robo-smart-Secure-System Cognitive analysis	
	#26	Robo-smart-Secure-System Connection between SmartWatch and Mobile Phone	bug, help wanted
To Do (17)	#31	Robo-smart-Secure-System Predicting the user activities	enhancement, help wanted
	#32	Robo-smart-Secure-System Adding SmartWatch data as training data	enhancement, help wanted
	#24	Robo-smart-Secure-System Notification to Smart Watch	enhancement, help wanted
	#37	Robo-smart-Secure-System	
In Progress (8)	#17	Robo-smart-Secure-System Work on Geo Location API	enhancement
	#12	Robo-smart-Secure-System Application of machine learning algorithm to data set	enhancement, help wanted
	#19	Robo-smart-Secure-System Work on Event full API	
	#11	Robo-smart-Secure-System Sensor data collection	
Done (9)	#10	Robo-smart-Secure-System Research on APIs	enhancement, help wanted, question
	#6	Robo-smart-Secure-System Refine and rebuild project proposal	enhancement
	#22	Robo-smart-Secure-System Modular Description	enhancement, help wanted
	#5	Robo-smart-Secure-System Documentation project proposal	
Closed (3)	#27	Robo-smart-Secure-System converting data collected into csv format	bug, help wanted
	#25	Robo-smart-Secure-System Test Smartwatch functionality	bug, duplicate, question
	#1	Robo-smart-Secure-System Getting more done in GitHub with ZenHub	

### Screenshot 3:

The following screenshot shows the phase where the tasks in progress have been concentrated mainly on the design specifications. It combines along with the done tasks to stabilize the development of project in due course to simultaneously bring out the working modules.

The screenshot displays the GitHub interface for the repository 'rakeshreddybandi / Robo-smart-Secure-System'. The URL in the browser is <https://github.com/rakeshreddybandi/Robo-smart-Secure-System/issues#boards?repos=50711311>. The repository has 4 Unwatch, 0 Stars, and 0 Forks. The main navigation bar includes links for Code, Issues (25), Pull requests (0), Boards, Burndown, Wiki, Pulse, Graphs, and Settings. The 'Boards' tab is active, showing a Kanban board with five columns: New Issues, To Do (17), In Progress (8), Done (9), and Closed (3). Each column contains a list of issues with their titles, numbers, and labels. The 'To Do' column has issues #13, #15, #18, and #31. The 'In Progress' column has issues #4, #20, #21, and #17. The 'Done' column has issues #2, #7, #3, and #10. The 'Closed' column has issues #27, #25, and #1. The 'New Issues' column is empty. The interface also includes a search bar, a 'New issue' button, and various filters like Labels, Milestones, and Assignees.

Column	Issue Number	Issue Title	Labels
New Issues	#23	Robo-smart-Secure-System Cognitive analysis	
	#26	Robo-smart-Secure-System Connection between SmartWatch and Mobile Phone	bug, help wanted
To Do (17)	#13	Robo-smart-Secure-System Testing	enhancement, help wanted
	#15	Robo-smart-Secure-System Maintenance	bug, enhancement, help wanted
	#18	Robo-smart-Secure-System Debugging	bug, help wanted, question
	#31	Robo-smart-Secure-System	
In Progress (8)	#4	Robo-smart-Secure-System Analyzing the machine learning algorithms	help wanted
	#20	Robo-smart-Secure-System Architecture design	enhancement, help wanted
	#21	Robo-smart-Secure-System Working on Weather API	enhancement, help wanted
	#17	Robo-smart-Secure-System Work on Geo Location API	
Done (9)	#2	Robo-smart-Secure-System Project Proposal Discussion	enhancement
	#7	Robo-smart-Secure-System Knowledge discovery of smarwatch	enhancement, question
	#3	Robo-smart-Secure-System Finalizing project plan	enhancement
	#10	Robo-smart-Secure-System Research on APIs	
Closed (3)	#27	Robo-smart-Secure-System converting data collected into csv format	bug, help wanted
	#25	Robo-smart-Secure-System Test Smartwatch functionality	bug, duplicate, question
	#1	Robo-smart-Secure-System Getting more done in GitHub with ZenHub	

## Screenshot 4:

The screenshot here shows the listing done where the penultimate requirement analysis and engineering is done to bring out the resulted output that is tested. The maintenance is to be done and especially debugging to negate results which are not in favour of project deployment.

The screenshot displays the GitHub interface for the repository 'rakeshreddybandi / Robo-smart-Secure-System'. The URL in the browser is <https://github.com/rakeshreddybandi/Robo-smart-Secure-System/issues#boards?repos=50711311>. The repository has 4 Unwatch, 0 Star, and 0 Fork. The navigation bar includes links for Code, Issues (25), Pull requests (0), Boards, Burndown, Wiki, Pulse, Graphs, and Settings. The main content area shows a Kanban board with five columns: New Issues (2), To Do (17), In Progress (8), Done (9), and Closed (3). Each column contains a list of issues with their titles, numbers, and labels like 'enhancement', 'bug', 'help wanted', 'duplicate', and 'question'.

Column	Count	Issue #	Title	Labels
New Issues (2)		#23	Robo-smart-Secure-System Cognitive analysis	
		#26	Robo-smart-Secure-System Connection between SmartWatch and Mobile Phone	bug, help wanted
To Do (17)	3		enhancement help wanted	
		#36	Robo-smart-Secure-System Refining all the features	
	8		enhancement help wanted	
		#38	Robo-smart-Secure-System Final Documentation	
In Progress (8)			Application of machine learning algorithm to data set	
	4		enhancement help wanted	
		#19	Robo-smart-Secure-System Work on Event full API	
Done (9)	2		enhancement help wanted	
		#5	Robo-smart-Secure-System Documenting project proposal	
	3		help wanted	
Closed (3)		#27	Robo-smart-Secure-System converting data collected into csv format	bug, help wanted
		#25	Robo-smart-Secure-System Test Smartwatch functionality	
	4		bug duplicate question	



## Screenshot 5:

The entire project schedule has been roughly shown here. A detailed insight of project plan is done in such a manner that there might be few more additional development or phase build in due course of the project. It shows a correlation of the various modules included as per specification and features.

The screenshot displays the GitHub interface for the repository 'rakeshreddybandi / Robo-smart-Secure-System'. The top navigation bar includes the repository name, a search bar, and links for Pull requests, Issues, Gist, and To Do. The repository statistics show 4 Unwatch, 0 Star, and 0 Fork. The main content area features a Kanban board with five columns: New Issues (2), To Do (17), In Progress (8), Done (9), and Closed (3). Each column contains a list of issues with their titles, numbers, and labels. The issues are categorized by labels such as 'enhancement', 'bug', 'help wanted', 'duplicate', and 'question'. The board provides a visual overview of the project's progress and upcoming tasks.

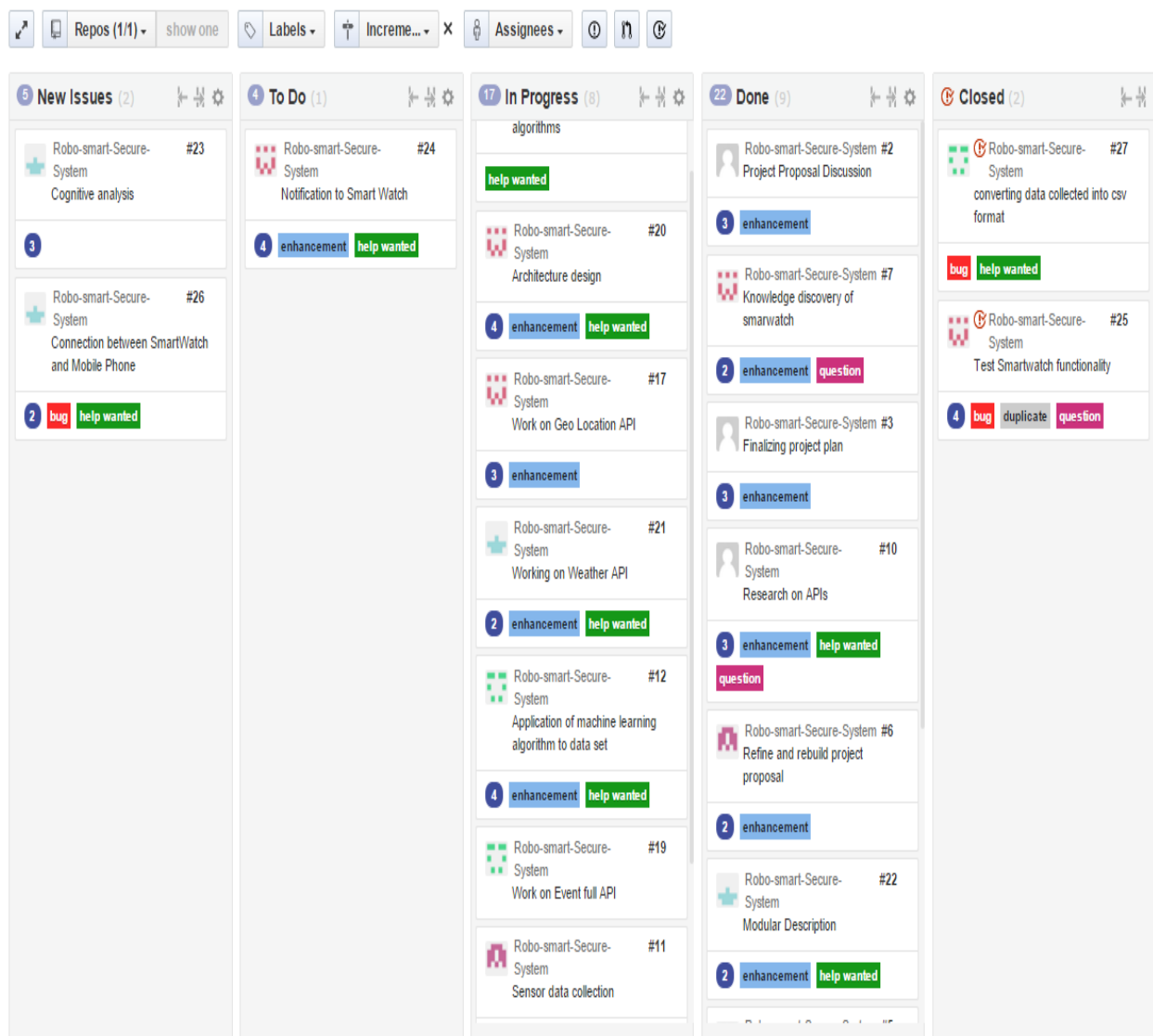
Column	Issue Number	Issue Title	Labels
New Issues (2)	#23	Robo-smart-Secure-System Cognitive analysis	
	#26	Robo-smart-Secure-System Connection between SmartWatch and Mobile Phone	bug, help wanted
To Do (17)	#34	Robo-smart-Secure-System Collecting json from Eventful API	enhancement
	#35	Robo-smart-Secure-System Analyze Location Data	enhancement, help wanted
	#36	Robo-smart-Secure-System Refining all the features	enhancement, help wanted
	#38	Robo-smart-Secure-System	
In Progress (8)		Application of machine learning algorithm to data set	enhancement, help wanted
	#19	Robo-smart-Secure-System Work on Event full API	
	#11	Robo-smart-Secure-System Sensor data collection	
	#14	Robo-smart-Secure-System Working on Yelp API	enhancement, help wanted
Done (9)	2	enhancement	help wanted
	#5	Robo-smart-Secure-System Documenting project proposal	
	3	help wanted	
	#16	Robo-smart-Secure-System Requirement analysis	enhancement, help wanted
Closed (3)	#27	Robo-smart-Secure-System converting data collected into csv format	bug, help wanted
	#25	Robo-smart-Secure-System Test Smartwatch functionality	bug, duplicate, question
	#1	Robo-smart-Secure-System Getting more done in GitHub with ZenHub	

## 2. Project Timeline:

The implementation of the project is covered with a specific timeline and this is shown as follows to implement it using Zenhub:

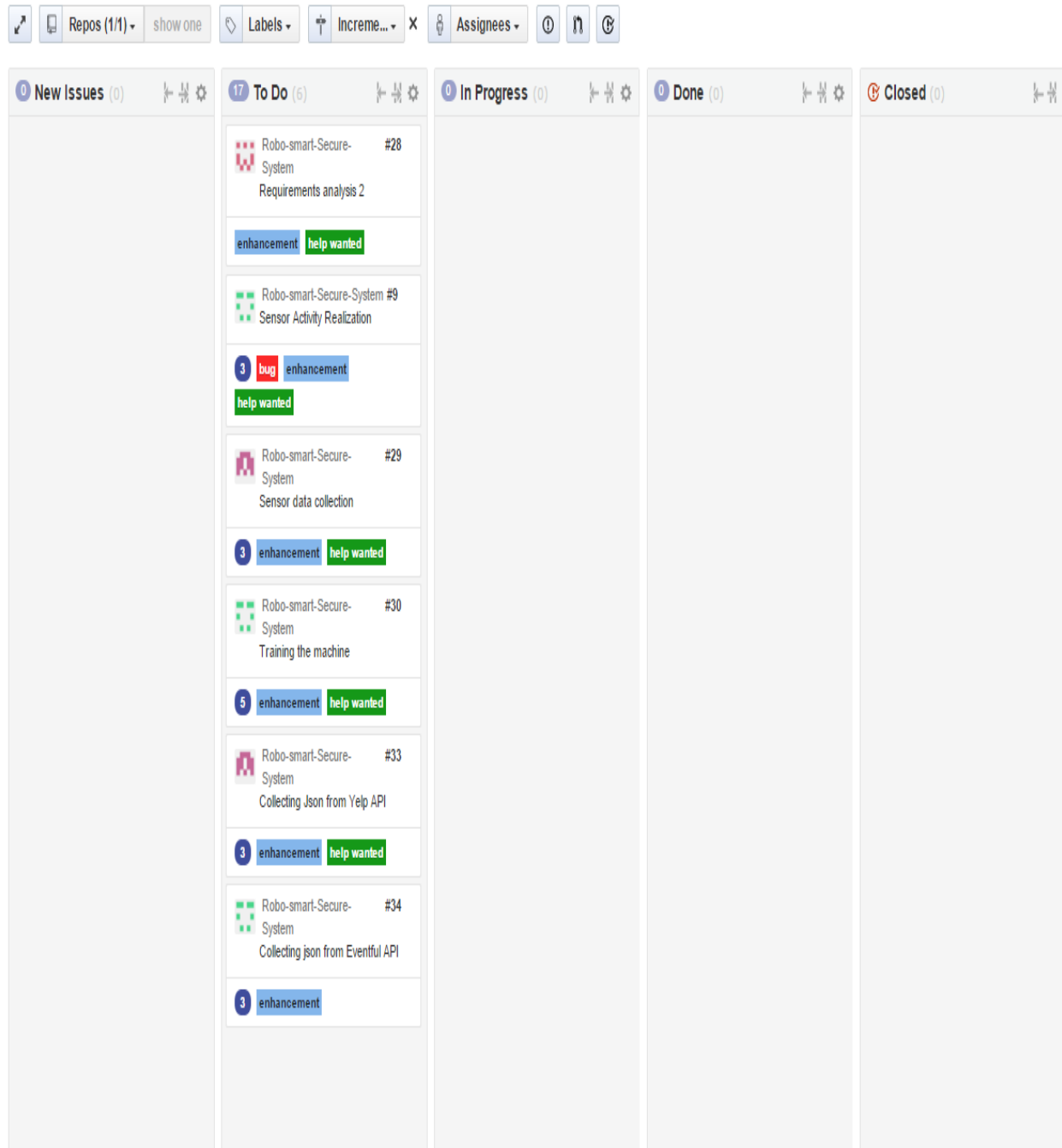
### Increment 1:

The first increment has the following tasks to be completed by 02/19/2016 with tasks assigned as such:



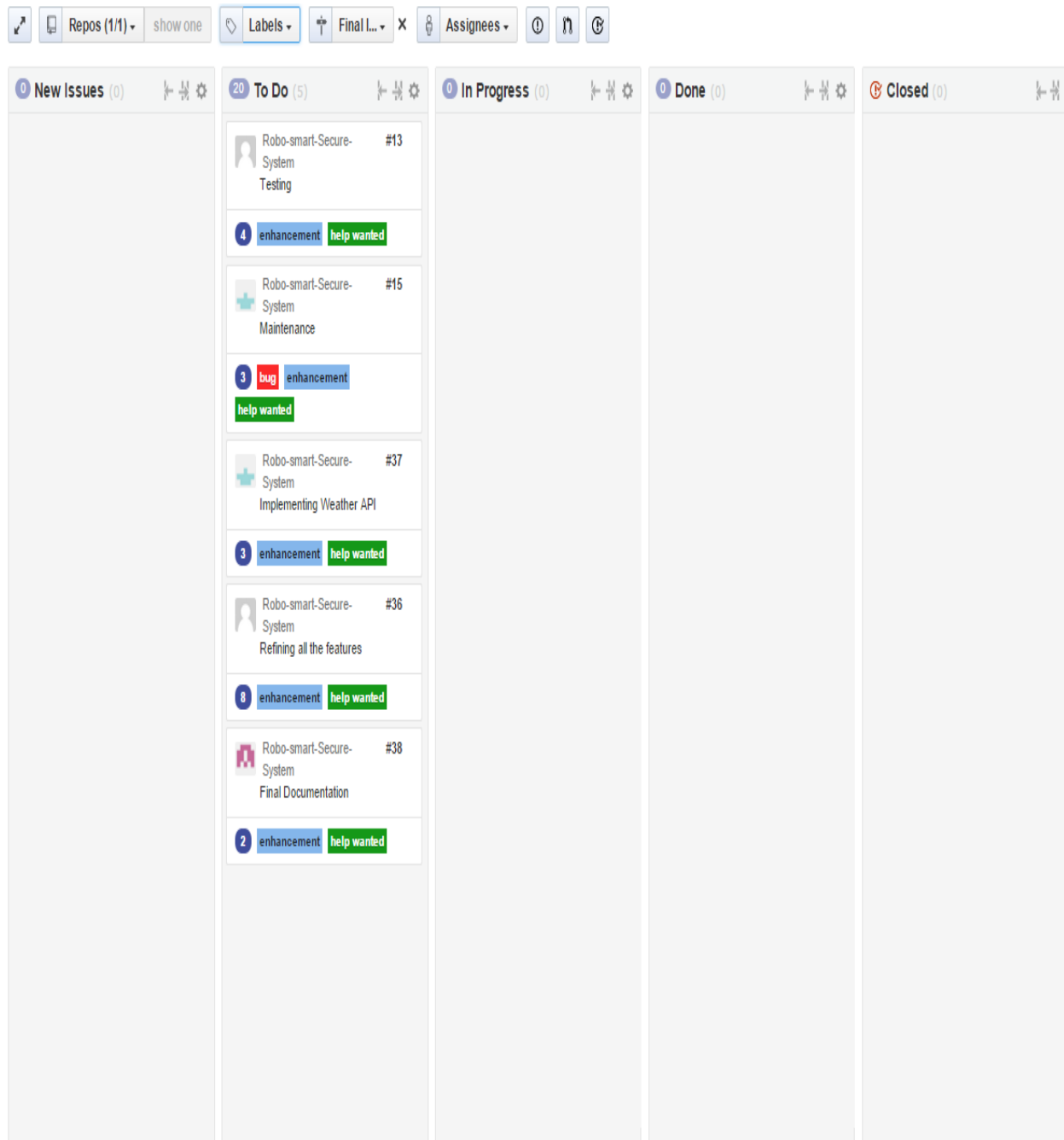
## Increment 2:

The timeline for the second increment due on 03/11/2016 will have the graph as follows:



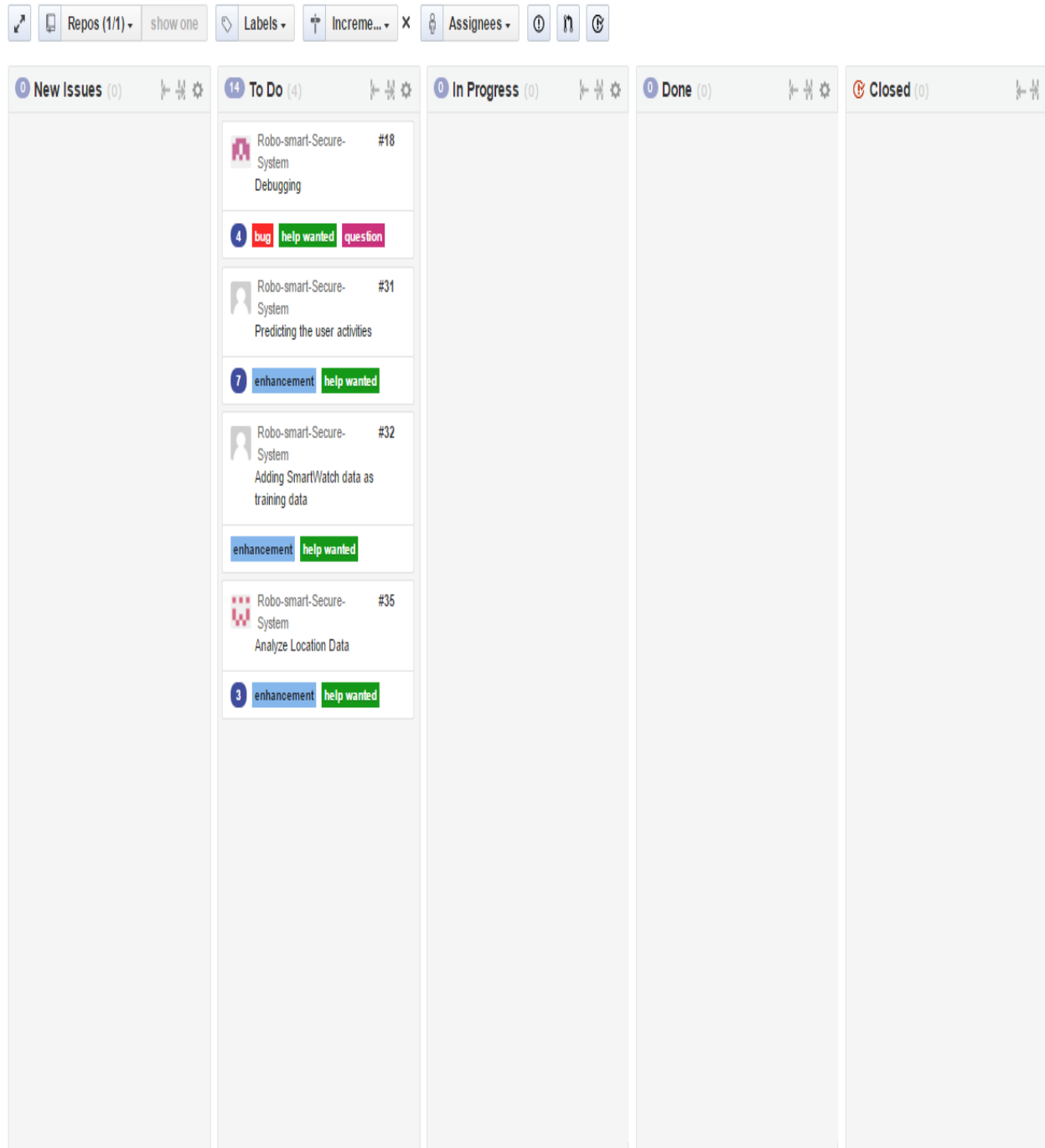
### Increment 3:

The timeline for the third increment due on 04/06/2016 will have the graph as follows:



## Final Increment:

The timeline for the final increment due on 04/29/2016 will have the graph as follows:

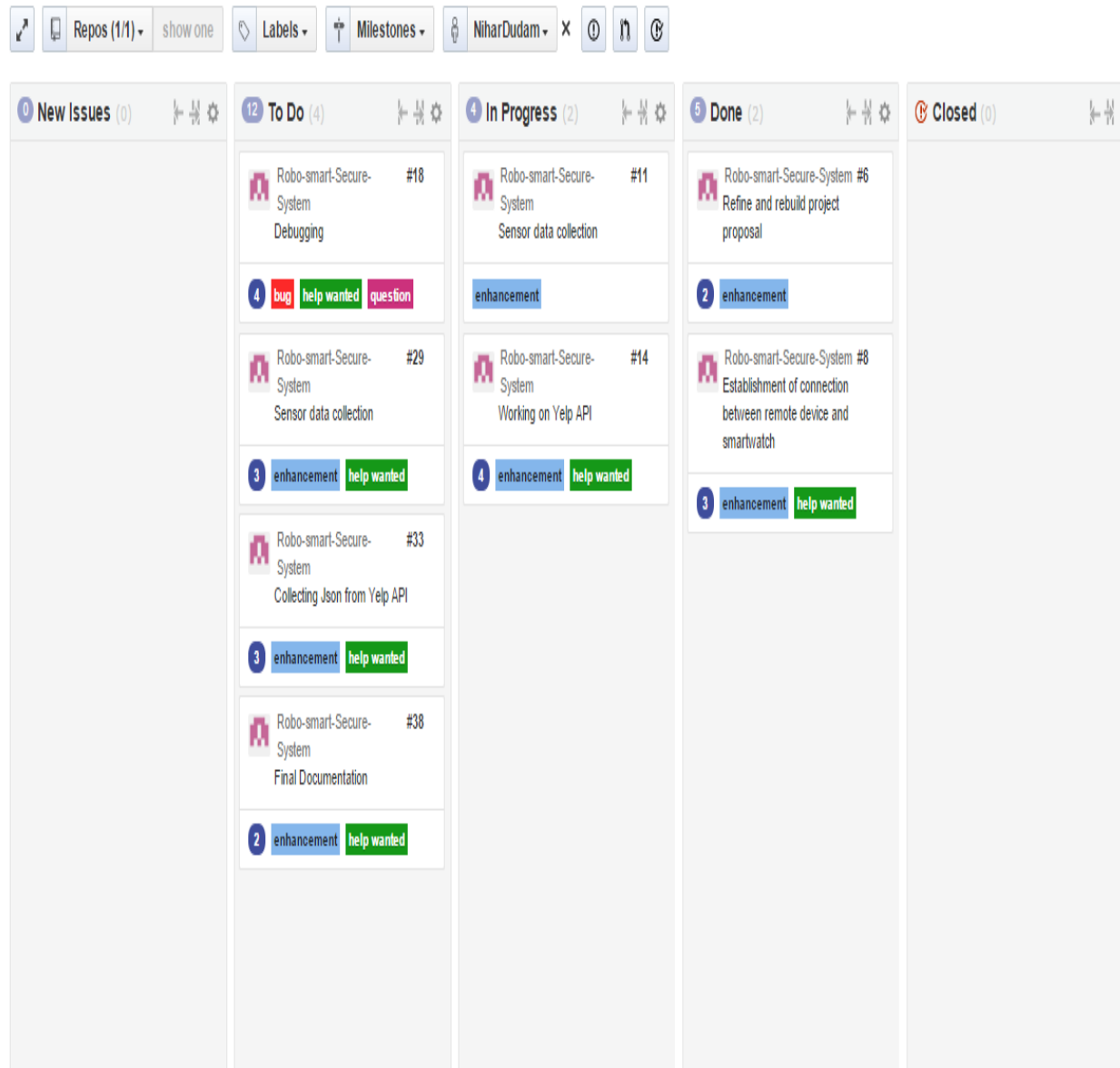


### 3. Project Members and Task Responsibility:

This section deals with the timeline of the project for each member along with the task responsibility:

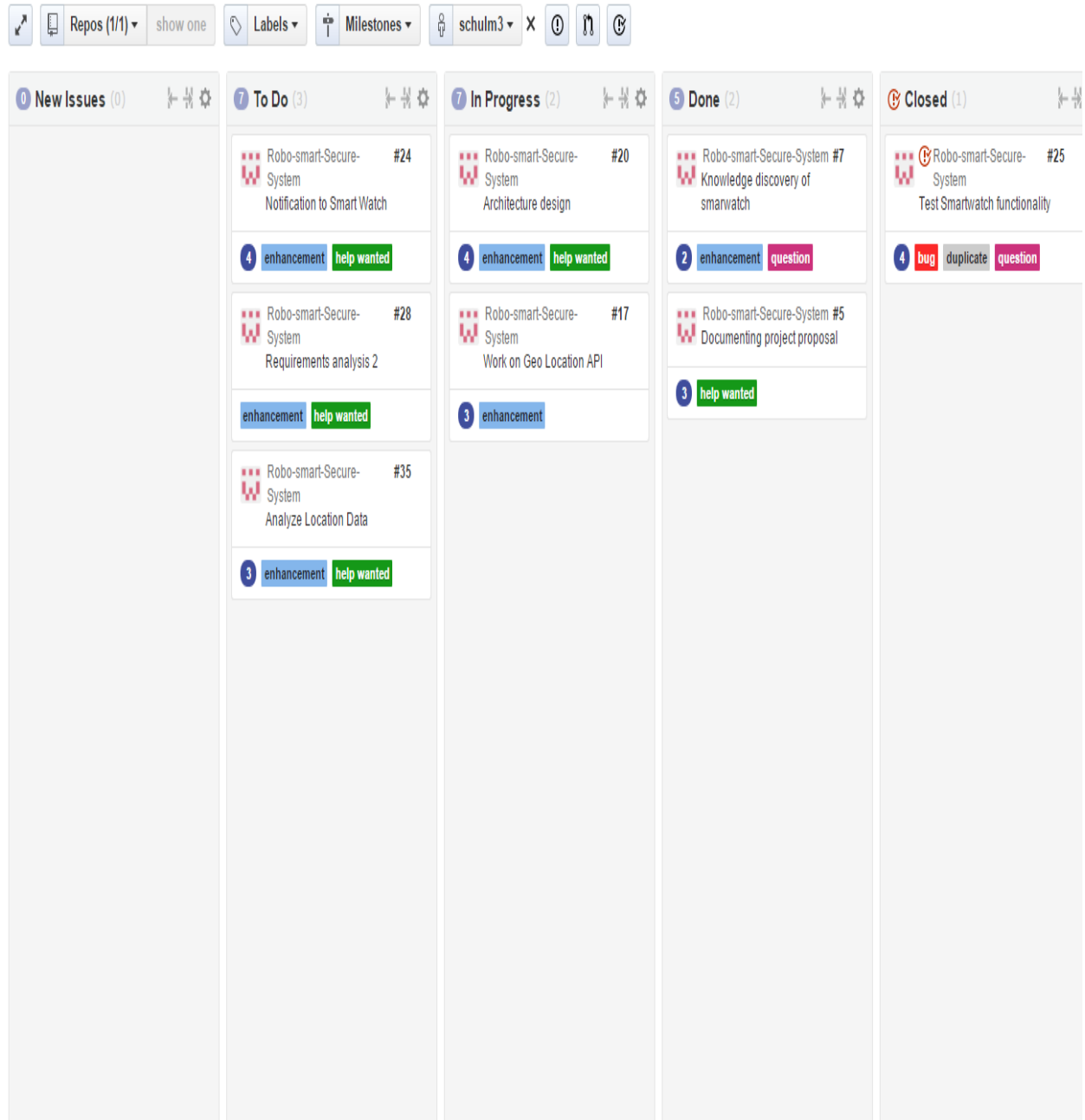
#### a. Nihar Dudam

The tasks assigned to Nihar in order to facilitate the completion of the project has the following diagram:



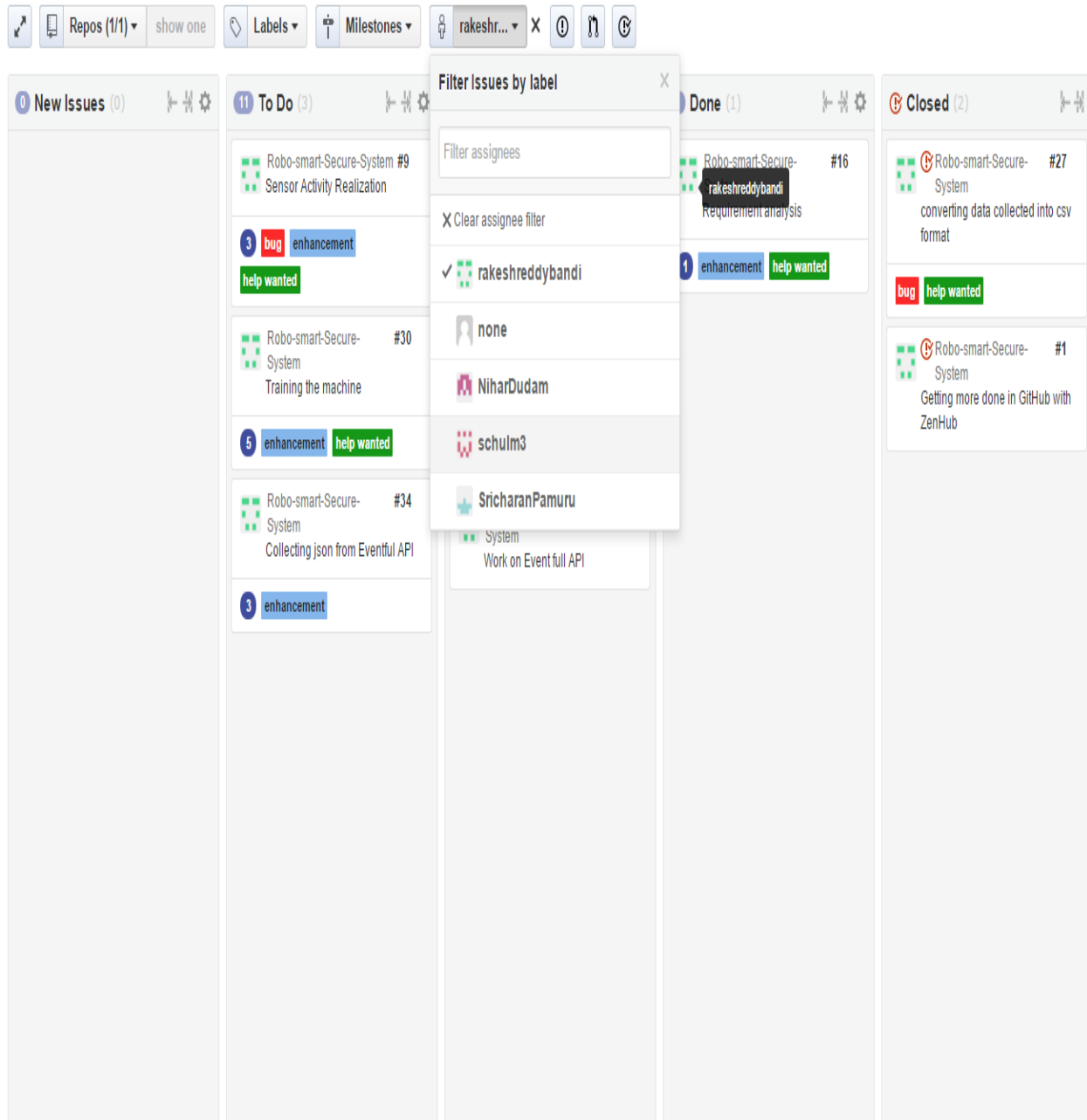
## b. Mark Schultz

The tasks assigned to Mark in order to facilitate the completion of the project has the following diagram:



### c. Rakesh Reddy Bandi

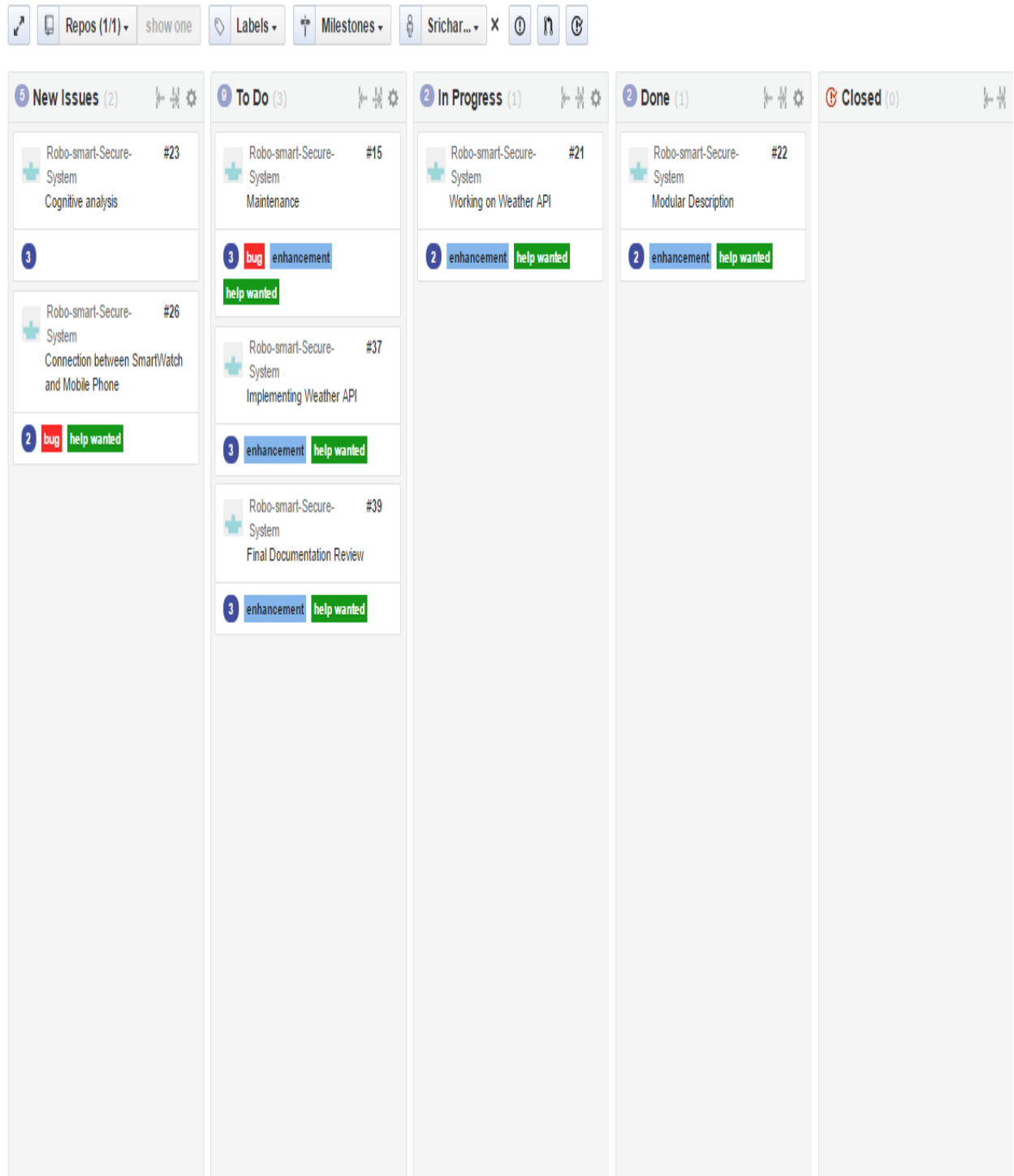
The tasks assigned to Rakesh in order to facilitate the completion of the project has the following diagram:





#### d. Sricharan Pamuru

The tasks assigned to Sricharan in order to facilitate the completion of the project has the following diagram:



#### 4. Burndown charts:

The burndown charts would basically represent the time deployment of project as per the various phases. It would instantiate the modules of time and its according development over the schedule.

