CHAPTER 1

INTRODUCTION

1.1 General

Recommender systems is a family of methods that enable filtering through large observation and information space in order to provide recommendations in the information space that user does not have any observation, where the information space is all of the available items that user could choose or select and observation space is what user experienced or observed so far.

1.2 Motivation

The traditional way to advertise has been to create a single offer for a channel and to use it across in all platforms. But as the systems become more intelligent it becomes a necessity to provide better recommendations to each customer based on their interests and history with product purchase. As we encounter more choices, customer retention as well new customer base becomes extremely important to survival of business.

Recommendations play their important role because they allow creation of user profiles and using attributes like location we can extrapolate the user similarities and combined with purchase history provide accurate, appealing and optimized advertisements to the consumer.

Older systems face problems as,

1. They are based on One-for-all concept, which is becoming increasingly irrelevant in today's data driven world.
2. They are limited to physical forms.
3. They do not provide for any way to appeal to older-loyal customers.
4. They do not draw on how users of similar profile buy what product for what reason.
5. Scalability and Adaptability is a major issue.

A computer based recommender systems hence provides a much better alternative to the user. Not only they do not have these shortcomings of the traditional methods, but also they can mine the historical information of the user and demographics information, which may result in a more accurate and finely tuned recommendation.

1.3 Problem description

The idea is to build an analytics application using Apache spark that takes data as input, processes the data and provide recommendations. The application will be a specialized recommendation system using collaborative filtering, which can provide domain dependent recommendations (in this case user locations) which involves processing big data. Current systems used in advertising employee 'One-for-all' technique that is not applicable in today's scenario.

We propose a wearable device with an audio output that shall relay personalized marketing offers to the user. The system will not only help users get best benefits but the manufacturers and service providers will be able to learn of generalized and anonymous users interests and shopping patterns that can greatly enhance their input.

1.4 Related Work

1.4.1 Industry Usage: In industry, considering the usage, the most widely known recommender system could be attributed to Amazon. Purchase history, using browser history and user history, they provided recommendations to the user for a variety selection of goods and products. Currently, many companies that sell a selection of products and have access to user information, employ a recommender system that promotes further purchases to the user.

1.4.2 Richness of the Ecosystem: Difference business and product needs and a variety of algorithms that could be used for recommender systems yielded a rich set of methods that could be used for recommendations.

1.4.3 Connection to Information Retrieval: This subsection of machine learning methods also have connections with information retrieval and actually the problem could be formulated as an information retrieval problem as well. Consider Google, the links (items) are brought to the first page to the users based on query information, location, user history and so on. Therefore, most of the algorithms invented in information retrieval could be adopted to recommendation systems with minimal changes. The reverse may not hold true in general, though.

System Requirements

- sbt
- Scala (2.10.3)
- Eclipse scala (Keplar version)
- Hadoop
- Hive
- Spark
- Spark-SQL
- Shark
- Thunderain

1.5 Report Organization

- Chapter 1 provides the general problem statement and the motivation behind implementing recommender system
- Chapter 2 provides an overview of the system and its concepts. It also discusses the architectural design and its description
- Chapter 3 lists the functional and non-functional requirements and also presents the detailed analysis and design.
- Chapter 4 discusses the software tools, methodology and implementation
- Chapter 5 presents the results obtained from the recommender system and its detailed performance analysis
- Chapter 6 provides a summary of the results of recommender system and also proposes future enhancements

CHAPTER 2

PROJECT DESCRIPTION AND GOALS

2.1 Project Description

The idea is to build a wearable device which is aptly supported by an analytics engine that relays personalized marketing solutions to end-users. The analytics engine will be a specialized recommendation system using collaborative filtering, which can provide domain dependent recommendations (in this case user locations) which involves processing big data (former user transactions). The project has several software and hardware modules.

2.1.1 Android GPS Application: The android application constantly sends GPS coordinates to a web server in 60 seconds time intervals. This light weight app runs as a background thread which facilitates low battery usage.

2.1.2 Spark Recommendation System: Uses ALS (Alternating Least Square) item based collaborative filtering algorithm to generate recommendations for each user. These personalized recommendations are generated in the form of per user per product rating matrix.

2.1.3 PHP Server Application: The php script authenticates the users, receives the geolocation data sent by the Android device, maps the latest GPS data with the personalized recommendations for the corresponding user and generates the audio file for relaying to the Arduino.

2.1.3 Arduino: In industry, considering the usage, the most widely known recommender system could be attributed to Amazon. Purchase history, using browser history and user history, they provided recommendations to the user for a variety selection of goods and products. Currently, many companies that sell a selection of products and have access to user information, employ a recommender system that promotes further purchases to the user.

2.2 Goals

The goal is to design the product such that it relays personalized recommendations. The traditional way to advertise has been to create a single offer for a channel and to use it across in all platforms. But as the systems become more intelligent it becomes a necessity to provide better recommendations to each customer based on their interests and history with product purchase. As we encounter more choices, customer retention as well new customer base becomes extremely important to survival of business. Recommendations play their important role because they allow creation of user profiles and using attributes like location we can extrapolate the user similarities and combined with purchase history provide accurate, appealing and optimized advertisements to the consumer.

