A Mobile-Cloud Framework for Context-aware and Mobility-driven Recommender for Smart Markets

By

Muhammad Aftab Khan

NUST201362748MSEECS61413F

Supervisor

Dr. Anis ur rehman

Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Computer Sciences (MS CS)

In

School of Electrical Engineering and Computer Science,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(May 2017)

Approval

It is certified that the contents and form of the thesis entitled "**A Mobile-Cloud Framework for Context-aware and Mobility-driven Recommender for Smart Markets**" submitted by **Muhammad Aftab Khan** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Anis ur rehman**

Signature: _____

Date: _____

Committee Member 1: **Dr. Asad Waqar Malik**

Signature: _____

Date: _____

Committee Member 2: **Dr. Imran Mehmood**

Signature: _____

Date: _____

Committee Member 3: **Dr. Mian Muhammad Hamayun**

Signature: _____

Date: _____

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Muhammad Aftab Khan**

Signature: _____

Dedication

This thesis is dedicated to my Mother for her love, endless support and encouragement. I would also like to dedicate my work to other members of my family, friends especially Syed Hassan Jaffar Kazmi, Professor Faiz Hussain and Dr. Aakash Ahmad for their endless support and encouragement. Without their efforts, support and encouragement, it would be a vision without action.

Acknowledgment

I am thankful to Almighty Allah SWT, the Most Gracious and the Most Merciful, I was able to complete my research work because of His infinite blessings. I am glad to express my sincere appreciation to my advisor Dr. Anis ur rehman for his continuous support and guidance throughout my thesis. Beside my advisor, I would like to express my deepest thanks to my master thesis committee members; Dr. Asad Waqar Malik, Dr. Imran Mehmood, and Dr. Mian Muhammad Hamayun for their guidance and encouragement. I am very grateful to Dr. Aakash Ahmed Abbasi who encouraged and guided me in each and every phase of research.

Abstract

Context and challenges: In recent years, smart city systems have emerged as solutions that transform the conventional cities and societies into information and communication (ICT) driven cities. Smart city systems offer improved and digitized urban services such as smart health and smart shopping to the stakeholders such as citizens, business entities and organizations. However, in mobile-cloud based smart city systems, challenges such as context-awareness, performance of cloud servers and the resource poverty of mobile devices must be addressed.

Solution and implications : In this thesis, we focus on the integration of the mobile computing and cloud computing technologies to develop a system that offers its users with context- aware and mobility-driven recommender system for smart markets. In the proposed solution, a mobile device represents the front-end (mobility and context-aware) interface for recommendations, while; cloud-based server represents the back-end (computation-intensive) processor of the system to enable digital match making between potential customer and business entities..

Evaluations and conclusions : We have utilized the ISO-IEC-9126 quality model to evaluate the accuracy and efficiency of the proposed recommender systems. Considering the resource poverty of a mobile device, the evaluation results suggest the accuracy, along with computational and energy efficiency of the recommender system. The future research is focused on the application of machine learning approaches for an intelligent, context-aware recommendation system.

Keywords: Mobile Cloud Computing, Mobile Recommender, Context-aware, Smart Cities.

Table of Contents

1 Introduction and Motivation **1**

1.1 Introduction 1

1.2 Problem Domain and Proposed Approach 4

1.2.1 Research Hypothesis 5

1.2.2 Research Questions 6

1.2.3 Overview of the Proposed Approach 6

1.3 Assumptions and Contributions of Research 8

1.4 Structure of the Thesis 10

1.5 Summary of Chapter 11

2 Literature Review **12**

2.1 Chapter Overview 12

2.2 Background 12

2.3 Recommendation for Tourism and Leisure Activities 14

2.3.1 Mobile Recommender for Places of Interest Recommendation 14

2.3.2 Knowledge-based Recommender For Movie Recommendation 15

2.3.3 Scalable Mobile Recommender for Leisure Time Activities Recommendation 17
2.3.4 Cloud-based Recommender for Media Items Recommendation 18
2.3.5 3D Context-aware Mobile Recommender 20
2.3.6 Cloud-based Recommender for Group Recommendation 22
2.4 Recommendations in the Context of e-Commerce 23
2.4.1 Mobile Recommender to Assist in Shopping 23
2.5 Recommendations for e-Health 25
2.5.1 Cloud-based Health Information Mobile Recommender 25
2.6 Application of Recommender systems to Generic Domains . . 27
2.6.1 General Recommendation Framework for Mobile Plateforms 27
2.6.2 Recommendation Framework For Indoor and Outdoor settings 28
2.6.3 Privacy Preserving Mobile Recommender System 29
2.6.4 Service Oriented Recommender System 30
2.7 Conclusions and Comparative Summary 31

3 Proposed Architecture of Mobile-Cloud Recommender Framework 33

3.1 Chapter Overview 33

3.2 Architectural Overview of the Framework 33
3.3 Mobile Computing layer 35
3.3.1 Ionic Plateform 36
3.4 Cloud Computing layer 37
3.4.1 Cloud Processing 39
3.4.2 Cloud Storage 40
3.5 Summary of Chapter 40

4 Implementation of Smart Market Recommender System 42
4.1 Chapter Overview 42
4.2 Implementing Context-awareness and Recommentation Processing 43
4.2.1 Assumptions for System Implementation 44
4.3 Context Aware Mobile Layer - Online Processing for Recommendations 45
4.3.1 Algorithm(s) for Mobile Context Aware Recommendations 47
4.4 Processing Based Cloud Layer - Offline Processing for Recommendations 51
4.4.1 Algorithm(s) for Cloud Based Processing of Recommendations 54
4.5 Summary of Chapter 57

5 Evaluation of Recommender System 58
5.1 Overview of the Chapter 58
5.2 Context, Objectives and Methodology for Evaluation 59

5.2.1 Context and Methodology of Evaluation 59
5.2.2 Objectives of the Evaluation 59
5.3 ISO/IEC 9126 Model for Evaluation 60
5.3.1 Criteria for Evaluation 60
5.3.2 Data Set and Use Cases for Evaluation 61
5.4 Evaluating System Accuracy for Recommendations 67
5.4.1 Accuracy of Computing Recommendations 68
5.5 Evaluating System Efficiency for Recommendations 71
5.6 Validity Threats and Future Evaluations 75
5.7 Chapter Summary 76

6 Conclusions and Future Work 77
6.1 Overview of the Chapter 77
6.2 Summary of Research Contributions 77
6.3 Dimensions of Future Research 78

A Source Code 88
A.1 Source Code Offline Recommendation 88
A.2 Source Code Online Recommendation 97

B DataSet Detail 103
B.1 Sample of Superstore Sales Data Set 105
B.2 Source Code to Process Dataset 107
B.3 Sample Of Processed Dataset 113

B.3.1 Sample Records from Users Table 113

B.3.2 Sample Records from Products Table 116

B.3.3 Sample Records from Ratings Table 118

List of Figures

1.1 Solution Overview as a Layered Representation of Mobile-Cloud Recommendation System 7
1.2 An Overview of the Thesis Structure 11

3.1 An Architectural Overview of Proposed Solution 34

4.1 Overview of Online and Offline Recommendation's Generation. 47

5.1 CPU and Memory Consumption in Foreground 72
5.2 CPU and Memory Consumption in Background 73
5.3 Battery Consumption in Background 74
5.4 Battery Consumption in Foreground 75

B.1 Overview Of Dataset Processing 104

List of Tables

2.1 A Comparison Summary of State-of-the-art-Research 32

4.1 Parameters for Online Recommendation Generation 48
4.2 Utility Methods for Online Recommendation Generation . . . 49
4.3 Parameters for Offline Recommendation Computation 54
4.4 Utility Methods for Offline Recommendation Computation . . 55

5.1 Use Case 01 for Accuracy Evaluation 63
5.2 Use Case 02 for Accuracy Evaluation 64
5.3 Use Case 01 for Efficiency Evaluation 65
5.4 Use Case 02 for Efficiency Evaluation 66
5.5 Precision and Recall for newly Registered User 71
5.6 Precision and Recall for Existing User 71

B.1 Sample Sales Data Set 106

List of Algorithms

1 recommend-products(activeuseID , currentLocation , activeUser-Preferences) to generate Recommendation 50
2 similiarProducts(User-Product rating matrix) 56
3 similiarUsers(User-Product rating matrix) 56

Abbreviations

Abbreviations	Description
CARS	Context Aware Recommender Systems
MCC	Mobile Cloud Computing
ICT	Information and Communication Technology
ISO	International Organization for Standardization
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
CPU	Central Processing Unit
RAM	Random Access Memory
POIs	Point Of Interests
GUI	Graphical User Interface
CF	Collabrative Filtering
CBF	Content Based Filtering
REST	Representational State Transfer
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
SaaS	Software as a Service

Glossary of Definitions

- Context-aware Recommender System is defined as software system that generate meaningful recommendations of interests to users according to her context and empower them with decision support.