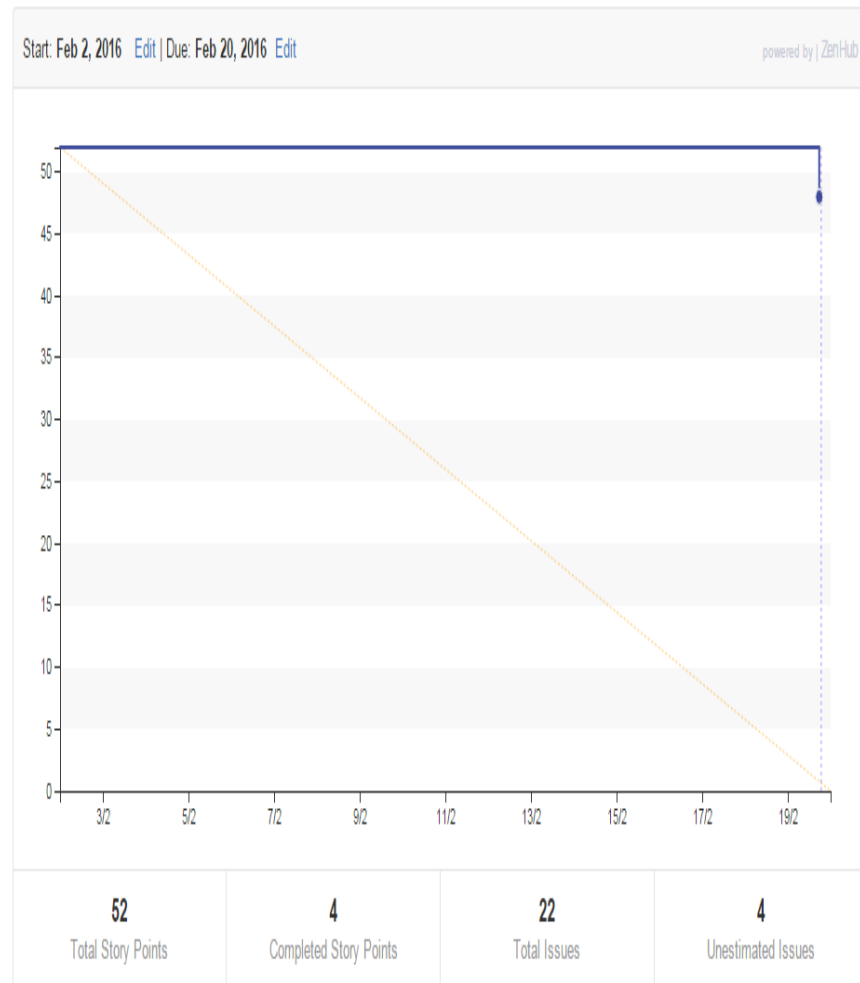
##### Increment 1:

The burndown chart of increment 1:

### Increment 1

[Edit Milestone](#) [Change Milestone](#)

In this increment group need to discuss about project plan and features. How can we implement all the features discussed ?. We should also finalize about the first increment tasks to be done.



## Increment 2:

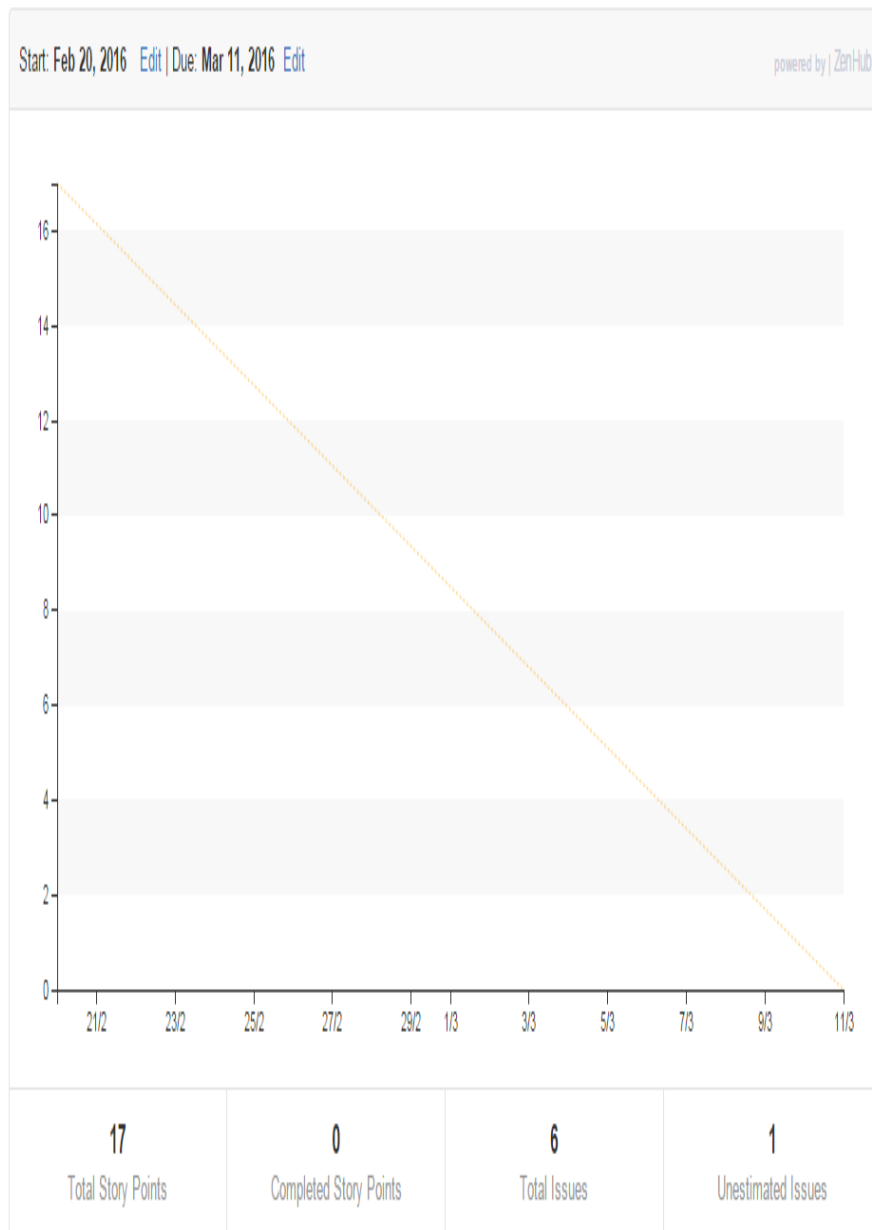
The burndown chart of increment 2 is as follows:

### Increment 2

 Edit Milestone

 Change Milestone ▾

Additional features must be implemented in this increment.



## Increment 3:

The burndown chart of increment 3 is:

### Increment 3

Discussed features must be implemented in this increment.

Edit Milestone

Change Milestone ▼

Filter milestone

powered by | Zen-Hub X

Filter milestone

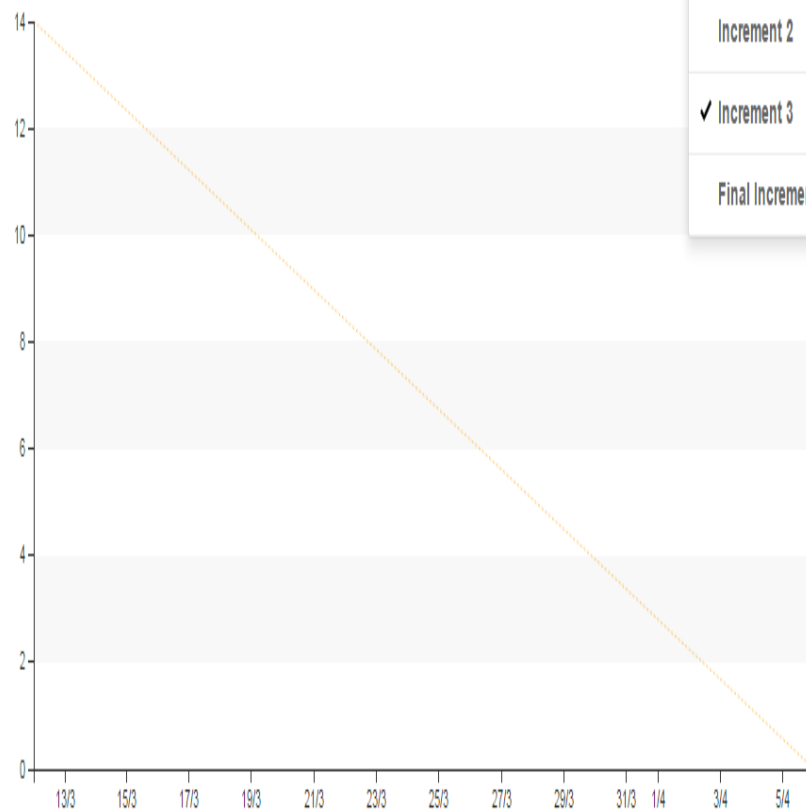
Increment 1

Increment 2

✓ Increment 3

Final Increment

Start: Mar 12, 2016 Edit | Due: Apr 6, 2016 Edit



14

Total Story Points

0

Completed Story Points

4

Total Issues

1

Unestimated Issues

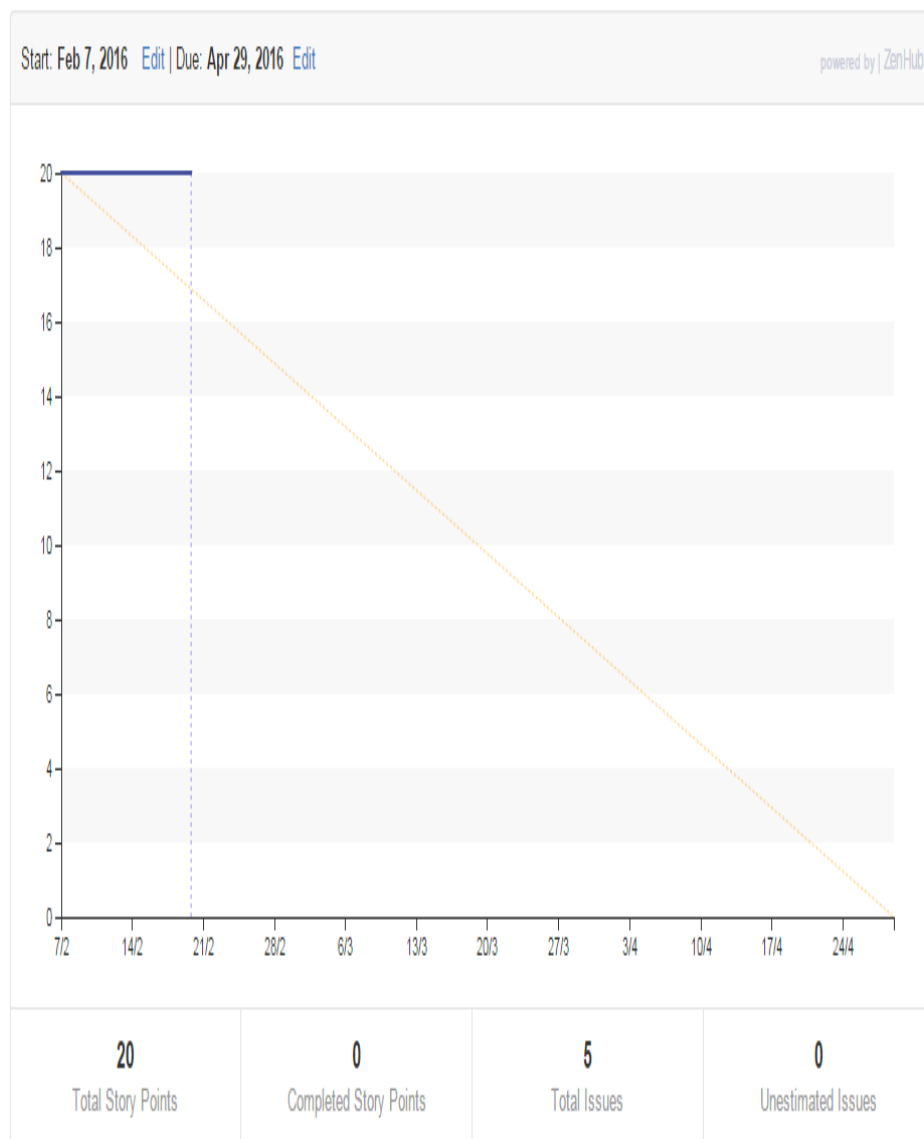
## Final Increment:

The final increment has the burndown chart as follows:

### Final Increment

 Edit Milestone  Change Milestone ▼

Testing the various features implemented in the project and deploying.