Older systems face problems as,

1. They are based on One-for-all concept, which is becoming increasingly irrelevant in today's data driven world.
2. They are limited to physical forms.
3. They do not provide for any way to appeal to older-loyal customers.
4. They do not draw on how users of similar profile buy what product for what reason.
5. Scalability and Adaptability is a major issue.

A computer based recommender systems hence provides a much better alternative to the user. Not only they do not have these shortcomings of the traditional methods, but also they can mine the historical information of the user and demographics information, which may result in a more accurate and finely tuned recommendation.

CHAPTER 3

TECHNICAL SPECIFICATION

3.1 Software Specifications

The following software modules were programmed with certain specifications.

3.1.1 Android GPS Application and Server Script: The android application facilitates swift transfer of geolocation data at regular time intervals to the web server. The app leverages the concept of Location Listeners, Intent Services, Alarm Manager and Async Tasks to accomplish the former task. The detailed description of each component is as under:

Location Listeners: The Location Listener interface is used for determining user location. It uses cell tower triangulation, GPS data and Wi-Fi signals for providing location information in a way that works indoors and outdoors, responds faster, and uses less battery power. It uses callback mechanism to trigger events whenever the user location changes or the status of the service changes. It triggers the asyn task which transfers GPS data to the server, on receiving the location change callback.

Intent Services: A Service is an application component that can perform long-running operations in the background. An intent service facilitates uninterrupted retrieving of location data using lightweight background threads. These background threads do not require a UI. The Location listeners are implemented in the form of background tasks using the intent services. This implementation technique enables un-intervened fetching of location data without affecting the performance of the application.

Alarm Manager: The alarm manager facilitates scheduling application to be run at some point in the future. This task scheduling ability is used for fetching geolocation data at regular time intervals. The alarm manager triggers the Location Listener service, every 60 seconds for fetching GPS coordinates. This approach is adequate for battery saving, since the application thread works on the sleep and wake up mechanism rather than continuously running in the background.

Async Task: The asyn task enables proper and easy use of the UI thread. The asyc task runs asynchronous to the main UI thread and tackles the unresponsive UI thread problems. This ability of asyc task is leveraged for asynchronously transferring GPS data to a remote server using the HTTP Post request. The data sent to the server contains username, password, latitude, longitude and timestamp

3.1.2 Spark Recommendation System: The Spark Recommendation system uses the collaborative filtering algorithm which has been implemented in the Mllib, the Machine Learning Library for Apache Spark. The algorithm was implemented in Scala using the Eclipse IDE.

Collaborative Filtering: Collaborative filtering is commonly used for recommender systems. These techniques aim to fill in the missing entries of a user-item association matrix, in our case, the user-movie rating matrix. MLlib currently supports model-based collaborative filtering, in which users and products are described by a small set of latent factors that can be used to predict missing entries. In particular, we implement the alternating least squares (ALS) algorithm to learn these latent factors.

3.1.3 PHP Server Application: It is the central server connecting Android application and the Arduino. It also houses the database functionality. The server authenticates the users, receives the geolocation data sent by the Android device, maps the latest GPS data with the personalized recommendations for the corresponding user and generates the audio file for relaying to the Arduino.

The functionality of the server can be split into the following modules:

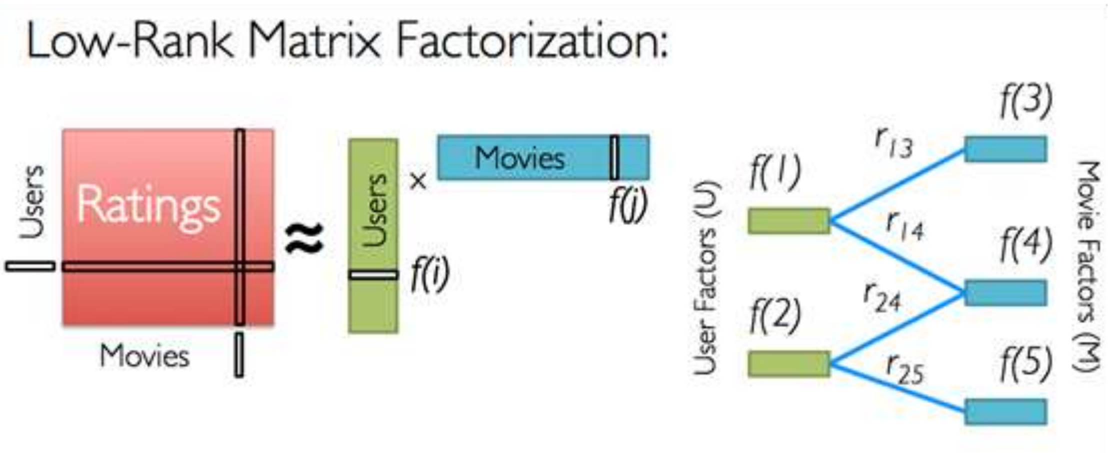
Database service: This module provides database connectivity to the project. The database contains a User Authentication table and a Mapping table. The user authentication table is used for authenticating the user credentials and implementing access controls and restrictions. The Mapping table is a lookup table which tags a latitude, longitude pair to its alphanumeric location description (here shop name).