- Mobile Cloud Computing is defined as unfication of mobile computing with cloud computing in such a way that storage and processing related tasks are peformed over cloud.

- Information and Communications Technology (ICT) is defined as an infrastructure that comprises any communication application or device, including: mobile devices, computer as well as network software and hardware , radio, TV, satellite systems and so on, as well as the numerous applications and services related to them.

- Smart Market is defined as electronic market place that exploits ICT technologies and infrastructure to enhance digitized commerce.

- Context is defined as a piece of information that help to describe the current situation of the end users or devices

Chapter 1

Introduction and Motivation

1.1 Introduction

Smart city systems have emerged as a recent trend that exploit the information and communication (ICT) technologies to offer improved urban services to individuals and collectively improving the life style of societies [1]. Specifically, a smart city system offers technology-driven urban services (such as smart transportation, smart health, smart shopping) and empowers organizations or institutes to offer such services efficiently and cost-effectively [1]. In the context of smart city systems, mobile computing has emerged as a pervasive technology that enables users - exploiting mobility and context-awareness - to perform a variety of tasks including portable computation and location-aware communication [2]. However, due to a portable nature, mobile devices are considered as resource constrained computers that lack the performance of computation and memory-intensive tasks. Although the size of mobile device is small and it has limited capabilities but on the other

hand it has a huge advantage of mobility [3]. In contrast to mobile computing, the cloud computing model exploits the pay-per-use software service model to provide virtually unlimited processing and storage resources [4]. Therefore, the unification of mobile-cloud computing can benefit from the mobility and context awareness (of mobile computing) and the computation and storage services (of cloud computing) to provide systems that are portable, yet resource sufficient [5]. One of the main challenges for managing and exploiting context-aware recommendations is to identify the contextual factors (e.g., location, preferences, mood) that influence the decisions and actions of people or entities in a smart city context [6]. Context is actually a piece of information that help to describe the current situation of the end users or devices [7]. It is clear as crystal that contextual information of user is vital to suggest user accurate services in pervasive networks [8]. Mobile computing is enabled with a combination of mobile device (hardware), mobile apps (software) and remote server (network connectivity) to offer location/context-sensitive and mobility-driven computational services to its users [9]. Specifically, a mobile device acts as a back-bone of mobile computing and represents any (handheld) equipment or machine that allows its user to perform computation, information sharing and other relevant activities. For example, in mobile computing the embedded hardware resources of a mobile device (e.g.; GPS, Wi-Fi sensors) that can be combined with available software applications on remote servers (e.g.; location tracking, pulse monitoring) to offer context and location-aware recommendations to the users [10]. Mobility is regarded among the central features of mobile computing that empowers its user to perform complex computation on

the go. However, mobility also enforces (hardware-level) resource constraints such as limited computation, storage and battery available on a mobile device. There is a need for solutions that maintain a balance between mobility and resource availability in the context of mobile computing [11]. Cloud computing can offer the entities or organizations to off-load or deploy their (on premise) software systems, computation or storage to servers by means of cloud-based services. For example, the research in [12] highlights an approach known as cyber-foraging that off-loads the computationally-intensive tasks and storage from a mobile-device to (cloud-based) servers in order to enhance computation and energy efficiency of mobile devices. Therefore, in the context of (resource-constrained) mobile-computing, (resource-sufficient) cloud computing can be viewed as an opportunistic model that allows mobile devices to off-load data and computation over cloud-servers [13]. Context-aware computing is a paradigm of mobile computing in which applications can detect information about user context and make use of it. Contextual information include weather, time of day, user location, neighboring users as well as devices and activity of user. Recommender Systems represent a class of software systems that generate meaningful recommendations of interests to their users and empower them with decision support. More precisely, due to huge volume of information it is difficult for people to find related and helpful information for them. Recommender systems help people to find items that are hard to find people by themselves [14]. Hence, Context-aware Recommender Systems provide dynamic adaptive recommendations to users based on contextual information of the user. In smart city context, smart markets refer to electronic market places that exploits ICT technologies and

infrastructure to enhance digitized commerce. Recommendation and matchmaking between the potential customers and business entities determined by relevancy of contextual information giving rise to the concept of Smart Markets. Smart Markets assist in personalized content delivery and advertisement while minimizing the irrelevant mass publicity. Recently, much research and development is being carried on theories as well as solutions that support Context-aware Recommender Systems but existing mobile recommender systems fall short of context-aware recommendations in a smart market domain. Mobile Cloud Computing (MCC) as state-of-the-art for mobile computing needs to be exploited for context-aware recommendations in the context of for smart markets.

1.2 Problem Domain and Proposed Approach

In the past, the electronic commerce (e-commerce) systems have proven to be influential to offer products and services in international marketplaces. In modern day world, business systems heavily rely upon reaching their potential customers and offering recommendation systems. Now with mobile commerce (m-commerce), the use of context-information (such as age, gender, preferences, location) for customer recommendation has gained a significant attention. Considering a wide-spread adoption of the mobile and cloud computing, there is still a lack of solutions that facilitate its user with recommendation of their preferences primarily based on their local context. The proposed project aims to address the challenges of mobile-recommenders and empowering its user with decision making to recommend the events or

the places of their interests [15]. Specifically, the solution offers a location-aware and context-sensitive mobile application that will allow its user to get the most relevant recommendations about the items of their interests such as shopping deals, discount on events and food, sharing a taxi-ride based on common location along with many other possibilities along the same direction. In principle, the solution unifies the concept of mobile computing and cloud computing. The solution frameworks will allow markets, business and transaction-driven entities to communicate with their potential customers in a smart way – product recommendations to the customer based on their location and preferences. The primary challenge to generate context-aware recommendations is identification of the contextual elements such as users location, preferences from different sources. Another research challenge for recommendation systems is to yield recommendations in a real-time fashion for a given user from mega and diverse dataset of persons past preferences. It is vital to mention that recommendation algorithms are often highly computational so deploying recommender server upon cloud for computing efficiency is another challenge [15]. We have discussed the central hypothesis, research question and proposed solution in the sections below.

1.2.1 Research Hypothesis

We outline the central research hypothesis as:

The unification of mobile computing (as frontend layer) with cloud computing (as backend layer) can enable context-aware recommendations for efficient and effective matchmaking between business entities (providers) and

customers (consumers).

1.2.2 Research Questions

Analysis of the hypothesis suggests that the research problem that we aim to address is, how to empower the customers that can utilize their mobile devices to obtain context-aware and mobility-driven recommendations from business entities, while also ensuring computational and energy efficiency in resource-constrained mobile computing environment.

- **Research Question 1 -** *How to identify user contextual information to provide context-aware recommendations that allows potential match-making between customer and business entities?*

  Objective : The objective of this question is to find the correct tools and techniques to get the user context information and suggest best possible items relevant to that information.

- **Research Question 2 -** *How to ensure computational and energy efficiency in resource-constrained mobile computing environment?*

  Objective : The objective of this research question is to use proper algorithms and techniques for sake of computing and energy efficiency.

1.2.3 Overview of the Proposed Approach

The proposed solution provides a bridge between the users (or potential customers) and the business entities. Specifically, the businesses can provide all the relevant information (e.g.; product details, discounts, food item deals, taxi sharing) to our system by means of simple and touch-sensitive interfaces.

An overview of our proposed solution as a layered system comprising of the (i) Mobile Computing as the front end and (ii) Cloud Computing as the back end layer is provided in Figure 1.1. Specifically, mobile computing layer by means of portable and context-aware devices allows the user and business entities referred to as potential customers and market place in Figure 1.1 to specify their preferences in terms of identifying and enquiring consumer services and goods. Mobile computing layers allows sensing the context in terms of users location and proximity, along with business preferences. All the contextual data is off-loaded to the cloud computing layers. More precisely, the cloud computing layer exploits the storage and processing capabilities to compute recommendations that are sent back to the mobile. The integration between the mobile and cloud computing is enabled through network connectivity as illustrated in Figure 1.1.