#### **IV. Second Increment Report**

In this increment, we have focused on extending our Naïve Bayes algorithm from increment one and implementing it in Spark. Although R is a great tool for testing this algorithm, we need the capability of streaming large amounts of data dynamically moving forward. Spark streaming handles this quite well. We have also extended our project to include sentence data. Eventually, we want to be able to recommend restaurants to the user based on their location and Restaurants nearby that users are praising. In this increment, we have created a program that can read text and make recommendations sent back to the phone.

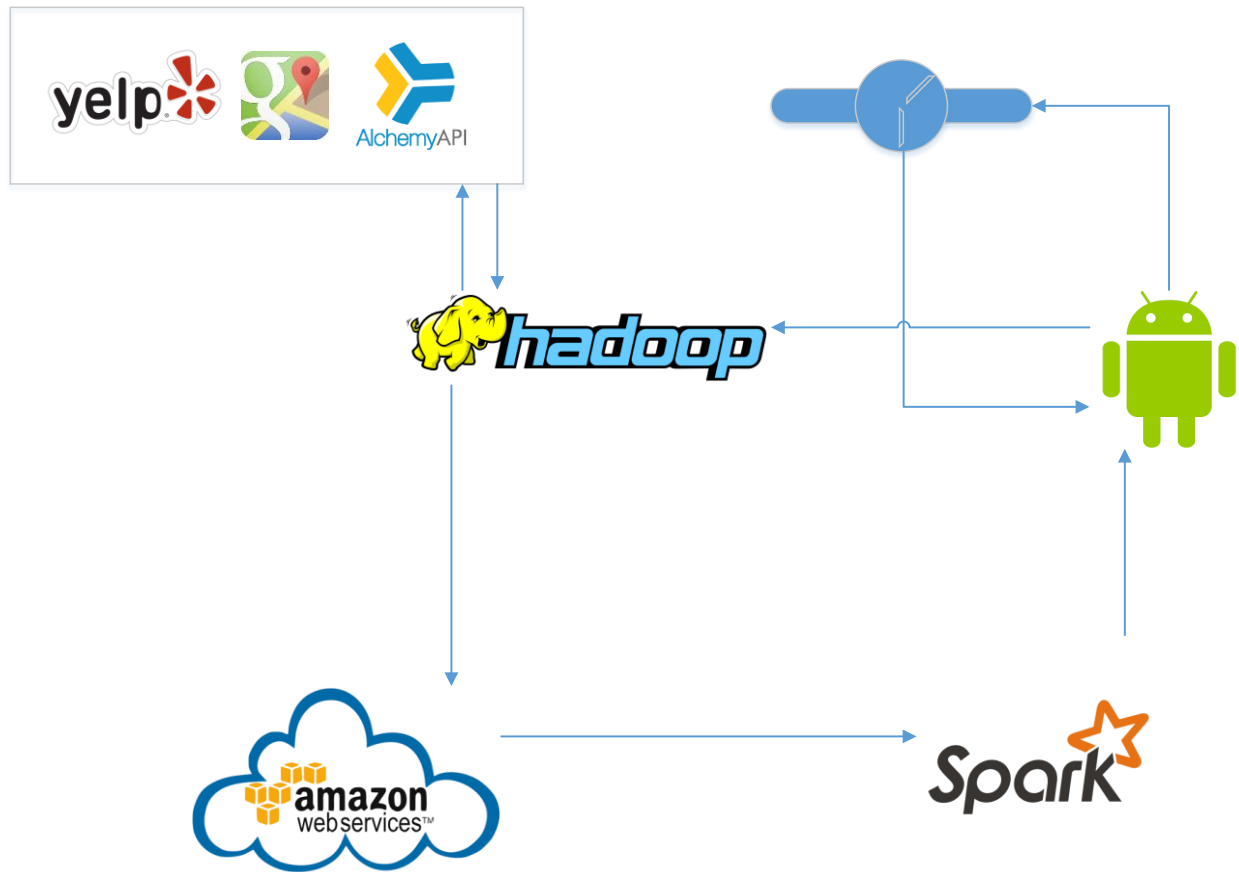
Because our goal is to recommend restaurants to the User based on sentences sent in the application, we need to do two things. First, we have tested a Semantic analyzer that will tell us if the sentence is positive or negative. After collecting this information about the sentence, we use a Collaborative Filter algorithm to rank restaurants based on the user's preferences.

Finally, we must send these recommendations to the user. We have extended the notification sending from the previous lab and extended it to the restaurant rankings.

Moving forward, we must accomplish a few goals. First of all, we must improve the quality of the user interface. The user interface is key to making our application a product that people will want to use. Next, we need to integrate several of the features that we have been developing independent of one another. This project will only be successful if all of these features work together efficiently. We also must extend our Machine learning algorithm to include the location of the user and the time of day.

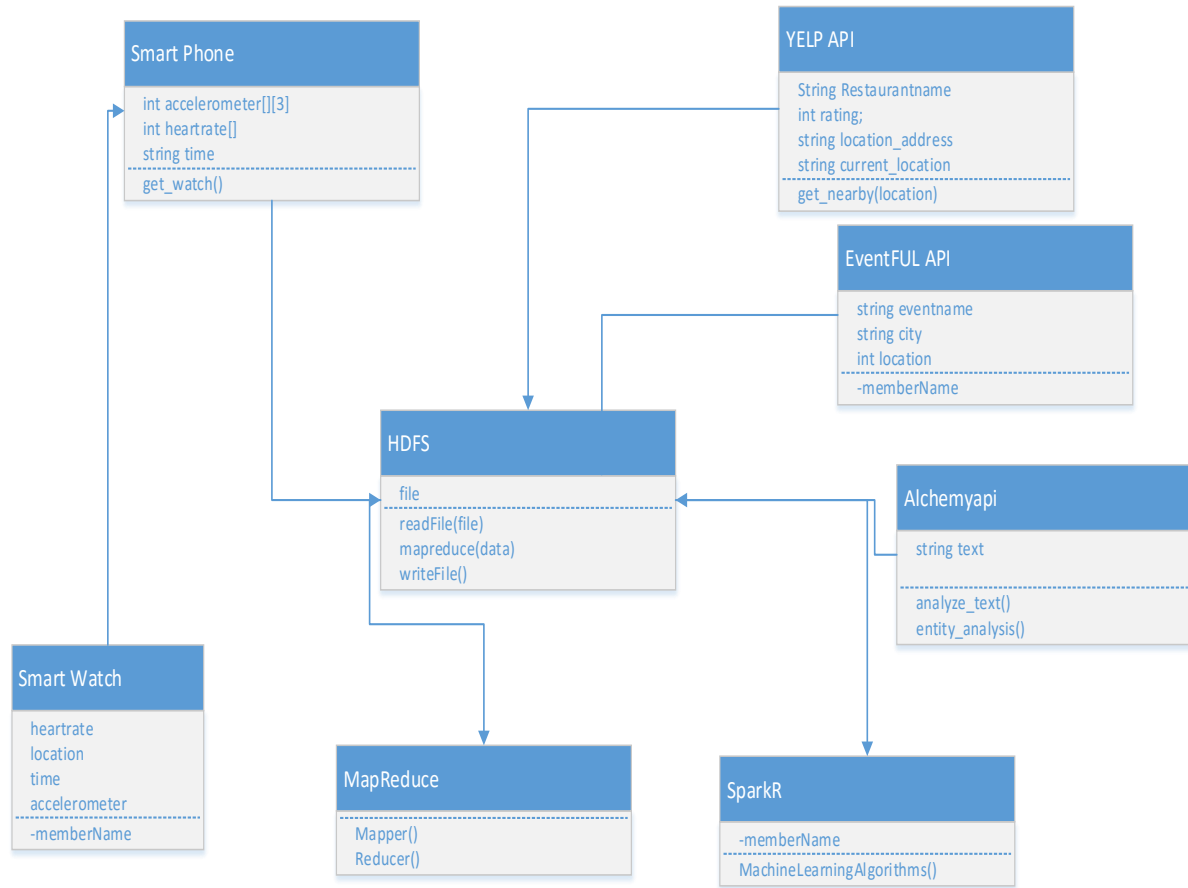
## 1. Design of Features:

### a. Architecture:



Our application begins and ends with the smartwatch user interface. The user will input to the watch. watch sends data to the Hadoop File System through the Android application. The Hadoop System will organize data that comes both from the user and from APIs. The application will then send data to the Amazon Web Services to run analytics on the data. We will implement MapReduce programs in Spark and Spark R. We will use several machine learning algorithms for prediction. After running the Query in spark, the data is sent back to the Android application, which can then be viewed by the user on their smartwatch.

## b. Class Diagram:



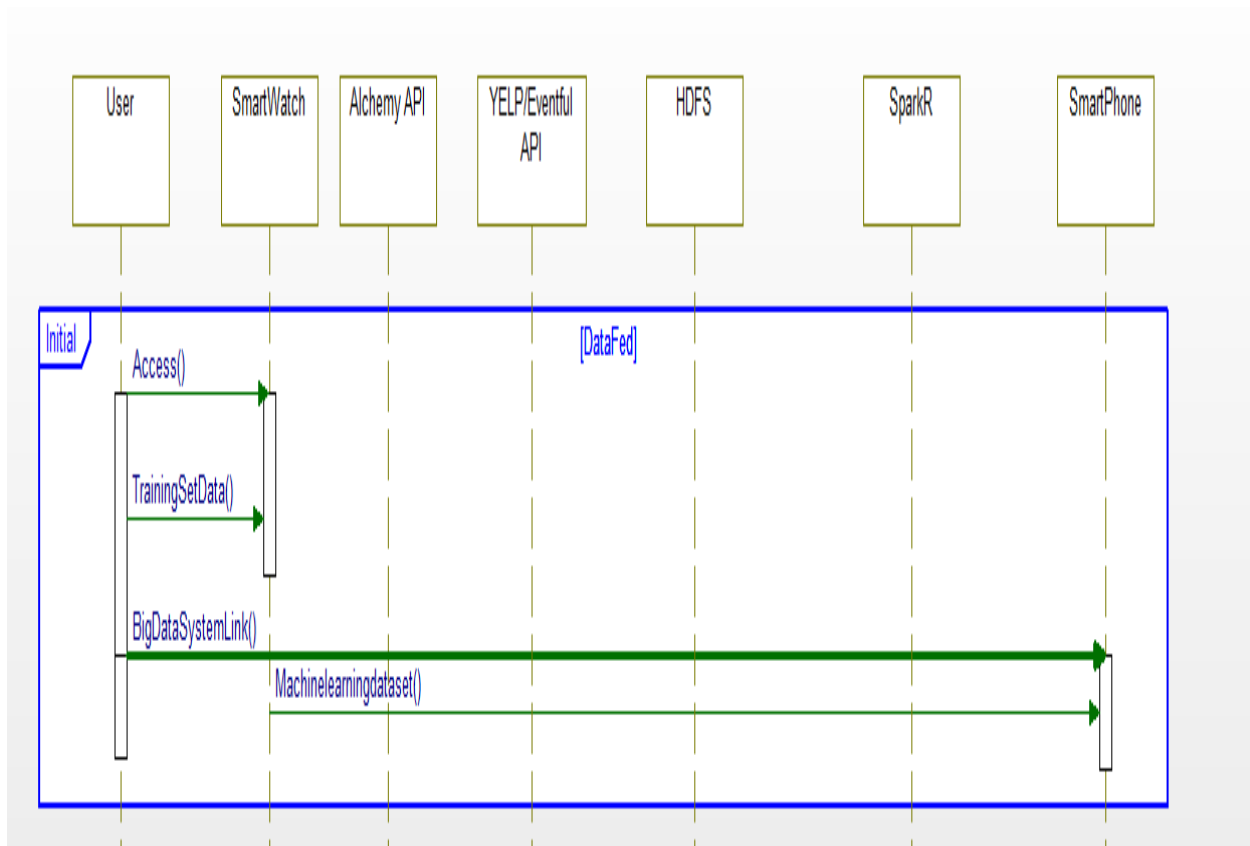
The class diagram we have drawn outlines the structure of our current project. We have three classes for three APIs, which provide data to the HDFS class. Also providing data to the HDFS is the smart phone, which calls `get_watch()` to retrieve smart watch data. The smart watch collects heartrate, location, time, and accelerometer data. HDFS will implement MapReduce and SparkR classes to analyze data it is given. SparkR will implement several of the machine learning algorithms, which will be discussed in more detail later in this paper.