Location module: The server receives GPS data every 60 seconds. The received data contains latitude longitude pair. For generating recommendations, this pair needs to be mapped to the shop closest to the user. Vincenty great circle distance is used for finding the distance of current user location with every shop location present in the Mapping table. The current user location is then mapped to the closest located shop. The Vincenty great circle distance formula is give by:

$$\Delta\sigma = 2\arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1\cos\phi_2\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

$\Delta\phi$ = (latitude1 – latitude2), $\Delta\lambda$ = (longitude1 – longitude2) and $\Delta\sigma$ indicates the difference between two locations.

Text to Speech module: The audio recommendation module receives the nearest shop name from the location module. It filters the output of the Spark Recommendation Engine to find out top rated items, from the nearest shop, that might interest the user. The top 'n' recommended items are converted into mp3 audio file which is relayed to the Arduino. The Google Translate Text to Speech Api is used for generating audio files from the text.



Alternating Least Square (ALS): We have users u for items i matrix as in the following:

$$Q_{ui} = \begin{cases} r & \text{if user u rate item i} \\ 0 & \text{if user u did not rate item i} \end{cases}$$

Where r is what rating values can be. If we have m users and n items, then we want to learn a matrix of factors which represent movies. That is, the factor vector for each movie and that would be how we represent the movie in the feature space. Note that, we do not have any knowledge of the category of the movie at this point. We also want to learn a factor vector for each user in a similar way how we represent the movie. Factor matrix for movies $Y \in R^{fxn}$ and factor matrix (each movie is a column vector) for users $X \in R^{mxf}$ (each user is a row vector). However, we have two unknown variables. Therefore, we will adopt an alternating least squares approach with regularization. By doing so, we first estimate Y using X and estimate X by using Y. After enough number of iterations, we are aiming to reach a convergence point where either the matrices X and Y are no longer changing or the change is quite small. However, there is a small problem in the data. We have neither user full data nor full items data, this is also why we are trying to build the recommendation engine in the first place. Therefore, we may want to penalize the movies that do not have ratings in the update rule. By doing so, we will depend on only the movies that have ratings from the users and do not make any assumption around the movies that are not rated in the recommendation. Let's call this weight matrix w_{ui} as such:

$$w_{ui} = \begin{cases} 0 & \text{if } q_{ui} = 0 \\ 1 & \text{else} \end{cases}$$

Then, cost functions that we are trying to minimize is in the following:

$$J(x_u) = (q_u - x_u Y) W_u (q_u - x_u Y)^T + \lambda x_u x_u^T$$

$$J(y_i) = (q_i - X y_i) W_i (q_i - X y_i)^T + \lambda y_i y_i^T$$

We need regularization terms in order to avoid the over fitting the data. Ideally, regularization parameters need to be tuned using cross-validation in the dataset for algorithm to generalize better. Solutions for factor vectors are given as follows:

$$x_u = (YW_uY^T + \lambda I)^{-1} YW_u q_u$$

$$y_i = (X^T WiX + \lambda I)^{-1} X^T W_i q_i$$

where $W_u \in R^{nxn}$ and $W_u \in R^{mxm}$ diagonal matrices.