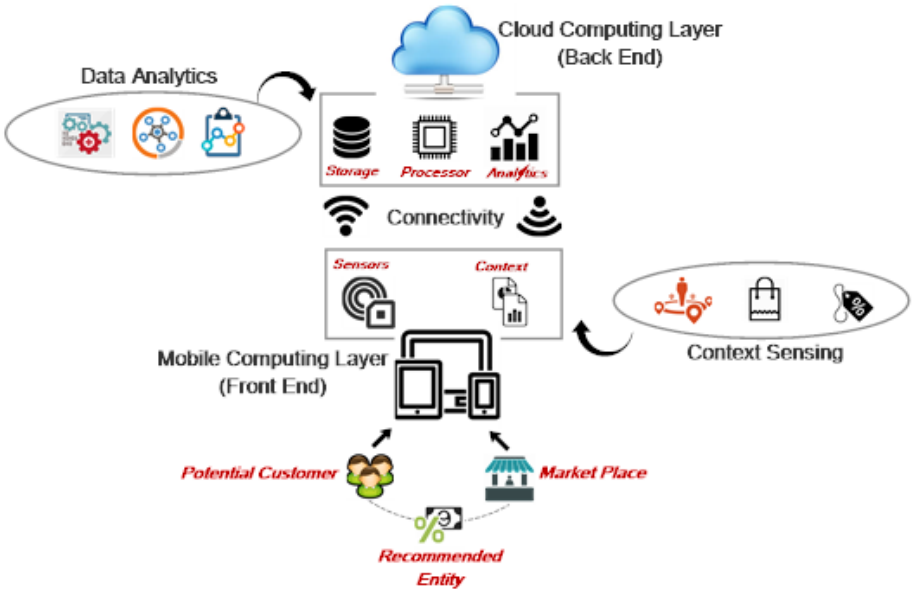

Figure 1.1: Solution Overview as a Layered Representation of Mobile-Cloud Recommendation System

Mobile computing as a pervasive technology empowers its users to exploit mobility and context-aware computations. Mobile computing has some limitations such as computation and storage constraints as well as power and energy constraints. Here comes the concept of cloud computing. Cloud computing exploits the pay-per-use software services to provide virtually unlimited processing and storage resources. The key advantage of cloud computing is elasticity which means that computational resources can be automatically scaled up and down on demand by the provider of cloud service. Another noteworthy benefit of cloud computing is that cloud architecture can be exploited to enable dynamic service composition [16]. Our proposed solution takes advantage of mobile cloud computing architecture and utilizes elastic processor and storage of cloud architecture and offers its consumers context aware and mobility driven recommendations.

1.3 Assumptions and Contributions of Research

We summarize the main contributions and the assumptions for this work. Our research work consist of two distinct fields: i) Mobile computing ii) Cloud computing. On the basis of solution overview, the proposed contributions and relevant assumptions are as follows.

- **Contribution 1 - A Framework for Context-aware Mobile Recommender System:** For the sake of computing and power efficiency we exploit the cloud computing idea. Overall, the solution combines the concept of mobile computing and cloud computing. So the main contribution is discovery of a framework that exploits state-of-the-

art of mobile computing namely mobile cloud computing for context-aware mobile recommender system.

- **Contribution 2 - An Architectural Framework in Smart Market Domain:** The framework permits business, markets and transaction-based entities to communicate with their potential consumers in a smart manner and recommend on the basis of their location as well as preferences. Another contribution is design of an architectural framework for a scalable match making between customer and enterprises in the context of smart market.

- **Assumption 1 - Network Availability and Context-aware Devices:** All data relevant to user is store on database installed upon cloud. User device sends its context to server deployed upon cloud and recommendations are generated.

Therefore the primary requirement or assumption is the availability of network connectivity and the devices that enable context-aware computing.

1.4 Structure of the Thesis

The rest of this report is organized as shown in Figure 1.2.

- **Chapter 1:** familiarizes the reader with the summary of the selected research topic along with the research questions as well as research challenge and the proposed solution.
- **Chapter 2:** presents the related work conducted so far in selected field in a Well-organized manner.
- **Chapter 3:** discusses the proposed solution of the chosen research challenge as well as its architecture.
- **Chapter 4:** sheds the light on the framework along with implementation of the solution proposed.
- **Chapter 5:** focuses on the methodologies followed to assess and evaluate the proposed solution .
- **Chapter 6:** concludes the research work by presenting the well-defined conclusion. This chapter will also assist researchers to conduct the future research in the domain of context aware mobile computing for smart markets.

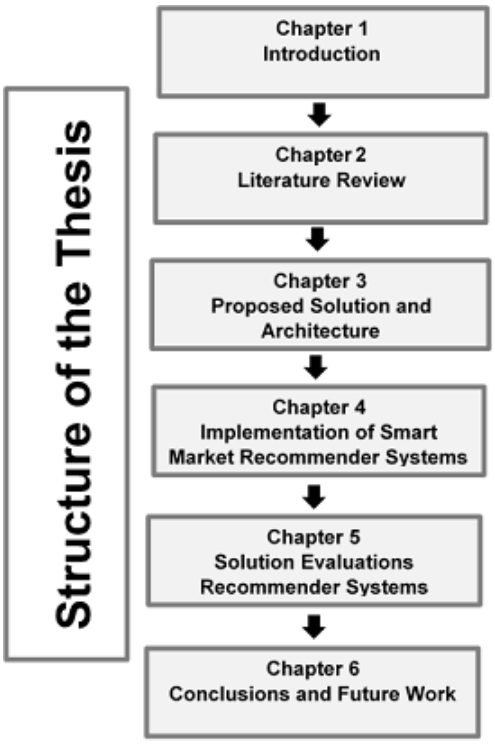

Figure 1.2: An Overview of the Thesis Structure

1.5 Summary of Chapter

The main objective of Chapter 1 was to provide the motivation behind this research work. Based on a brief overview of existing recommender systems , we defined research hypothesis and identified research questions as well. It is clear as crystal that research questions play an important role to illuminate the requirements for the solution regarding assumptions and contributions of research. A summary about objectives as well as the outcome of each and every chapter in this thesis reports is shown in Figure 1.2.

Chapter 2

Literature Review

2.1 Chapter Overview

In this chapter, we discuss the relevant related research to highlight the existing work. The discussion of related research also help us to justify the proposed solution later in this report,in the context of modern day recommender systems. After presenting the state-of-the-art on recommender systems, we provide comparative analysis of all the existing solutions to explicitly justify the contributions of our proposed solution.

2.2 Background

We can classify recommender systems into five distinct categories on the basis of technique they used to measure the utility of the items for the user of system [15], i.e.; based on the recommendation technique(s) [17]. The types of recommender systems include:

1. **Content-based recommender systems**– these are the systems that recommend items to user according to user preferences and past history.

2. **Collaborative filtering recommender systems**– these are the systems that recommend the items to user according to users with similar preferences rated highly in the past.

3. **Demographic recommender systems**– these are the systems that recommend items to users on the basis of their personal profile of demography. .

4. **Knowledge-based recommender systems**– these systems recommend items according to either inferences regarding user tastes or particular domain knowledge about how items fits better according to preferences of user.

5. **Hybrid recommender systems**– these are the systems that are based on the union of the above mentioned approaches and techniques.

In addition to the types of recommender systems above, we also discuss the three major categories or the real world domains where the types of recommender systems have been applied. The real world domains include

(A) Recommendation for Tourism and Leisure Activities

(B) Recommendations in the Context of e-Commerce

(C) Recommendations for e-Health

(D) Application of Recommender systems to Generic Domains

In the following, we discuss in details the application of various types of recommender systems to real world domains.

2.3 Recommendation for Tourism and Leisure Activities

A Context aware Recommendation System is a software which proposes items to user according to his/her taste and preferences as well as contextual factors (e.g., location, mood, weather).

2.3.1 Mobile Recommender for Places of Interest Recommendation

During current years, many researches have been conducted in different domains such as tourism, leisure, ecommerce and mobile-commerce. For example [18] has developed a context aware mobile application (STS) that suggest users places of their interest by using their mood, weather conditions and personality traits of user. They used Five Factor Model [19] along with user past rating to discover user personality traits. The personality traits have used in active learning module to get user recommendation about the places they have already visited. Weather information has been retrieved with the help of a web service. Context aware matrix factorization model for rating prediction in conjunction with set of known user attributes has been adopted to recommend places of interest. The STS implementation has one client application and one server side application. Client application has been de-

veloped using android technology. While server side application consists of PostgreSQL database and Apache Tomcat server. The functionality of server is exposed to client application with the help of RESTful web service that sends and received JSON objects which provide recommendations. STS is able to recommend places of interest according to the current context of user, for example, if weather is bad than indoor POIs such as castles, museums, churches etc. are recommended and if weather is good than outdoor POIs such as lakes, mountain hikes, scenic walks etc. are recommended to applications user. STS application does not provide push based recommendation to user and user has to explicitly request for recommendation.