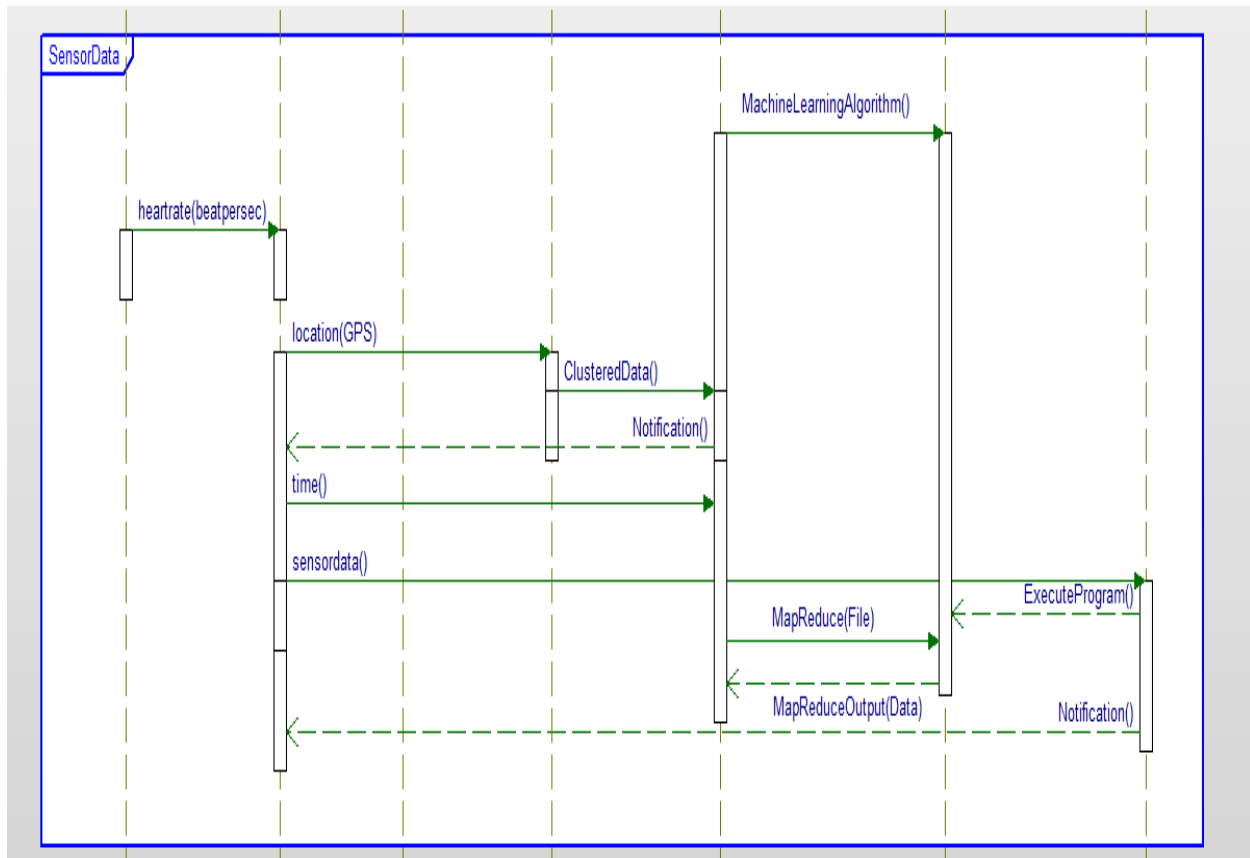
### c. Sequence Diagram:

The sequence diagram is shown as below. The sequence diagram is

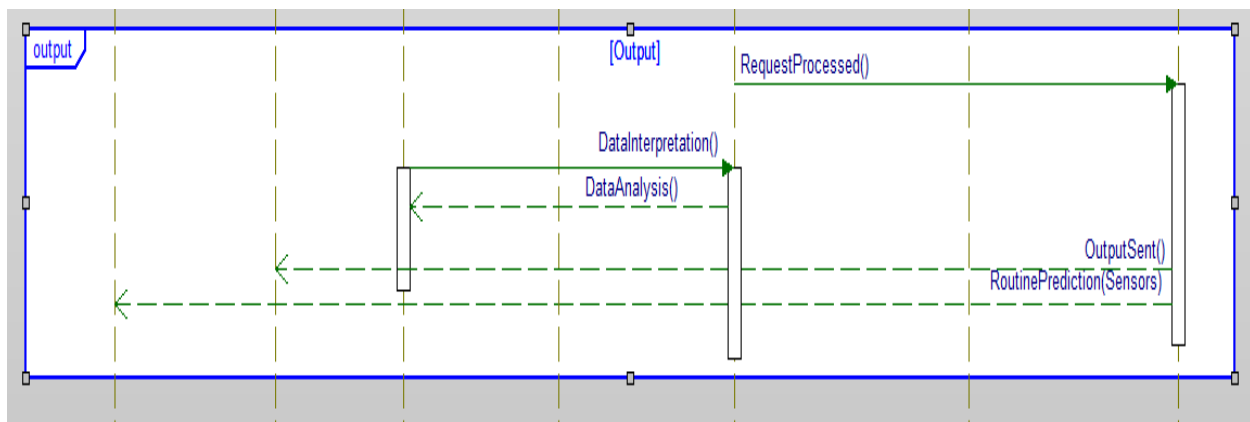
Disintegrated and attached:



The datafed process would mainly consist of all the classes to initially set up communication. The access and training set data is collected and then related with the smartwatch by the user as there is a setup of data fed into the remote machine. The machine learning dataset would be communicated through the smart phone and the device learns as to what the user pattern is.



The sensor data would basically do the operations involved mainly from the HDFS to obtain a programmed output that provides the user as to what the routine is or rather should be.



The final output would basically be a clear cut approach and the user gets a routine predicted.

#### **d. Machine Learning Algorithms:**

In Increment one, we have implemented a naïve bayes algorithm in R to classify datasets of user movements. The naïve bayes is a classifier that uses Bayes theorem to determine the most likely outcome. Bayes Theorem is written as such:  $P(A|B) = \frac{P(A)*P(B|A)}{P(B)}$ . In this equation, P(A) is the prior, P(B|A) is the likelihood, and P(B) is the evidence. This algorithm relies on P(A) and P(B) being independent of one another. While in reality, this is rarely the case, it provides an estimate that is extremely effective in classification.

In future increments, we plan on using a several more machine learning algorithms implemented on SparkR to scale larger sets of data.

#### **e. Datasets:**

A dataset is a collection of related sets of information iterated across an activity centric output. The dataset collected as input from the smartwatch will be fed via machine learning algorithms to the remote machine for it to be analyzed. The data/information is obtained from the different sensors which are required for providing an analytic comparison. Output is retrieved from the values fed. Supervised learning is in the initial stages since the machine is trained or rather intelligence provided regarding what to react in which case. After the training data set is provided, unsupervised learning comes to the frame. Here, the machine is able to analyze the data and compare with the training set. It will also be able to learn in case the user does an activity different from the routine. A notification is sent but nonetheless learning is also done by the machine, which in our case is the smart phone connected via USB.

## 2. Implementation

### Mobile Client Implementation (Smart Watch, Smart Phone, Robot):

#### Mobile Application for Collecting the Sensor Data:

- We have developed an application which is specifically for collecting the user smart phone and smart watch sensor data.
- In this application we have collected the Sensor data and we stored in to the csv files in the Device memory.
- Later we use this data to train over model to predict the user patterns regularly.
- This application mainly gets the all sensor data values from the Sensors such as Gyroscope, Accelerometer, Proximity, Light, Magnetic and Heart Rate in the Smart watch and smart phone.
- Apart from this we also collected the Location's latitude and longitude at which the sensor data collected and we store the data in csv file.

#### Sending and receiving data to Spark:

- We have developed an application which sends a string of text to Spark
- Spark analyzes the data, and returns a positive or negative sentiment using Sentiment Analysis
- Based on the statement, we are recommending to the user a list of movies

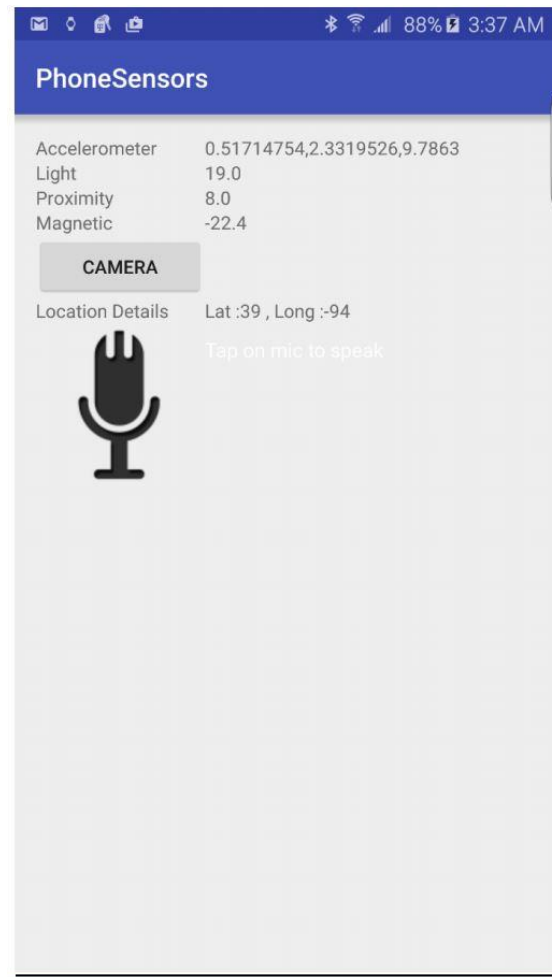
#### Machine Learning Application:

#### Machine Learning Classification Algorithm- Naïve Bayes Classification Algorithm to predict the User Activity:

- We have predicted the user activity using the Naïve Bayes Classification algorithm.
- We have trained the model based on the data obtained for different activities such as walking, getting out of bed and climbing up the stairs.
- So using the training data, we have predicted the user activity for the particular accelerometer readings using the Naïve Bayes Classification algorithm.
- Naïve Bayes Classification algorithm:
  1. Calculate the Mean and Standard deviation for each of column in the data set.
  2. The likelihood for each of the axis for activity is calculated.
  3. The likelihood of each of the axis reading was calculated using the Gaussian distribution function.
  4. Using the Naïve Bayes posterior probability formula we have computed the probability of occurrence of posterior.
  5. Posterior probability is directly proportional to product of prior probability and likelihood.
- Using the posterior probability we have predicted which activity is most likely to perform.
- The Machine learning algorithm runs using Spark

#### 4. Deployment:

##### 1. SmartPhone/SmartWatch Sensors Data Storage Application:

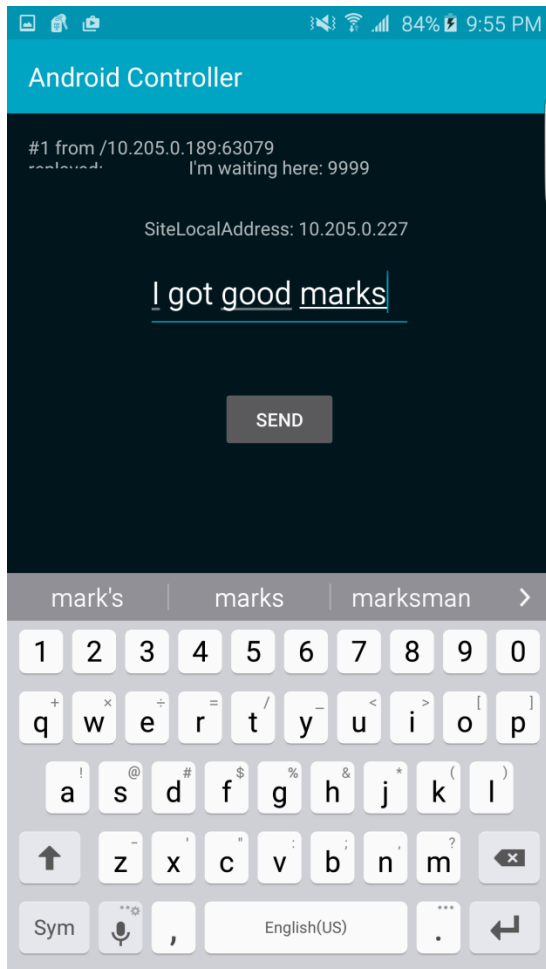


The various sensors in both mobile and smart watch gives the data values continuously for the user activities.

We store these values in the internal storage of the Device.

## 2. Sending Data to Spark :

In this application, the user can type a command and send it to Spark.