3.2 Hardware Specifications

ABHINAV

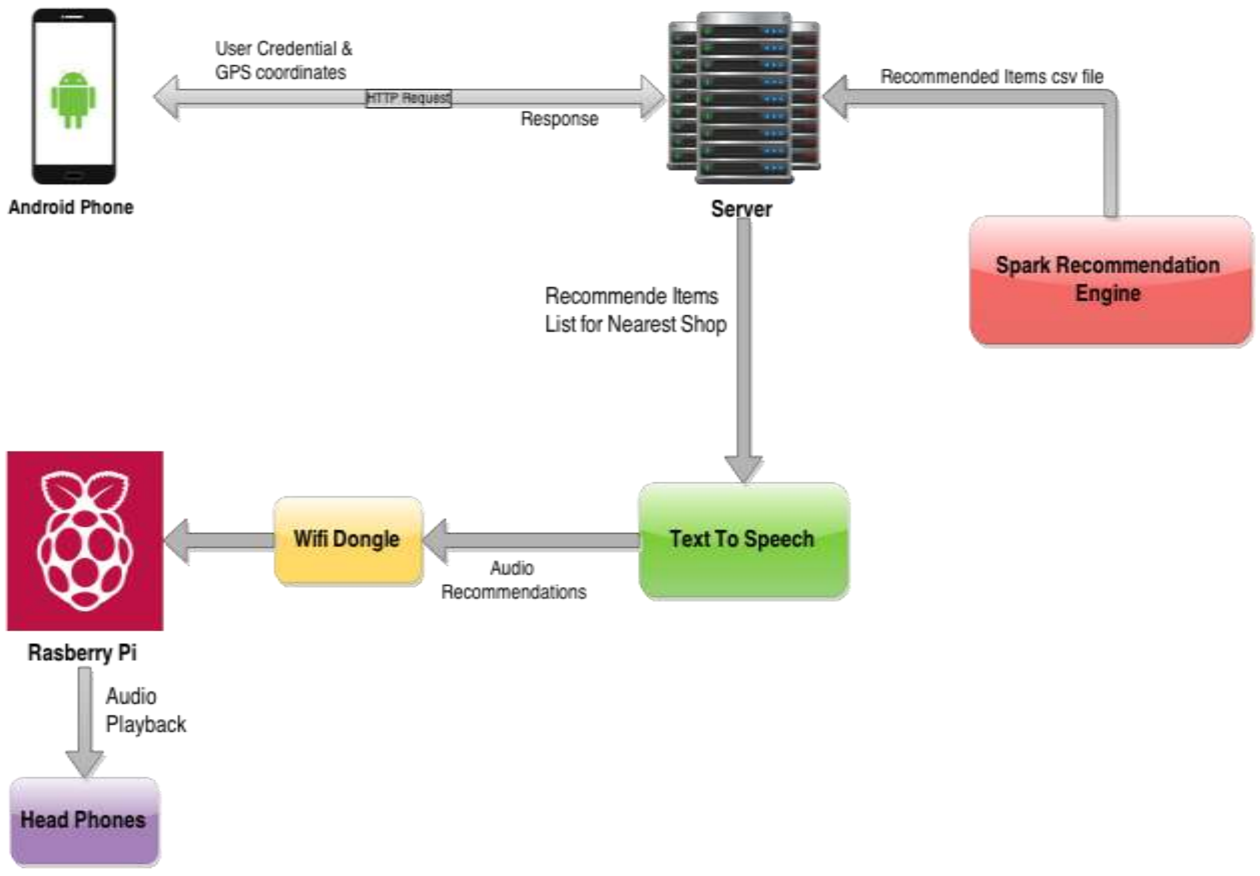
3.2.1 Arduino:

CHAPTER 4

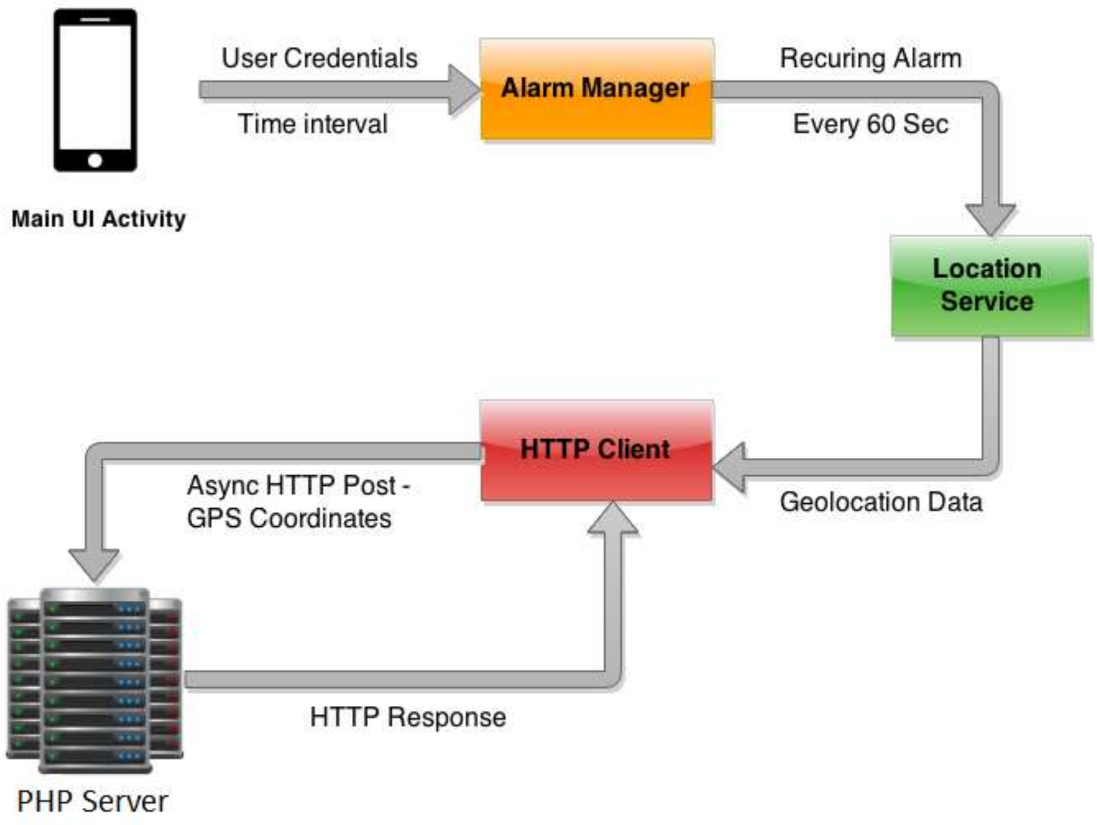
DESIGN APPROACH AND DETAILS

4.1 Design Approach

The project workflow is as follows:



4.1.1 Android GPS Service: The Android GPS Service which transmits the GPS Coordinates has the following workflow:



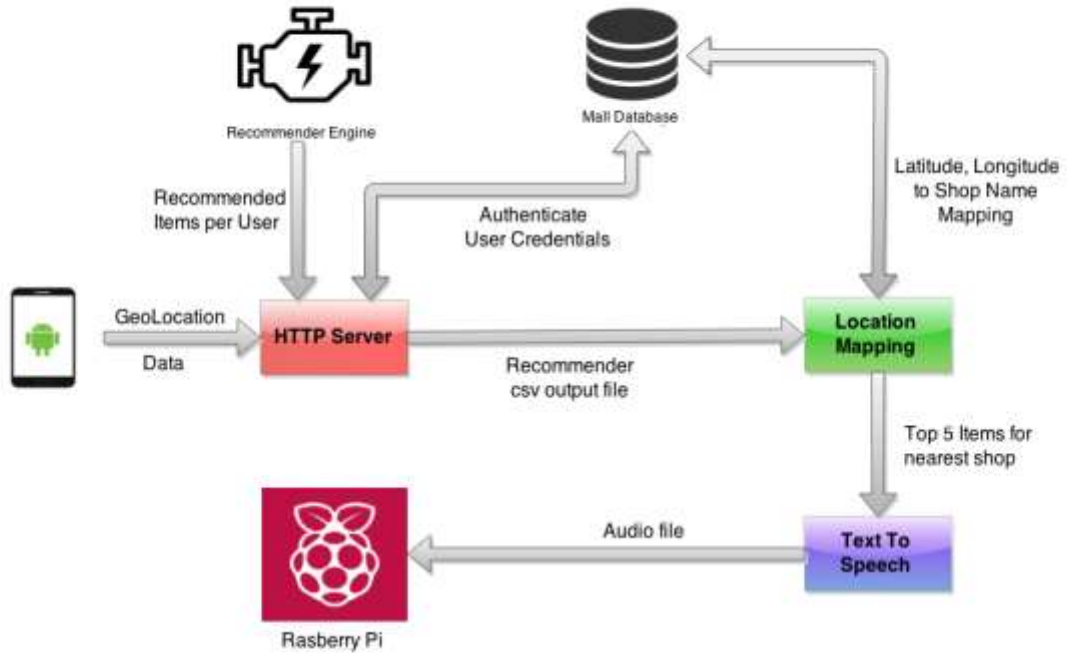
Main Activity: The android main activity is the UI for user interaction. It contains phone number, password textbox, Time Interval radio button and a Start shopping button. On pressing the start shopping button, the entered textboxes are validated. On successful validation the alarm manager is triggered with the provided time interval. The default time interval is 60 seconds.

Alarm Manager: The alarm manager sets a recurring alarm for the time interval provided by the user. The alarm gets triggered every 'n' seconds (default n = 60) and makes a call to the Location Service Class.

Location Service: The location services class is responsible for requesting geolocation data from the best location provider. It extracts time, latitude, longitude and accuracy from the retrieved geolocation data and sends it to the Http Client.

HTTP Client: This is an async task which runs asynchronous to the main UI thread. It is responsible for transferring geolocation data to the server. The method used for data transfer is HTTP Post. Based on the response from the server, the http client retries the request in case of failed transfer or kills itself in case of successful transfer.

4.1.2 PHP Server Scipt: The PHP Script that maps GPS coordinates with recommendations has the following workflow:

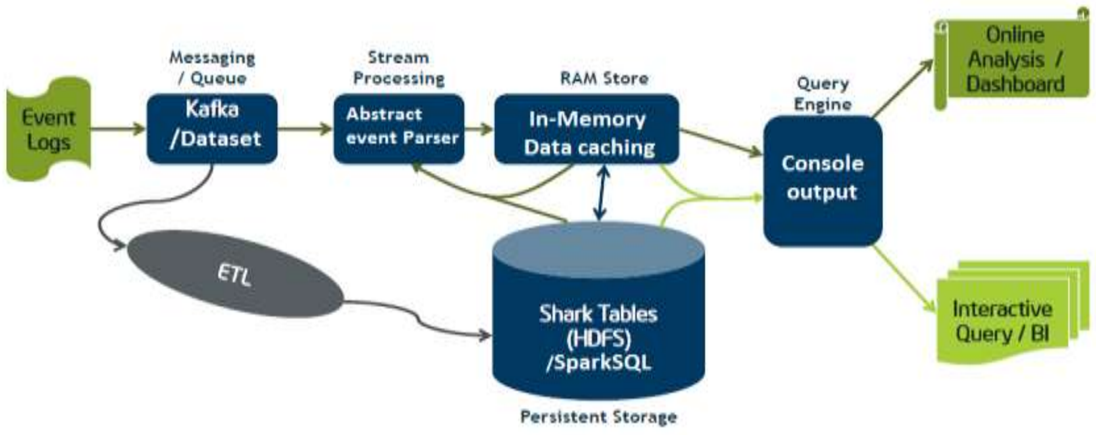

Recommender Engine
Mall Database
Recommended Items per User
Authenticate User Credentials
Latitude, Longitude to Shop Name Mapping
GeoLocation Data
HTTP Server
Recommender csv output file
Location Mapping
Top 5 Items for nearest shop
Audio file
Text To Speech
Rasberry Pi

HTTP Server: The server script is written in PHP. The server receives the output csv file from the spark recommender system. It also receives user credentials and geolocation data from the android phone. The user credentials are validated for authenticity using the database. On successful authentication, the recommender output is parses and sent to the location mapping script along with the geolocation data.