2.3.2 Knowledge-based Recommender For Movie Recommendation

Currently, recommender systems are making use of semantic web technologies to cope the challenges of data sources diversity and information overload. For example [15] presented a recommendation system named as RecomMetz in the domain of movie show times. Three different types of contextual factors are studied: *(a) location (b) crowd and (c) time.* The proposed architecture of RecomMetz is consist of six main modules, namely user interface, user check-in Subsystem, information collector, data repository, context-aware subsystem and recommendation engine. The active user requests for a recommendation and interface sends the users position and time information to the user profiler. The user profiler retrieves the users preferences from

the user database and builds a user profile. MySQL DBMS has been used to store user preference and past visiting history of users. The time-aware service retrieves the user profile as well as all the movies and movie theaters from the domain database and generates a list of movies currently available in movie theaters. The location-aware service retrieves the previously generated lists and previously built user profile. It calls the information collector to calculate the distances between the users current position (origin) and each one of the movie theaters (destinations) as well as the times required to get to each one of the destinations. The information collector sends a series of HTTP requests to the google distance matrix API. The location-aware service filters the list of movie theaters previously generated by the time-aware service. It retrieves all the movie showtimes from the domain database and generates a list of the movie showtimes that the user is likely to attend. The knowledge-based recommender retrieves the lists of movies, movie theaters and showtimes. It computes the semantic similarities between all the movies, movie theater and showtime triples resulting from the previous filtering operations and send user recommendations. An already existed ontology known as Movie Ontology which is maintained at the University of Zurich by its informatics department was exploited as the ground work for development of the domain ontology in this research. The most interesting part of the story is that in oppose to other research work conducted upon context-sensitive systems, In RecomMetz the user context is modeled as a two-part model consist of group context as well as individual context. So in this research work group context represented crowd information and it is modeled in the domain ontology. On the other hand, Individual context represented to the

other type of context information consist of location as well as time and this context is modeled in each and every user profile. Evaluation was carried on as a user interaction in a real-life scenario. In short, eight students from the Murcia University (Murcia, Spain) have used RecomMetz for the sake of precision, recall and f-measure measurements of application. The experimental evaluation demonstrate that system functions effectively in both a cold-start and a no cold-start-settings. RecomMetz supports by multi-platform mobile user.

2.3.3 Scalable Mobile Recommender for Leisure Time Activities Recommendation

A primary challenge in developing multi-user mobile information systems for real-life deployment lies in the building of systems which are scalable. By keeping scalability issue in mind [9] has designed a high-performance mobile recommender system (Magitti) which is scalable to many users operating at the same time. They proposed a technique for recommending leisure time activities based on what time of week it is and what venues are close to the user location by combining multiple recommendation patterns using predefined rules. Conventional client server approach has been used with heavy tasks (other than UI) executed on the server for the sake of minimizing the power consumption of device.

The GUI of the client application has been developed using C#. The server side application runs upon Apache Tomcat server. The client side application communicate to server side application over HTTP. The location

context of user is retrieve via an embedded GPS within client application. JSON has been used to send and receive messages to Server side application.

For Recommendation purpose a combination of CF with other techniques has been used. Spatialized data storage technique has been utilized to assure that computation is performed only on items that are within a defined geographic location. There are some possible negative aspects of spatial partitioning model in which the most important is there may be a specific geographic boundary which may have only a few items to send as recommendation to end user. To resolve this issue, they have implemented a technique which automatically expand the query range when there are too few items within the result set.

A set of testing applications a simulator (to simulate the users motion), a visualizer (to Monitor the state of server), and a re-player application (to replay the users motion of) have been used to evaluate solution. *Magitti outperforms when there are 400 to 500 simultaneous user per server instance. But when there are more than 500 users at the same time per server instance than the response time is degraded.*

2.3.4 Cloud-based Recommender for Media Items Recommendation

Due to the limited power and computational resources of mobile devices, cloud services turn into a great alternative to software configured on mobiles. To exploit cloud architecture [20] have proposed context aware recommendation techniques to recommend relevant cloud based media particulars to

mobile users. The presented architecture has three core services. The first service i.e.; context recognition service is responsible for monitoring, learning and predicting user context. To retrieve user context Wi-Fi, GPS, accelerometer, rotation as well as orientation vector sensors have been used. Nearest Neighbor (KNN) algorithm has been used as predictive model to predict the location and activity of user. context recognition service send users context related details to the cloud based service known as contextual user profile service which learns the media usage preferences of the user and associates them with user current and past context. It also stores the user consumption and preferences history in a database.

The third and last service is recommendation service and its responsibility is to crawl and fetch useful information and recommend media content to user according to user's taste. Hybrid context aware recommendation technique has been used to generate recommendation list. The recommendation technique is based upon mixture of context aware collaborative filtering (CF) and content based filtering (CBF) technique. The client application has been developed using android platform. Java technologies have been used to build users contextual profile. Oracle web logic server has been used for deployment of server side application. MySQL database has been used to save context history and other related data of the user. Precision evaluation metric has been used in the research and the results shown that context aware hybrid recommendation process performs much better than the traditional hybrid recommendation technique.

Apart from recommendation techniques, success of the systems in the domain of mobile tourism heavily rely on useable and intuitive user interface.

Until now, due to size limitation of mobile devices , user interaction is particularly hard and frustrating. Recommendation systems present items within a textual format and interface or, at max, within an abstract two dimensional map. With this into account,[21] upgraded a web based recommender system named REJA to a 3D mobile based context aware recommendation system. REJA addressed different drawbacks regarding mobile recommender systems and solved interface issues. First extension in the system is to include location of the user as context. As a second extension an interface has been provided for the sake of geo-referencing recommendation of places using three-dimensional view.

2.3.5 3D Context-aware Mobile Recommender

The proposed system has adopted hybrid approach consisting of knowledge based and collaborative filtering techniques. Location aware module has been introduced in the proposed solution for the sake of information pre-filtering. The process of contextual pre-filtering is applied to minimize items'count for the recommendation according to the location of user. These items are utilized by proposed system to generate a top-N list of item which has potentially higher match to users interest. Location aware module is used in both knowledge based and collaborative filtering techniques. Proposed collaborative filtering technique contains three phases.

First of all registered users are grouped according to their preferences and taste. In second step prediction is generated for every item which user has not viewed yet. Finally the top-N items with high prediction are recommended

to user. To handle cold start problem knowledge base recommendation technique is used by proposed system. Knowledge base filtering technique used multiple steps for the sake of item recommendation. First of all user set her preferences and set of features for the items she is interested in. These preference are used by the system to generate users profile. Than system compares items features and user profile using different similarities measures. Finally top-N items similar to users preferences are recommended to user. In the final step context post filtering is applied to re-ranks the list of top-N items recommended by the previous phase on the basis of distance. A three tier client-server architecture approach has been used to implement proposed solution. These three tiers are the mobile client application, the GIS server and the recommender server. The mobile client application is responsible to download and render 3D maps over cellular network. It also keep track of user location and speed via GPS/compass. Another responsibility of it is to request recommendation from server side. To make the communication efficient between mobile application and GIS server binary request-response protocol has been used [22]. On the server side two applications namely GIS server and recommender server have been used. The task of GIS server is to communicate with mobile application and provide data on users demand. It is also responsible to store terrain dataset. While the recommender system applied above mentioned recommendation techniques to generate recommendation. The mobile application sends request for recommendation along with context information to GIS server. The GIS server forward the request to recommendation server. The recommender system generate recommendations by adopting above mention hybrid approach and sent the list of recommended

items to GIS server. Eventually the resultant list is sent to client via GIS server which utilize it to populate 3D environment.