Here we see that the message has been received.

```
16/03/11 21:55:10 INFO DAGScheduler: ResultStage 1 (collect at MainStreaming.scala:25) finished in 0.027 s
16/03/11 21:55:10 INFO DAGScheduler: Job 8 finished: collect at MainStreaming.scala:25, took 0.064004 s
I got good marks
16/03/11 21:55:10 INFO BlockManagerInfo: Removed broadcast_1_piece0 on localhost:63060 in memory (size: 1037.0 B, free: 1127.2 MB)
16/03/11 21:55:10 INFO ContextCleaner: Cleaned accumulator 2
16/03/11 21:55:10 WARN : Your hostname, DESKTOP-P023Q7N resolves to a loopback/non-reachable address: fe80:0:0:0:5efe:acd:bd:net5, but we couldn't find any external IP address
16/03/11 21:55:11 INFO ReceiverSupervisorImpl: Starting receiver again
16/03/11 21:55:11 INFO ReceiverTracker: Registered receiver for stream 0 from 10.205.0.189:63018
16/03/11 21:55:11 INFO ReceiverSupervisorImpl: Starting receiver
16/03/11 21:55:11 INFO ReceiverSupervisorImpl: Starting receiver
```

### 3. Sentiment Analysis

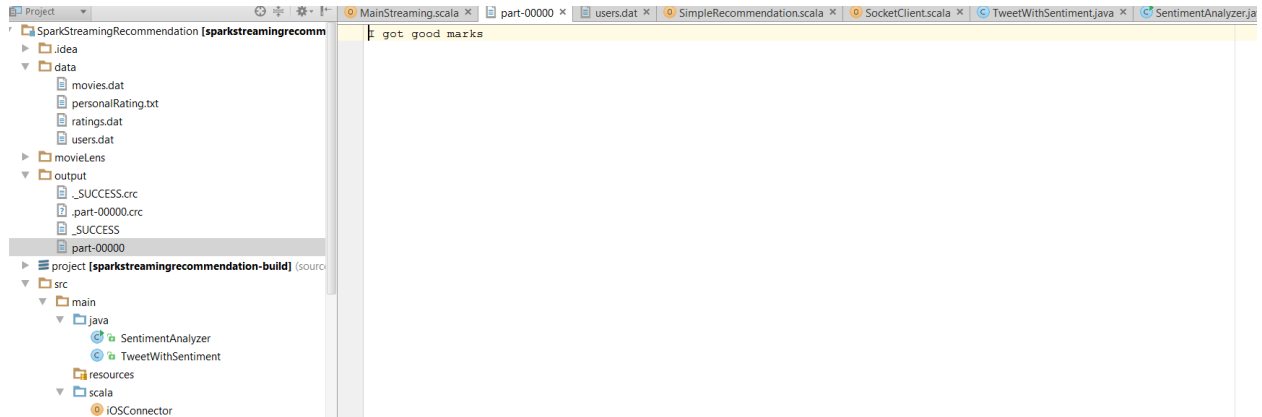
Once the message is received, we analyze the tweet for positive or negative sentiment. Below we show the code that we used.

```
.setAppName("SparkStreaming")
.set("spark.executor.memory", "4g").setMaster("local[*]")
val ssc= new StreamingContext(sparkConf,Seconds(2))
val sc=ssc.sparkContext
val ip=InetAddress.getByName("10.205.0.227").getHostName
val lines=ssc.socketTextStream(ip,9999)
// lines.saveAsTextFiles("output")
val command= lines.map(x=>{
  x.toString()
})
command.foreachRDD(
rdd=> rdd.collect().foreach(text => {
  println(text)
  rdd.saveAsTextFile("output")
  val sentimentAnalyzer: SentimentAnalyzer = new SentimentAnalyzer
  val tweetWithSentiment: TweetWithSentiment = sentimentAnalyzer.findSentiment(text)
  System.out.println("SENTIMENT is"+tweetWithSentiment)
  if(tweetWithSentiment.toString().contains("positive")){
    SimpleRecommendation.recommend(rdd.context )
  }
})
})
```

And the results:

```
16/03/11 21:55:13 ERROR ReceiverTracker: DeRegistered receiver for stream 0. Rescheduling receiver with delay 2000ms. Socket data size: 0
16/03/11 21:55:13 INFO ReceiverSupervisorImpl: Stopped receiver 0
done [1.3 sec].
Adding annotator sentiment
16/03/11 21:55:14 INFO JobScheduler: Added jobs for time 1457754914000 ms
SENTIMENT isTweetWithSentiment [line=I got good marks, cssClass=sentiment : positive]
16/03/11 21:55:14 INFO MemoryStore: Block broadcast_3 stored as values in memory (estimated size 59.6 KB, free 104.8 KB)
16/03/11 21:55:14 INFO MemoryStore: Block broadcast_3_piece0 stored as bytes in memory (estimated size 13.8 KB, free 118.6 KB)
16/03/11 21:55:14 INFO BlockManagerInfo: Added broadcast_3_piece0 in memory on localhost:63060 (size: 13.8 KB, free: 1127.2 MB)
```

We are also storing the text for recommendations:



#### 4. Creating Recommendations and sending them as notifications to the Smartphone

We want to send the user useful recommendations in this application. Therefore, we have created restaurant suggestions based on the user's preference.

First, we break it down into training and testing data. The code below shows how we have done this.

```
val numPartitions = 4
val training = ratings.filter(x => x._1 < 6)
    .values
    .union(myRatingsRDD)
    .repartition(numPartitions)
    .cache()
val validation = ratings.filter(x => x._1 >= 6 && x._1 < 8)
    .values
    .repartition(numPartitions)
    .cache()
val test = ratings.filter(x => x._1 >= 8).values.cache()

val numTraining = training.count()
val numValidation = validation.count()
val numTest = test.count()

println("Training: " + numTraining + ", validation: " + numValidation + ", test: " + numTest)
```

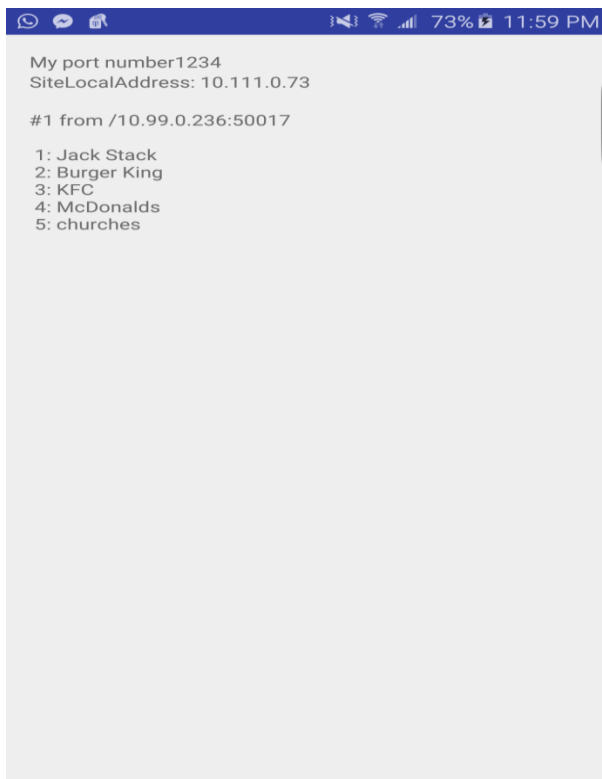
Next, we show the collaborative filtering model used to train the model:



```
// train models and evaluate them on the validation set

val ranks = List(8, 12)
val lambdas = List(0.1, 10.0)
val numIters = List(10, 20)
var bestModel: Option[MatrixFactorizationModel] = None
var bestValidationRmse = Double.MaxValue
var bestRank = 0
var bestLambda = -1.0
var bestNumIter = -1
for (rank <- ranks; lambda <- lambdas; numIter <- numIters) {
  val model = ALS.train(training, rank, numIter, lambda)
  val validationRmse = computeRmse(model, validation, numValidation)
  println("RMSE (validation) = " + validationRmse + " for the model trained with rank = "
    + rank + ", lambda = " + lambda + ", and numIter = " + numIter + ".")
  if (validationRmse < bestValidationRmse) {
    bestModel = Some(model)
    bestValidationRmse = validationRmse
    bestRank = rank
    bestLambda = lambda
    bestNumIter = numIter
  }
}
```

We send the results to the smartphone:

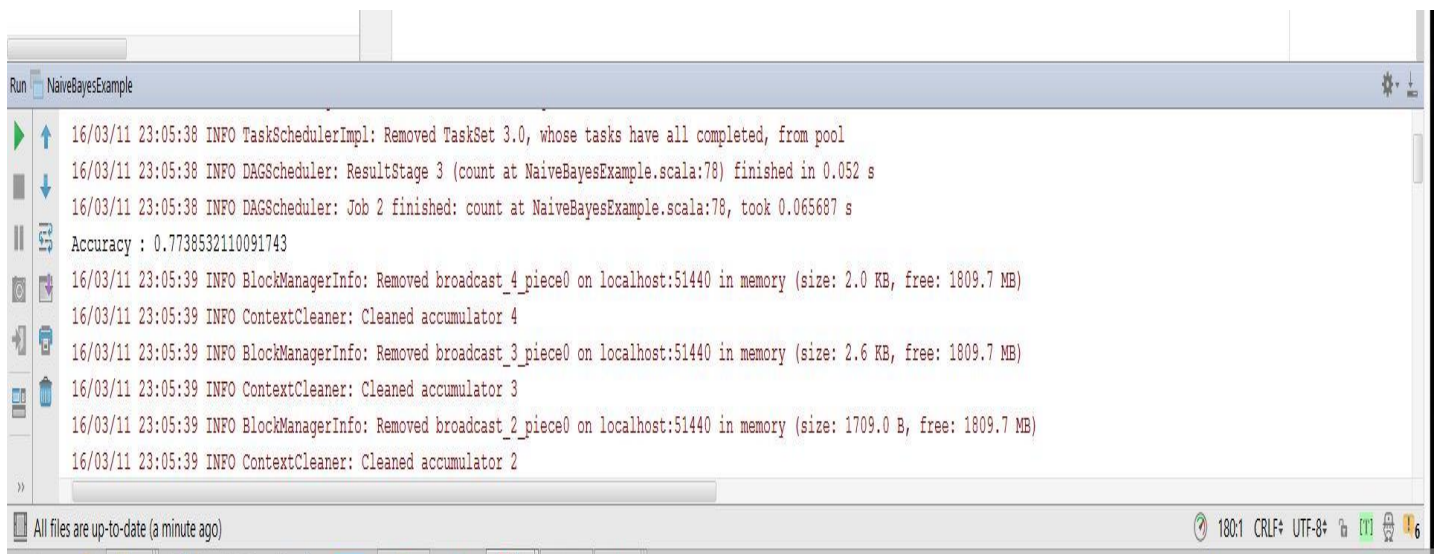


## Supervised Learning Algorithm Machine Learning Model:

### Classification of User movements and activities using the Naïve Bayes Classification:

- We have divided the data into two parts testing data and training data.
- Training data—60%
- Testing data ---40%
- We have trained the model to predict the different user activities and movements such as sitting up the chair, sitting down the chair, walking, sleeping etc.,
- We used the Multinomial distribution and we have **normalized** the input data values to fall between 0-1 to increase the accuracy.
- Normalized is done using MLLib function.
- After Normalizing the features, We have converted the RDD into **labelled Vector RDD**
- Accuracy is calculated by validating the results predicted from the testing data to the actual values. We have got accuracy of 77.38%
- We can use this model to classify the incoming data from the user.
- We have saved this model to predict the later data from the user. From this data we provide the overall stats of user time spending on activities.

#### 1. Model Accuracy – nearly 77%:



```
Run NaiveBayesExample
16/03/11 23:05:38 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
16/03/11 23:05:38 INFO DAGScheduler: ResultStage 3 (count at NaiveBayesExample.scala:78) finished in 0.052 s
16/03/11 23:05:38 INFO DAGScheduler: Job 2 finished: count at NaiveBayesExample.scala:78, took 0.065687 s
Accuracy : 0.7738532110091743
16/03/11 23:05:39 INFO BlockManagerInfo: Removed broadcast_4_piece0 on localhost:51440 in memory (size: 2.0 KB, free: 1809.7 MB)
16/03/11 23:05:39 INFO ContextCleaner: Cleaned accumulator 4
16/03/11 23:05:39 INFO BlockManagerInfo: Removed broadcast_3_piece0 on localhost:51440 in memory (size: 2.6 KB, free: 1809.7 MB)
16/03/11 23:05:39 INFO ContextCleaner: Cleaned accumulator 3
16/03/11 23:05:39 INFO BlockManagerInfo: Removed broadcast_2_piece0 on localhost:51440 in memory (size: 1709.0 B, free: 1809.7 MB)
16/03/11 23:05:39 INFO ContextCleaner: Cleaned accumulator 2
All files are up-to-date (a minute ago)
180:1 CRLF+ UTF-8+ 6
```

## 2. Displaying Confusion Matrix for the User Activity :



The screenshot shows a Java IDE window titled "Run NaiveBayesExample". The console output displays ten identical lines: "Predicted rating for the moves is: (1.0,1.0)". The IDE interface includes a toolbar on the left with icons for running, debugging, and other actions. The status bar at the bottom indicates "All files are up-to-date (2 minutes ago)" and shows the current line number as 59045.

```
Run NaiveBayesExample
Predicted rating for the moves is: (1.0,1.0)
Predicted rating for the moves is: (1.0,1.0)
Predicted rating for the moves is: (1.0,1.0)
Predicted rating for the moves is: (1.0,1.0)
Predicted rating for the moves is: (1.0,1.0)
Predicted rating for the moves is: (1.0,1.0)
Predicted rating for the moves is: (1.0,1.0)
Predicted rating for the moves is: (1.0,1.0)
Predicted rating for the moves is: (1.0,1.0)
Predicted rating for the moves is: (1.0,1.0)
```

All files are up-to-date (2 minutes ago) 59045 CRLF+ UTF-8+ 6

## 5. Project Management

### Increment 2:

The following image shows how the image is categorized and the implementation using the Zenhub Tool.