Location Mapping: This script uses the mall database to find a shop which is nearest to the use, by calculating Vincenty great circle distance between geolocation data of the user and all the shops. It then finds the top 'n' items (default n=5) that corresponding to the user and the nearest shop. The top items list is then sent to the Text to Speech script.

Text to Speech: The top items list is passed to this script from the location mapping script. This script uses Google translator's text to speech Api to convert the top items list into audio files. These files are then sent to the Rasberry Pi.

4.1.3 Analytical Engine: The Spark Analytical Recommendation Engine follows the ETL model (Extract, Transform and Load). The workflow is best described as follows:



• The dataset is acquired through the various event logs which is assigned a permanent location in the memory

• This location is used by the Abstract event Parser to obtain the data and generate 'Event' which roughly represent any row of the dataset

• These entities of these events (attribute value) is then mapped to the respective schema before being transferred to the training operator as Dstream[Event]

• A model is generated from these parsed events and cached in the memory for future references

• The personalized recommendations are then generated as output and can then be sent to the console, stored in the Shark Tables or represented through SparkSQL.

• The output recommendations are then mapped with the incoming GPS Coordinates to generate top recommendations for the corresponding user and shop, which can then be sent to an online analytic application (which may represent it graphically) or can be interactively queried to generate the audio messages.

4.1.4 Raspberry Pie

4.2 Codes and Standards

The PENDING

4.3 Constraints, Alternatives and Tradeoffs

This section discusses the various constraints, alternatives and tradeoffs during the designing, prototyping and assembling of the final product.

4.3.1 Constraints: The constraints can be classified as follows:

Hardware Constraints: As a finished product, the system must meet the following requirements:

- It must not be too heavy for usage
- It must be efficiently packed
- It must offer easy mobility and
- It must be robust in its handling

These constraints can be met with decreasing the size of the individual modules and compacting to one single entity.

Software Constraints: As the data collection and processing is intensive there is a need to optimize the code.

Manufacturability Constraints: Housing the entire setup in one device will be difficult to accommodate considering that the wearable must be light weight and easy to handle. Initial cost of production may be high.

Usability Constraints: We aim to make this system as simple and usable as possible. We wish to deploy a simple system that is deployed over the headphones and connected to the server for seamless transmission and receiving of data.

Social Constraints: Privacy risks in research relate to the identification of participants, and the potential harms they or groups to which they belong, may experience from the collection, use and disclosure of personal information.

Economic Constraints: Two competing companies or manufacturers may not appreciate trade data to become knowledge to a third party vendor. As such, these constraints can arise not due to want to hide data but as a need to protect confidentiality over various aspect majorly finances and logistic.

Political Constraints: As this project is built more on the terms of business customer interface political issue make little entry into it. By making all the permission needed transparent and completely anonymous any political issues can be curbed or avoided. This project is purely for improving the quality of offers he/she gets.

Health and Safety Constraints: Health and safety play a very important role today. By ensuring that the device uses standards for communication and appropriate protocols for security we can ensure that there are no side effects to health and users' privacy is maintained.

4.3.2 Alternatives: Several alternatives were considered during the deigning and prototyping phase of the project and their inclusion was debated and analyzed.

GPS Sensor: A GPS sensor was included in the prototype to send user's GPS location to the analytical engine. It was later discarded due to its bulky size which did not fit well while packaging the final product. Also, the GPS coordinates were not quite sensitive for closed small range locations.

FM Trans-receiver: A frequency modulator transmitter and receiver modules were chosen over the AM (Amplitude Modulation) modules because of former's superior audio transmission quality. Also, FM approach was preferred for shorter transmission routes.

4.3.3 Tradeoffs: Several tradeoffs were encountered during the implementation of the product.

Battery Dependency: The device requires continuous connection to a battery to derive power/energy in order to continue to receive audio messages from the analytical engine.

Smartphone Dependency: The success of the product involves the users to carry an android smartphone, which shall relay the GPS coordinates using a background service. Although, an android smartphone is almost ubiquitous these days, still it is a tradeoff for non-android smartphone users.

GPS Sensitivity: Although, the GPS coordinates are as accurate as possible, there exists a certain sensitivity in closed range locations.

CHAPTER 5

SCHEDULE, TASKS AND MILESTONES

5.1 Schedule Tasks

The following scheduled was agreed upon during the planning stage of the project.