2.3.6 Cloud-based Recommender for Group Recommendation

Current approaches do not provide an optimum solution for the problem of group recommendation as well as cold start and data sparseness problems. To overcome this [23] have implemented cloud based solution (OmniSuggest) for the problem of venue recommendation in the domain of social networks for a single user and/or a group of friends. OmniSuggest uses mixture of ant colony algorithms with social filtering technique consist of both model and memory based CF to retrieve most favorable location recommendations addressing real-time issues such as traffic and weather conditions. There are three major component of OmniSuggest. One component is user profile which is data repository to save user identity, name and identification of venue, location of venue, timestamp at which user carry out check-in at a venue. The other component is top-K users and venues which is distribution and parallel execution of processing tasks on cloud framework as each place has local sets of users and venues. The final component is recommendation module in which ant colony algorithm and collaborative filtering (CF) have been applied to generate best possible solution in the form of venues that best match an active users preference. OmniSuggest framework has two main modules in terms of functionality: *1) an offline processing module and 2) an online recommendation module.* The offline processing component

executes recurring jobs to pre-process data related to check-ins. Data pre-processing consist two stages: a) popularity ranking of venues as well as venues, and b) creation of similarity graph among popular users. On the other hand responsibility of online recommendation module is to generate the recommendations for a group consist of friends or a single user. To minimize computational cost and to ensure scalability, OmniSuggest leverage cloud architecture. Real dataset gathered from Foursquare has been used to conduct experiments. *The Dataset contains 425,680 check-in given by 49,027 users for 206,416 venues located in the city of New York and 327,431 check-in provided by 38,134 end users in Los Angeles. The users tips history is break down into two parts: 1) testing set (consist of 20 percent of the data), 2) training set (consist of 80 percent of the data). Three performance metrics precision, recall and f-measure have been used for the evaluation of application.*

2.4 Recommendations in the Context of e-Commerce

2.4.1 Mobile Recommender to Assist in Shopping

Current recommendation techniques cannot fully applied in environment of mobile shopping. To cope this [24] proposed a location-aware recommender system PR (personal recommender) that fulfill customers' shopping demands with location-based seller offers and publicities. They adopted a decentralized approach instead of centralized approach. On customer side there are

two component a) web browser b) location manager. When customer sends request for recommendation, request is sent to server along with customer current location. Current location of the customer is retrieved with the help of GPS technology. Server side recommendation consist of two components. *a) offline subsystem b) online subsystem.* Offline system maintains database for customer history and derived customer's profile. The online system contains database having customer profile and vendors web pages information and recommend customer pages according to customers interest.

The data about which web pages have been explored by the end user is stored in the database called WEB ACCESS. Once the database is populated with the customer web pages visit history. Structured data is obtained from unstructured text of web pages by using Information extraction technique. In this technique raw pages are parsed, Prepositions and punctuation is removed and stemming words are grouped into general term. For every single web page term vector has been created. Bag-of-words model has been exploited to build term vector. In the term vector each cell represents the occurrences of the each and every term in the web page. Similarly the end user profile of interest is represented as term of vector.

With the help of customer past history which is stored in the form of customer preferences, similarity is estimated with any new web page by the vendor. The distance between the vendor and customer is also important parameter used by the proposed system. The PR system yields a list of recommendation that contains the top-N webpages for the requesting customer.

2.5 Recommendations for e-Health

Services related to health information technology can be easily run over web due to current technological advancement in the field of cloud computing and strong infrastructure of wireless communication and sensor networks. Multiple organizations have provided on-line information regarding medical available for public use. It has been observed that people make use of this information for personal health care management or patient-specific decision making [25].

2.5.1 Cloud-based Health Information Mobile Recommender

As the information pertinent to the patients concern is generally distributed across a huge number of different web sites so it is hard for patients to explore authentic healthcare information from large data volume. It has been found that young people preferred to use mobile devices to download or browse health information [26]. Taking this into account [27] proposed a framework to develop a recommendation service that facilitate user to get relevant health information on mobile devices. The proposed framework has five components: a database set up in the cloud environment; a module to query health information; web based user interface; a recommender based upon collaborative filtering technique and physiological indicators-based recommender.

Server is configures upon VMware workstation for the sake of high availability and computing efficiency. Server has knowledge base which stores a

large volume of health knowledge provided by an online health and medical information portal in Taiwan, a database to store user profile , a database to store physiological indicators and a matrix of rough-set-based prediction. Collaborative recommender uses user Profile and rough-set-based prediction to compute recommendations.

Database for physiological indicator store vital signs such as pulse, body temperature, blood pressure, respiration and blood oxygen. Physiological indicator based recommender uses different context aware devices for physiological indicators measurement.

Collaborative recommender is primarily based on the users tastes through the collaborative filtering techniques to recommend information regarding health to users. Proposed framework applied the FRSA algorithm proposed by [28] to compute and recommend health information matching the users tastes and preferences.

The physiological indicator-based recommender is primarily based on users physiological indicators for recommendation of health information. This recommender depends heavily upon physiological data measuring devices such as pulse and blood oxygen meters, heart rate monitors ,blood pressure monitors, weight meters and so on. These devices have been embedded sensors to record and transmit physiological data signals to physiological indicator database. These signals are transmitted with the help of wireless sensor networks to the database setup in the cloud and the system will recommend health information matching users physiological indicators based on inference rules.

To increase strength of signal transmission of physiological indicators eval-

uating device sensors and to decrease the costs of power consumption, a hybrid predictive model has been used, which make use of the Markov chain and Grey theory to predict the path of moving object. The system learns the schedules of waking and sleeping for every single sensor node. This will reduce the cost which originates from errors tracking and prolong the lifetime of network. Proposed framework has web based user through which user can obtain health recommendations result from collaborative-filtering based recommender and the physiological indicator-based recommender.

2.6 Application of Recommender systems to Generic Domains

2.6.1 General Recommendation Framework for Mobile Plateforms

Although traditional frameworks for the implementation of recommendation engines (e.g., content and collaborative filtering) can handle large sets of static user and item detail, they lack a general solution for using context related information. To handle this problem [29] has designed and implemented a general framework that support implementation of context-aware recommendation engines, particularly for mobile platforms. A combination of existing traditional collaborative filtering techniques and context data relevant to user has been used to implement proposed framework. The proposed solution is domain free and can been used in different application areas. The general architecture that implements the proposed approach follows client-

server model. The server implements all modules required for data access as well as information retrieval subsystem and recommendation module. The server database consist of product and user profiles, a matrix of provided ratings and additional semantic information. RDF and SPARQL web query languages can be used to access and retrieve additional semantic information from external sources such as facebook, twitter etc. for the sake of better measure of item-base similarity. The responsibility of recommendation module is to send ratings of resulting product to client. Rest service architecture has been applied to implement communication between client and server. The proposed framework fulfill all five requirements for the implementation of context aware recommendation i.e.; *a) Flexible and dynamic customization b) Temporal aspect c) Transparency d) Performance e) Quality.*

2.6.2 Recommendation Framework For Indoor and Outdoor settings

To support both push-based and pull-based recommendations [30] has proposed a domain free and generic context-aware recommendation framework to be used specifically in the domain of mobile computing. Proposed framework can be utilized for both indoor and outdoor settings. It is a three tier architecture in which the first layer is view layer and is used to take user queries and show different recommendation according to user preferences and context. The second layer has multiple modules such as sentiment analyzer, push based and pull based recommenders, user profile and contexts etc. for accurate recommendations to user. Third layer (data layer) provides access

to data relevant for the recommendation process, such as maps of the environment or external data sources. Architecture is based upon theories and evaluated against case studies.

2.6.3 Privacy Preserving Mobile Recommender System

Location aware recommender highlight serious privacy issues, such as the right of users to know how, when and under which circumstances their location as well as identity can be accessible to other users or services. To fix this genuine issue a privacy preserving mobile recommender system called PRECISE [10] has developed using cloud architecture.Context aware services have been deployed as SaaS upon cloud. Storage and computation is performed upon IaaS for the sake of efficiency. The main component is PRECISE to which the end user interact. It is middleware and deployed as PaaS. PRECISE consist of different components to complete all tasks. The responsibility of engine module is to send requests for recommendations to the context aware services and provides recommendations to users. The responsibility of reasoner component is to decide whether or not to send request for recommendations to specific services. The responsibility of communication manager module is to receive the context and space related information from middleware. The responsibility of administration component is to support administration related tasks such as context-aware services registration and policy management etc. Ontologies have been used to model context in PRECISE.

User register him/her to this service. Spatial information are retrieved by services and these services can be on cloud or somewhere else. PRECISE interact with context aware services and get recommendation and then send results to user. PRECISE allows users to define privacy policies to specific services, not just to users.

2.6.4 Service Oriented Recommender System

As service-oriented ideas are getting popular, research community is taking interest in context-aware service provisioning instead of context-aware system reconfiguration. One proof of this claim is implementation of a framework [31] that enables tasks of getting context, find out what context-specific adaptation is required, tailoring nominee services for the context, and executing the adapted service. Consider a smartphone connected to a distant cloud service via Internet. As the context of the user varies, this invoke different cloud services depending upon the users current context. With this type of context-awareness, a service would not be tied up to a user. Instead, when user of request for a cloud service, it is attached to users context detail, and the most appropriate service is chose based on that information. Therefore, context is used to render personalized services, and also as fault tolerant strategy such as rectifying low QoS issues.