#### Graphs:

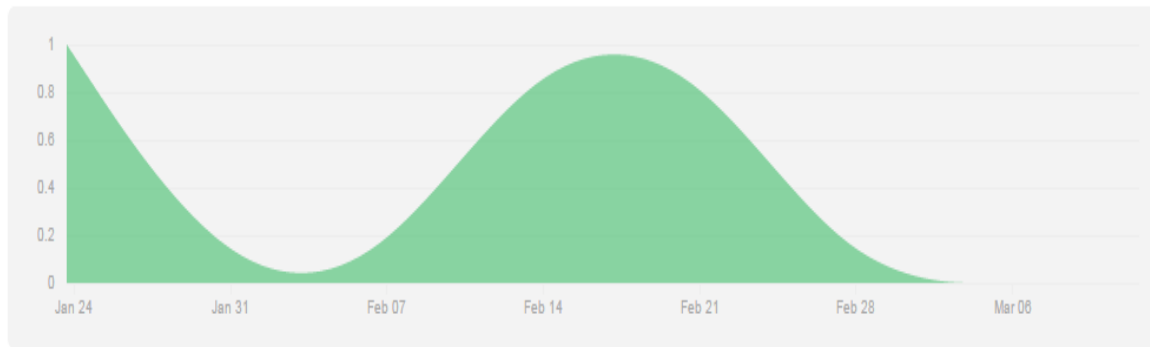
The following graphs show the analysis of the project management done over the period between first and second increment.

#### Contributors:

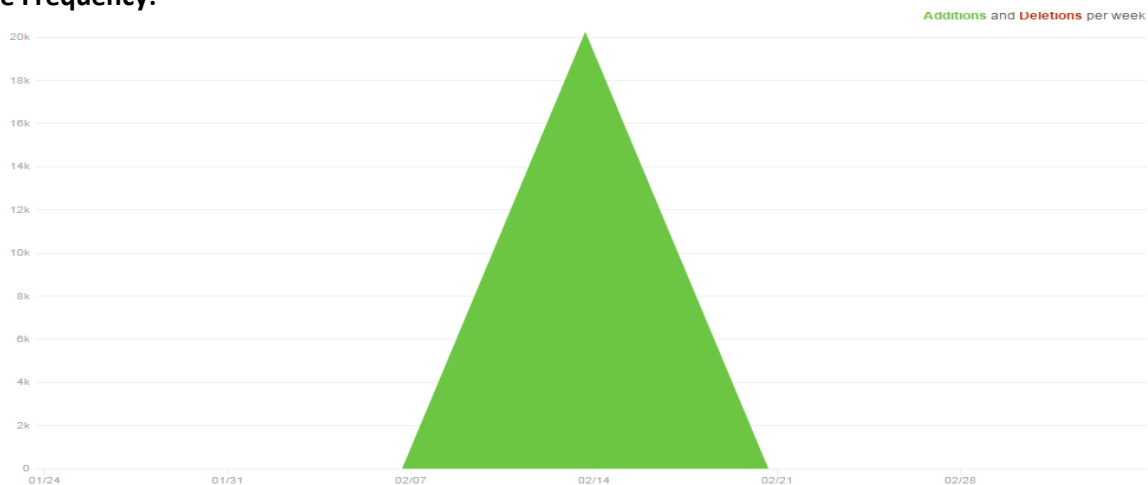
Jan 24, 2016 – Mar 12, 2016

Contributions: Commits ▾

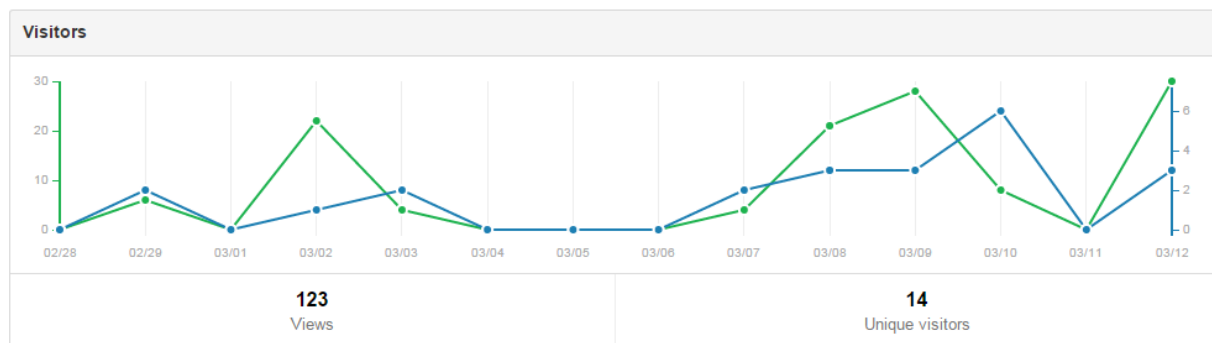
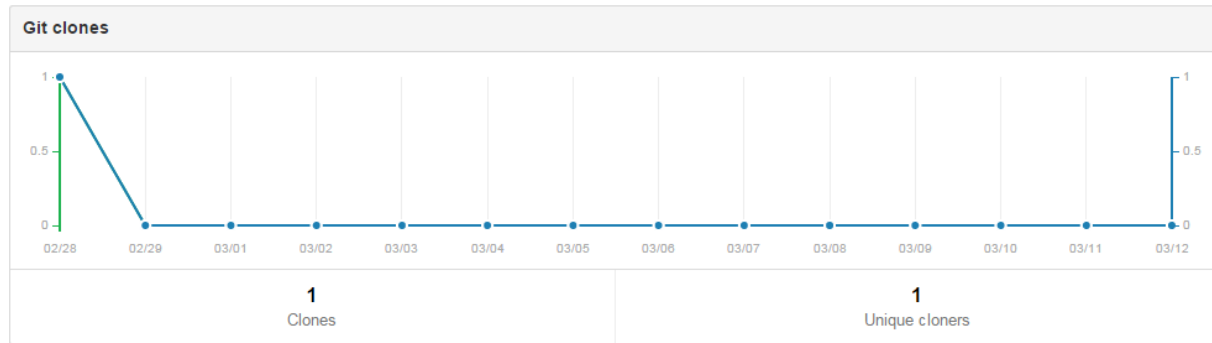
Contributions to master, excluding merge commits



#### Code Frequency:



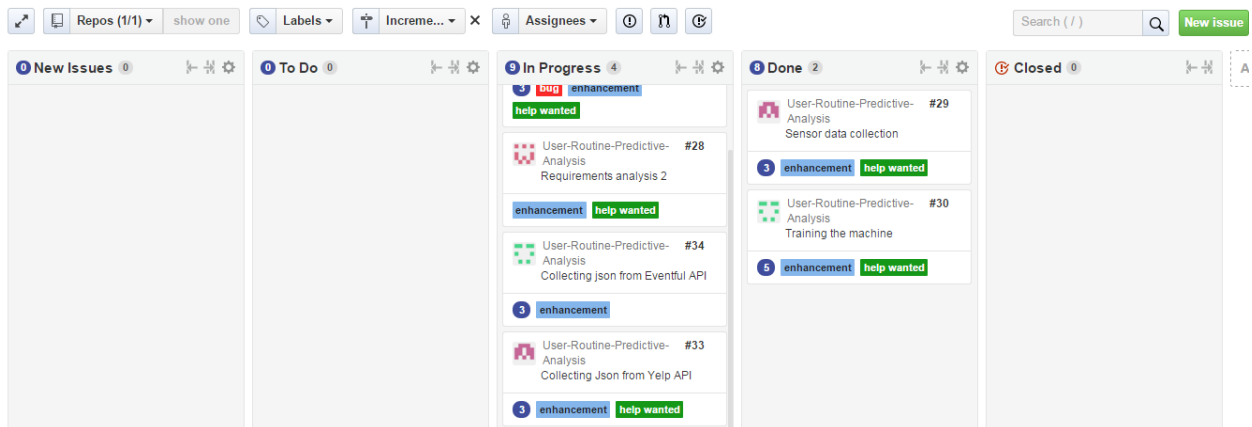
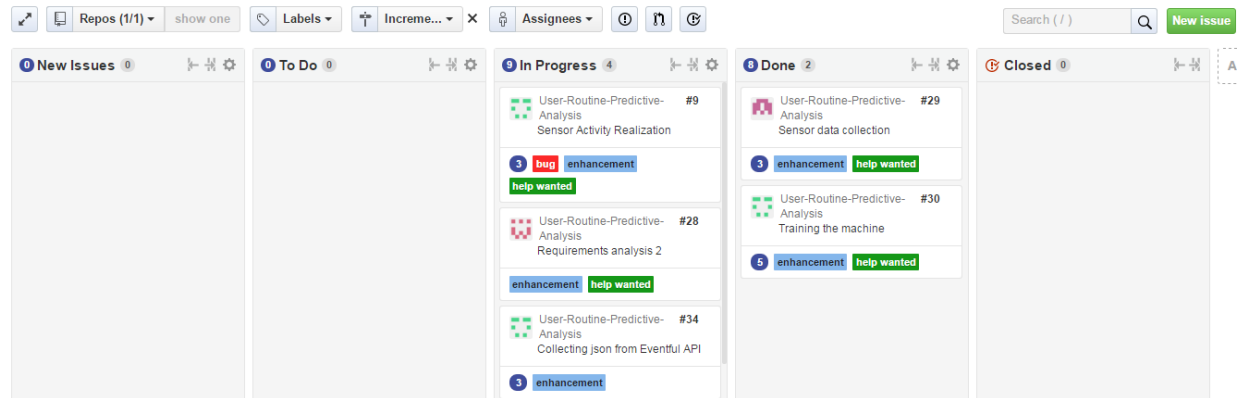
## Traffic :



## Commits:



## Increment2 ZenHub tool:



- Implementation status report:

**a. Work completed**

Member	Task Description	Member Responsibility	Contribution %	Time (hours)	Comments /Issues
<b>Rakesh</b>	1. Implementing Naïve Bayes Machine Learning Algorithm in Spark. 2. Collecting data from smart phone.	Develop, design, build and testing the task	100	22	
<b>Nihar</b>	1. Stream data from Smart phone to Spark. 2. Sentimental Analysis on the text received from the user	Develop, design, build and testing the task	100	20	
<b>Mark</b>	1. Recommendation system using Collaborative Filtering 2. Collecting restaurant data for the recommendation system.	Develop, design, build and testing the task	100	22	
<b>Sricharan</b>	1. Sending the recommendations to Smart phone. 2. Sentimental Analysis.	Develop, design, build and testing the task	100	20	

**b. Work yet to be completed**

Member	Task Description	Member Responsibility	Time (hours)	Comments /Issues
<b>Rakesh</b>	1. Implementing Machine Learning Algorithms in Spark R 2. Sending notification of predicted data. 3. Predicting the user patterns regularly. 4. Implementing API's.	Develop, design, build and testing the task	40	
<b>Nihar</b>	1. Implementing Machine Learning Algorithms in Spark R. 2. Collecting more sensor data sets. 3. Predicting the user patterns regularly. 4. Implementing API's.	Develop, design, build and testing the task	40	
<b>Mark</b>	1. Implementing Machine Learning Algorithms in Spark R 2. Sending notification of extracted data 3. Improving GUI. 4. Predicting the user patterns regularly.	Develop, design, build and testing the task	40	
<b>Sricharan</b>	1. Implementing Machine Learning Algorithms in Spark R 2. Sending notification of extracted data. 3. Testing the implemented features. 4. Predicting the user patterns regularly.	Develop, design, build and testing the task	40	



## 6. Bibliography:

- i. Komninos, Andreas, and Mark Dunlop. "Text input on a smart watch." Pervasive Computing, IEEE 13.4 (2014): 50-58.
- ii. Lutze, Rainer, and Klemens Waldhor. "A Smartwatch Software Architecture for Health Hazard Handling for Elderly People." Healthcare Informatics (ICHI), 2015 International Conference on. IEEE, 2015.
- iii. <http://www.motorola.com/us/products/moto-360>
- iv. <https://www.thegrommet.com/robome-app-enabled-robot>
- v. <https://www.yelp.com/developers/documentation/v2/overview>
- vi. <https://api.eventful.com/>