Sr. No.	Scheduled Task	Scheduled Date of Completion	Actual Date of Completion	Assignee	Reason for Delay (if any)
1	Product Brain Storming	20th December, 2014	20th December, 2014	Complete Team	
2	Product Feasibility Assessment	3rd January, 2015	3rd January, 2015	Complete Team	
3	Components Listing and Ordering	20th January, 2015	25th January, 2015	Mayank, Abhinav	Components were ordered online
4	Setting up Apache Spark for Analytic Engine	30th January, 2015	31st January, 2015	Mayank, Divyansh	Internet Connectivity Issue
5	Implementing Recommendation Logic using Scala and Spark	20th February, 2015	20th February, 2015	Divyansh	
6	Implementing the Android GPS Service	5th March, 2015	3rd March, 2015	Mayank	
7	Assembling the Arduino Kit and Testing	10th March, 2015	10th March, 2015	Abhinav	
8	Implementing server side script	25th March, 2015	27th March, 2015	Mayank, Abhinav	Issue while extracting

	for mapping recommendations and GPS Coordinates				coordinates from the GPS Service
9	Implementing the Audio Receiver on Arduino	5th April, 2015	5th April, 2015	Divyansh, Abhinav	
10	Interfacing the Hardware with Analytical Engine	14th April, 2015	14th April, 2015	Complete Team	
11	Implementing the Prototype	20th April, 2015		Complete Team	
12	Implementing and Packaging the Final Product	25th April, 2015		Complete Team	
13	Testing and Quality Assessment	30th April, 2015		Complete Team	

5.2 Significant Milestones

The following were the significant milestones recorded during the course of implementation of project:

Sr. No.	Milestone	Date of Completion
1	Recommendation Module Completion	20th February, 2015
2	Android GPS Service Implementation	3rd March, 2015
3	Implementation of Audio Receiver and Interfacing	14th April, 2015
4	Successful Implementation of Prototype	20th April, 2015
5	Testing and Project Completion	30th April, 2015