The authors propose a context sensitive service provisioning architecture which has tiers. One is user layer that consist of mobile devices upon which the applications operate. The second layer is agent layer that adjusts the services according to user's context. The third layer is service layer that is

responsible for services deployment.

2.7 Conclusions and Comparative Summary

The primary objectives of this chapter is to explore, shortlist and study the published research work in the domain of context aware recommendation system. This literatur review discusses different tool and techniques proposed by researchers for better and effiecnt recommendations, over the passage of time. Table 2.1 summarizes the comparison of state-of-the-art-Research. Research community has given a great attention to the area of mobile recommender in the past few years but existing mobile recommender systems fall short of context-aware recommendations in a smart market domain. Mobile Cloud Computing (MCC) as state-of-the-art for mobile computing needs to be exploited for context-aware recommendations in the context of for smart markets.

Domain	**Mechanism**	**Context Dimension**	**Work**	**Mobile Paradigm**	**Cloud Paradigm**
E-Commerce	Content Filtering	Location	[24]	✓	×
Leisure (music)	Content Filtering, Collaborative Filtering	Weather,Location, Companions	[29]	✓	×
Leisure(Media Items)	Content Filtering, Collaborative Filtering	Location, time of the day, day of the week	[20]	✓	✓
Leisure (movies)	Collaborative Filtering, Knowledge based Modeling	Location, Crowd, Time	[15]	✓	×
Tourism	Collaborative Filtering, hub and authority scores, Ant Colony Algorithm	Time, Location	[23]	✓	✓
Health	Collabrative Filtering, FRSA Algorithm	User preferences, physiological data	[27]	✓	✓
Tourism	Collabrative Filtering, Knowlege based Filtering	Location	[21]	✓	×

Table 2.1: A Comparison Summary of State-of-the-art-Research

Chapter 3

Proposed Architecture of Mobile-Cloud Recommender Framework

3.1 Chapter Overview

In this chapter, we discuss the architecture of proposed solution. We also present in depth details of different modules and underlying layers of the proposed architecture. The output of this chapter is an architecture of the framework to implement context aware recommender system.

3.2 Architectural Overview of the Framework

The proposed mobile-cloud recommender system can be viewed as an enabling mechanism of the smart city systems. Specifically, with the proposed

project we aim to empower the mobile users with a solution that provides customized and localized recommendations to help with appropriate decision making. The project aims to strengthen the engineering and development of a location aware and context sensitive recommender system for the mobile users.As illustrated in Figure 3.1, at a higher level the architecture is viewed as two distinct layers namely mobile computing layer and cloud computing layers. Details about both the layers of the architecture and the components contained in each layer are detailed as below.

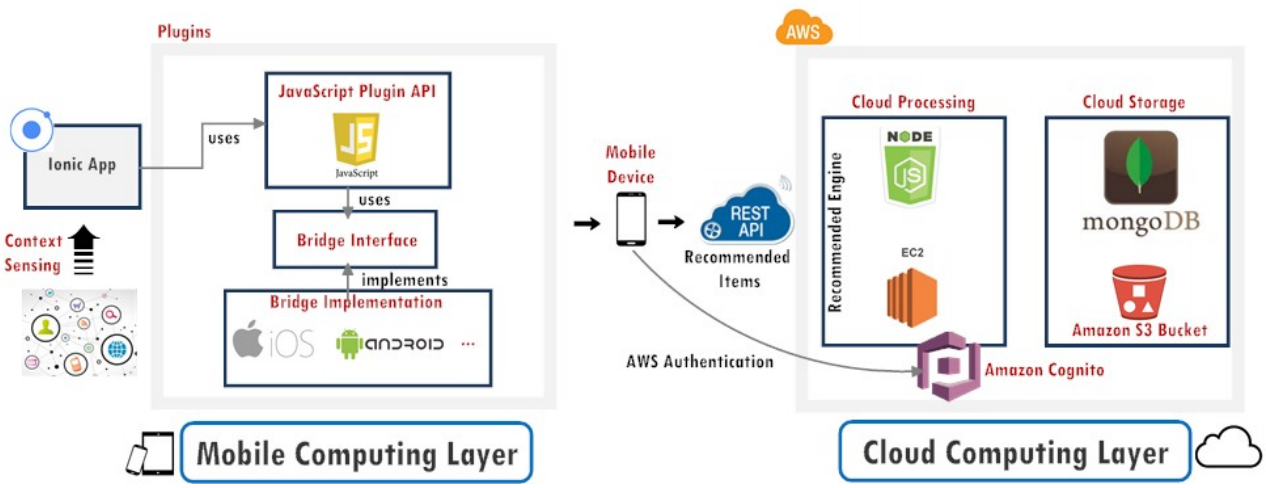

Figure 3.1: An Architectural Overview of Proposed Solution

Mobile computing layer is the front-end layer to which end user interacts with. The core responsibility of this layer is to get users current context and show recommendations to end user after retrieving data from cloud computing layer. In short mobile device is like dumb terminal whose sole responsibility is send context related data and show recommended items to end user. Cloud computing layer utilize the storage as well as processing capabilities to calculate and compute recommendations that are sent back

to the mobile computing layer. In the context of recommendation data, we utilize the cloud-based database and storage to maintain all the context and recommendation information including media items and files to avoid any computational and storage overhead on the mobile device.

Figure 3.1 also shows the detailed picture of the modules of our proposed framework. We discuss every component of framework in the subsequent sections.

3.3 Mobile Computing layer

Mobile Computing as a pervasive technology empowers its users to exploit mobility and context-aware computations. It allows us to manage life , connect easily with family and friends and perform day to day tasks more efficiently. Basically there are three core components in mobile computing i.e.; *(a) Mobile communication (b) Mobile hardware(c) Mobile software.*

1. **Mobile communication**– is the infrastructure to ensure seamless and reliable communication such as services, protocols, bandwidth and portals essential to support and facilitate data transmission.

2. **Mobile hardware**– encompasses a wide range of different mobile devices such as smartphones, tablet and PDAs.

3. **Mobile software**– is the real program that executes on the hardware of mobile device and it is known as operating system of the device.

3.3.1 Ionic Plateform

We utilize HTML5 techonolgies for mobile front end development to target multiple mobile plateforms. Another reason of using hybrid mobile application instead of native mobile application development is that we carry out all performance intensive tasks over cloud layer. Considering the mobile services development via HTML5 techonolgies , we exploit ionic framework to target android and IOS plateform. It is the leading open source framework or SDK to develop awesome hybrid mobile applications. Built on top of Apache Cordova and Angular, Ionic offers services and tools to develop cross-platform mobile applications via web technologies such as JavaScript, HTML5, CSS and SaaS.

Hybrid mobile apps are developed in a similar way as websites. Both use a stack of web technologies like JavaScript, HTML and CSS. However, hybrid apps target a Web View which is rendered within native container instead of mobile browser. This Web View acts as bridge interface between operating system and app as shown in Figure 3.1 and enables hybrid app to access hardware capabilities of the mobile device via plug-ins. Ionic framework exploits Apache Cordova platform that provides a lot of JavaScript APIs to exploit capabilities of device by plug-ins, which are written in native code. The said plug-ins encompass APIs for accessing the devices camera , contacts , accelerometer , GPS and more to retrieve the user current location and contextual information etc.

The landscape of our daily life has been changed due to mobile computing. Some noteworthy advantages of mobile computing are

1. Context Awareness
2. Location Flexibility
3. Portability
4. Enhanced Productivity

Every technology has its limitations and mobile computing is not an exception. Some disadvantages of mobile computing are

1. Energy/Power Constraints
2. Processor and Storage Poverty

To coup the above mentioned challenges regarding computational and storage constraints , we rely upon cloud computing technology that aims to provide elastic storage and virtually unlimited computational power.

3.4 Cloud Computing layer

In layman terms, cloud computing is to store and access data as well as execution of software programs over the Internet rather than hard drive of your computer. It is also called on-demand computing. According to NIST [4] definition cloud computing stack can be divided into three different models of service i.e.; *Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Software as a Service (SaaS).* A short description of these service modelsl is as following:

1. **Software as a Service (SaaS)**–SaaS applications are built for end-users and delivered over the internet. SaaS eliminates the requirement of installation and execution of applications on client computers due to web delivery model.

2. **Platform as a Service (PaaS)**–PaaS is actually a stack of tools and technologies designed for efficient development of software applications and make deployment and configuration of those applications easy as well as quick.

3. **Infrastructure as a Service (IaaS)**–IaaS is self-service model to access, monitor, and manage distant datacenter infrastructures such as processing power, servers, storage, and networking as well as services related to networking (e.g. firewalls).

We exploit Amazon cloud services for storage and computing efficiency. Pricing model adotped by AWS is pay-as-you-go. We launch a virtual server on Amazon cloud called Amazon EC2 instance and set up Red Hat Linux operating system over that instance. We develop server side application using Node.js and set up Node.js web server on Amazon EC2 Instance. For the sake of efficient data retrieval we use Mongo DB. We install MongoDB on Amazon EC2 Instance. To use files and media we utilize Amazon S3 storage services. Cloud end of the proposed framework can be divided into two categories.

(A) Cloud Processing

(B) Cloud Storage

3.4.1 Cloud Processing

3.4.1.1 Amazon EC2

Amazon EC2 is a web service that allows users to rent virtual machines upon which computing application can be easily deployed. These virtuall machines are called Amazon EC2 instances. To utilize EC2 service,a developer configures Amazon Machine Image which contains an operating system, configuration related settings and application programs. Amazon EC2 enables resizable compute capacity and it can be used to launch as few or as many virtual machines / servers as businesses need.

3.4.1.2 Node.js server

Node.js is an open source server side platform for the development variety of scalable as well as fast network applications. Node.js applications are developed using JavaScript and it has rich library support to make developers life easy. It is lightweighted as well as efficient and exploits event driven, asynchronous I/O architecture. We utilize RESTful API of Node.js to develop backend of application.

3.4.1.3 Recommendation Engine

It is the main component of the framework. The task of recommendation engine is to retrieve data from MongoDB collections, run recommendation algorithms and techniques and save the result set back to MongoDB for users recommendations. The algorithmic details are discussed in Chapter 4. The recommendation engine is written in python language.

3.4.2 Cloud Storage

3.4.2.1 MongoDB

MongoDB is an open source database based on document oriented architecture. It falls in the category of NoSQL databases. It has rich query structure and no complex joins are required. MongoDB support horizontal scalability via sharding opting in which data is distributed across multiple machine to provide high availability of data and consistency. Due to huge performance advantages we use MongoDB to store and manage data.

3.4.2.2 Amazon S3

Amazon S3 is cloud service provided by Amazon to store media items such as pictures and audio/video files. It can save and retrieve any amount of data, from anywhere, at any time on the web. This storage service is highly-scalable as well as reliable. It is low-latency data storage infrastructure at very low costs. To obtain such benefits we store media items upon Amazon S3.

3.5 Summary of Chapter

The primary goal of this chapter was to explain different components of the architecture for the proposed framework. We have illustrated the architecture of the framework as two distinct layers namely mobile and cloud computing layers. We have also highlighted the details of the individual components in each of the layer. We discuss the details of different tools and technologies

to render the proposed framework. The proposed architectural representation of framework provide us the foundation to implement the smart market recommender system as presented in Chapter 4.

Chapter 4

Implementation of Smart Market Recommender System

4.1 Chapter Overview

In this chapter we present the techniques and algorithms used to implement proposed system. We demonstrate how mobile layer have been exploited to retrieve contextual information to generate best possible recommendation. We also present the procedure to offload users as well as products related data and computing over cloud end for the sake of storage and power efficiency. We also highlight how the proposed recommendation engine process data offline and generate pre computed similarity table of users and products to overcome scalability issue.

The primary contribution of this chapter is to present the algorithmic and analytical-oriented details of the mobile-cloud recommender system. The algorithms, in addition to demonstrating the functional aspects; also highlight

the system-level inputs and outputs.

4.2 Implementing Context-awareness and Recommentation Processing

We highlight the implementation details of the proposed system from Chapter 3. The system implementation refers to two aspects namely (i) algorithmic details, (ii) executable specifications. This chapter is exclusively focused on the algorithmic details of the system implementation that is independent of any specific tool(s) or technologies. However, to complement the algorithms presented in this chapter, we provide their corresponding; (executable) source code in Appendix A. Both the algorithms detailed in this chapter and the code presented in Appendix A represent the generic as well as concrete implementation of the system.

Many state of the art existing recommender system does not take into account the information regarding context of the user. On the other hand, the aim of the proposed recommender system is to provide information according to the current context of the user. We have incorporated current location of the user, user buying power, gender and age group as contextual information into the process of recommendation. On the basis of current context of the user , our recommender system suggest products near to them and discard the distant ones. We have adopted hybrid approach consist of context information along with content filtering technique and collaborative filtering algorithm to generate best possible recommendations. We have three core

tables for recommendation generation: *user Information table; product information table and user-product rating matrix.* We have implemented indexing upon product and user location in MongoDB to avoid the full scanning of user and product tables. This indexing also help us in fast retrieval of users subset having same demographical region. The utility function (U) of the implemented system is as follow:

$$\mathsf{U}: Contex \times User \times Product \rightarrow \mathsf{Recommendation}$$

The utility function depicts that proposed recommender could recommend the products with highest predicted scores to the users within specific context.The relationship between user and product is based on a rating structure restricted by context of the user that shows the products'usefulness to users.

4.2.1 Assumptions for System Implementation

Before the technical details, we must explicitly highlights the assumptions for system implementations. These assumptions also represent the conditions that must be followed for system implementation and operations. These assumptions are

- **Demographic Trends:** We assume that the user of the same demography have the similar preferences.
- **Contextual Information:** Another assumption is that exploitation of user current location, user buying power, gender and age group as contextual information helps to speed up the process of online recommendation as we dont have to scan the whole dataset.

- **Network Availability:**We assume that client has network availability over mobile device during application use to get recommendations.

- **Geolocation API Accuracy:** We believe that HTML5 Geolocation API provide the precise location of the active user for GPS-enabled devices , like android and iPhone.

- **Privacy Permissions :** We assume that user have allowed permissions to retrieve their geographical position via HTML5 Geolocation API .

- **Modern Browsers Support :** Another assumption is that modern browser is installed in user's device that has full support for Geolocation API.

4.3 Context Aware Mobile Layer - Online Processing for Recommendations

This section described the functionality performed over mobile end of the recommender system. The core responsibility of the mobile layer is to retrieve the current location of the user and send it to cloud end of the system as well as display the recommended products to end user. To find the current location of the user we have used HTML5 Geolocation API. Restful Architecture has been utilized to perform communication between mobile and cloud end of the system. User has to be registered to system to get recommendations. User information is stored into user table/collection

in Mongo DB installed over Amazon EC2. After user get logged into the system user location is retrieved. Current location of the active user and preferences are sent to Node JS server via Restful API. On the server side a method named as recommend-products is called. First of all, all those products available in nearby shops are selected from product table and inserted into a list named NearBy-ProductsList. The second step is to apply content filtering technique to retrieve all the products matching with taste of user and saved in a list named as Content-ProductList. After population of Content-ProductList two parallel methods known as filter-similar-products-table and filter-similar-users-products-table are executed within recommend-product method. The task of filter-similar-products-table method is to find and select top k items rated highly against active user row entry from pre computed table named similar-products table and save those items into a list named as Alike-ProductList. The responsibility of filter-similar-users-products-table is to find and select top k items rated highly against active user row entry from table similar-user-table and save into list know as AlikeUser-ProductsList. The next step is to merge three lists known as Content-ProductList,Alike-ProductList and AlikeUser-ProductsList into one list know as Detailed-ProductsList and remove all duplicated entries of products from the list. In the end context post filtering is applied and only those items are retained which also exist into NearBy-ProductsList to make final recommendation list named Recommended-ProductsList. Finally Recommended-ProductsList is sent back to end user.