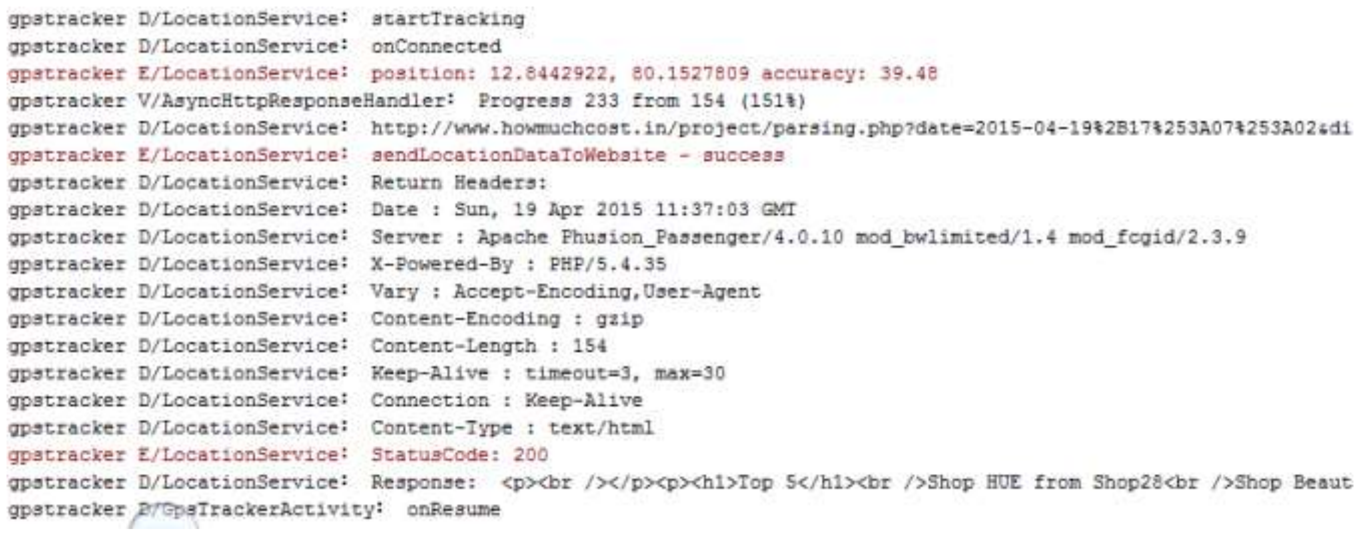
CHAPTER 6

PROJECT DEMONSTRATION

5.1 Schedule Tasks

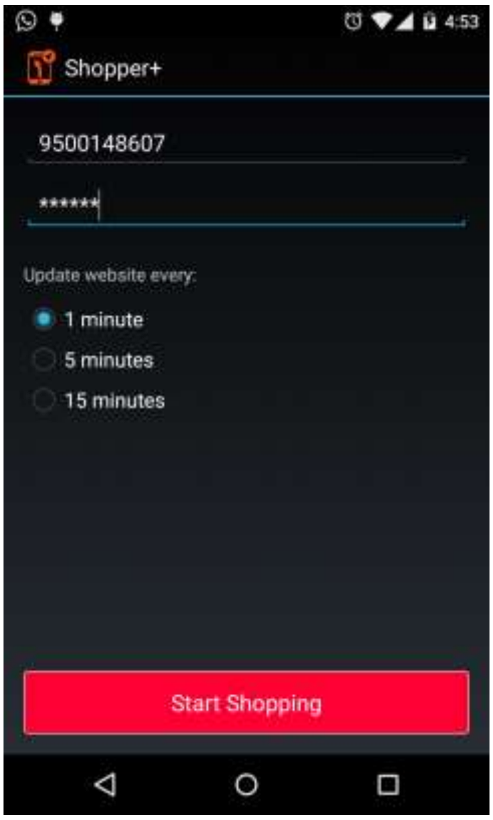
The following scheduled was agreed upon during the planning

Android:



```
gpstracker D/LocationService:  startTracking
gpstracker D/LocationService:  onConnected
gpstracker E/LocationService:  position: 12.8442922, 80.1527809 accuracy: 39.48
gpstracker V/AsyncHttpResponseHandler:  Progress 233 from 154 (151%)
gpstracker D/LocationService:  http://www.howmuchcost.in/project/parsing.php?date=2015-04-19%2B17%253A07%253A02&di
gpstracker E/LocationService:  sendLocationDataToWebsite - success
gpstracker D/LocationService:  Return Headers:
gpstracker D/LocationService:  Date : Sun, 19 Apr 2015 11:37:03 GMT
gpstracker D/LocationService:  Server : Apache Phusion_Passenger/4.0.10 mod_bwlimited/1.4 mod_fcgid/2.3.9
gpstracker D/LocationService:  X-Powered-By : PHP/5.4.35
gpstracker D/LocationService:  Vary : Accept-Encoding,User-Agent
gpstracker D/LocationService:  Content-Encoding : gzip
gpstracker D/LocationService:  Content-Length : 154
gpstracker D/LocationService:  Keep-Alive : timeout=3, max=30
gpstracker D/LocationService:  Connection : Keep-Alive
gpstracker D/LocationService:  Content-Type : text/html
gpstracker E/LocationService:  StatusCode: 200
gpstracker D/LocationService:  Response:  <p><br /></p><p><h1>Top 5</h1><br />Shop HUE from Shop28<br />Shop Beaut
gpstracker D/GpsTrackerActivity:  onResume
```

Figure 1. Log for Android application



Shopper+

9500148607

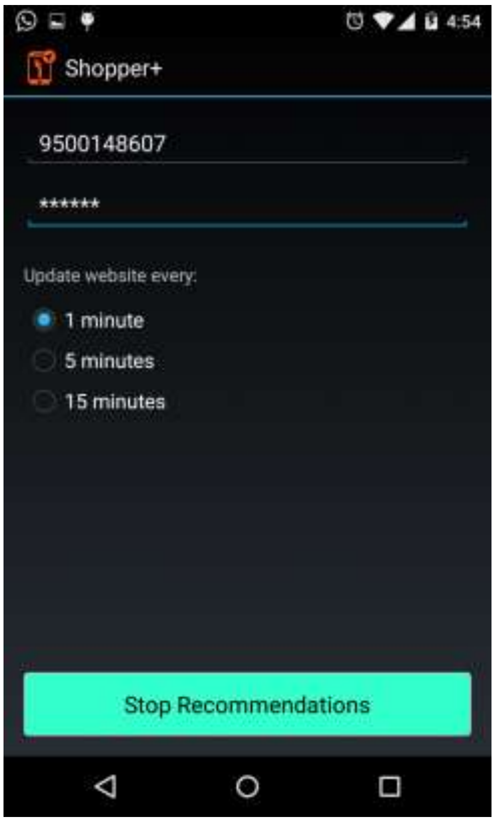
******

Update website every:

[✓ 1 minute]

5 minutes

15 minutes



Shopper+

9500148607

******

Update website every:

[✓ 1 minute]

5 minutes

15 minutes

Figure 2. Application Stop State **Figure 3. Application Start State**

CHAPTER 7

COST ANALYSIS

5.1 Schedule Tasks

The following scheduled was agreed upon during the planning

The **first level heading** (Title of the chapter like **INTRODUCTION**, **RELATED WORK, etc.,** should be in Times New Roman, 14 point scale, Bold, Centered, UPPERCASE with 1.5 line.

The second level headings(like sub headings in the chapters like **General, Motivation** should be in Times new roman, 13 point scale, Bold, Left Aligned, Title Case with 1.5 line spacing.

The third level headings like 1**.4.1 System Requirements** should be in Times new roman, 12 point scale.

The running characters in the paragraph should be in Times new roman, 12 point scale, Normal, **Justified** with 1.5 line spacing. (Sub- Sub Headings in the chapters should be in Times new roman, 12 point scale, Bold, Left Aligned after 1 tab position, Title Case with 1.5 line spacing. The running characters should start in the same line after leaving 5 spaces from the third level headings.