4.3.1 Algorithm(s) for Mobile Context Aware Recommendations

This section consist of Algorithm(s) used during the process of online recommendation. Figure 4.1 depicts a visual look and feel of algorithmic

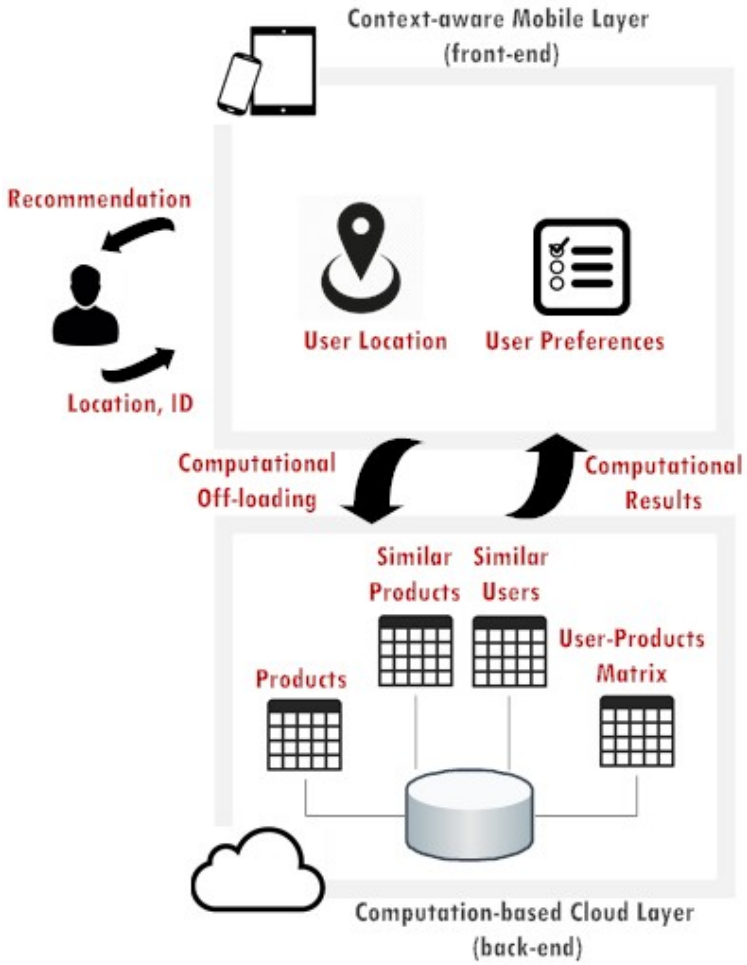

Figure 4.1: Overview of Online and Offline Recommendation's Generation.

details used in recommendation generation.

Table 4.1 contains a list of variables that help parametrisation of algorithms for online recommendation's generation.

Parameter	**Description**
NearBy-ProductsList	The List of all products which are located in a short distance to active user
Content-ProductList	The List of all products which falls in the category similiar to preferences of active user
Alike-ProductList	The List of all products from similiar-products table rated highly against record of active user
AlikeUser-ProductsList	The List of all products from similiar-users table rated highly against record of active user
Detailed-ProductsList	The List is union of three Lists i.e.; Content-ProductList , Alike-ProductList , AlikeUser-ProductsList
Recommended-ProductsList	The List contains only those products which exist in both Detailed-ProductsList and NearBy-ProductsList Lists

Table 4.1: Parameters for Online Recommendation Generation

Table 4.2 describe all the utility methods which have been used during the the process of online recommendation's generation.

Function(param)	**Return**	**Description**
find-similiar-users(ID)	List	Get list of those products from similiar-users table having high scores against active user
find-similiar-products(ID)	List	Get list of those products from similiar-products table having high scores against active user
getLocalProd(GeoLocation)	List	Get list of all products from Product table near to user current location
getPersonalProd(activeUserPreferences)	List	Get list of products from Product table according to user taste
getTopItems(simProdTable)	List	Get list of top k product from similiar-products table rated highly against record of active user
getTopUsersItems(simUserTable)	List	Get list of top k product from similiar-users table rated highly against record of active user

Table 4.2: Utility Methods for Online Recommendation Generation

Algorithm 1: recommend-products(activeuseID , currentLocation , activeUserPreferences) to generate Recommendation

Input : ID , GeoLocation and activeUserPreferences
Output: A List of Recommended Products

1: *NearBy-ProductsList* ← getLocalProd(GeoLocation)
2: *Content-ProductList* ← getPersonalProd(activeUserPreferences)
3: *Alike-ProductList* ← find-similiar-products(userID)
4: *AlikeUser-ProductsList* ← find-similiar-users(userID)
5: *Detailed-ProductsList* ←
Alike-ProductList+ AlikeUser-ProductsList+ Content-ProductList
6: *Recommended-ProductsList* ←
Inetresection of Detailed-ProductsList and NearBy-ProductsList

7: **function** FIND-SIMILIAR-PRODUCTS(ID)
8: *counter* ← 0
9: *similiar-ProductsList* ← empty
if *counter = 0* **then**
| return similiar-ProductsList
else
| *similiar-ProductsList* ← getTopItems(simProdTable)
end
return *similiar-ProductsList*
10: **end function**

11: **function** FIND-SIMILIAR-USERS(ID)
12: *counter* ← 0
13: *AlikeUsers-PredictedList* ← empty
if *ID = NULL* **then**
| return AlikeUsers-PredictedList
else
| *AlikeUsers-PredictedList* ← getTopUsersItems(simUserTable)
end
return *AlikeUsers-PredictedList*
14: **end function**
return *Recommended-ProductsList*

4.4 Processing Based Cloud Layer - Offline Processing for Recommendations

This section described methodology to implement cloud end of the recommender system. It is noteworthy that all computational and storage work for recommendation generation is performed over cloud layer such as node JS server and MongoDB are deployed upon amazon EC2 instance. Another benefit gained by cloud layer is offline processing performed by python based recommendation engine. *The purpose of offline processing is to overcome the problem of scalability.*

To implement offline processing we have used collaborative filtering algorithm. The core idea behind the collaborative filtering technique is to generate recommendations for active user according to rating or behavior of other users having same taste or preferences. There are two popular flavors of the collaborative filtering i.e.; *a) user based filtering b) item base filtering.* The aim of these techniques is to find the similar user or items from the user item space. As user item space grows to millions of users and items, it is not practical to find similar users or items on real time. *So the better option is to have precomputed tables of similar users and items.*

We have utilize the table user-product rating matrix to apply above mentioned techniques and generate precomputed tables of similar users as well as similar products. Python based recommendation engine run these algorithms to refresh pre computed tables on daily basis in late night timings. *To compute item based similarities we have utilized Cosine-based Similarity and for the sake of user based similarities'computation we have utilized Pearson*

correlation method. There are various methods to calculate the similarity between items. Here we introduce two such techniques which have been exploited to perform this research work. The first one is cosine-based similarity and the other one is correlation-based similarity [32].

1. **Cosine-based Similarity**– In this method, two items are considered as two vectors in the m dimensional space of user. The similarity between these two vectors is determined by computing the cosine of the angle between them. Formally, in the $m \times n$ scores matrix, similarity between items x and y, denoted by sim(x,y) is as follows

$$sim(\mathbf{x}, \mathbf{y}) = \cos(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \frac{\tilde{\mathbf{x}}.\tilde{\mathbf{y}}}{\|\tilde{\mathbf{x}}\|_2 * \|\tilde{\mathbf{y}}\|_2} \tag{4.1}$$

where "." represents dot product of vectors x and y.

2. **Pearson Correlation Score**– A bit more advanced method to compute the similarity between users interests is to exploit Pearson correlation coefficient. In this technique, similarity between two items x and y is determined by measuring the Pearson correlation coefficient. This formula has a good probability to produce better outcomes in scenarios where the data normalization is poor. Let the set of customers who rated both items x and y are denoted by C then the Pearson correlation similarity is given by

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\sum c \in C(S_{c,x} - \vec{S_x})(S_{c,y} - \vec{S_y})}{\sqrt{\Sigma c \in C(S_{c,x} - \vec{S_x})^2}\sqrt{\Sigma c \in C(S_{c,y} - \vec{S_y})^2}} \tag{4.2}$$

Here $S_{c,x}$ represents the score of Customer c on item x and S_x is the

average score of the x-th item.

We have unified the item based collaborative filtering technique with user based collaborative filtering technique during offline processing. Here is the brief overview of these both techniques in our scenario.

1. **User Based filtering**– In this technique first of all neighbor users with the similar preferences or past rating history are find out for current user. After finding neighbor users, rating score is predicted for all those product items which has not been explored by current user on the basis of neighbors ratings. *To speed up this process we have exploited clustering technique and created clusters of usesr from the same demography by applying indexing technique provided by MongoDB over Geolocation of the user.*

   The general process is to pick a current user and target product item which has not been rated by current user. Than find all neighbors of current user which have rated those items via Pearson correlation method [33]. It is worth remembering that only those users have been considered which are within the same demographic during creation of neighbors users set.In the end rating of the target product item is calculated on the basis of weighted average of neighbor users'rating. The weight of each users rating score depends on how similar is they are to current user.

2. **Items based filtering** – In this technique first of all users which have rated target product are selected. In the next step all products are searched which have been rated by those users. After this product

similarity is calculated by using Cosine similiarity[33] and a subset of most similar product is created. On the basis of most similar products rating the score for the target product is predicted.

4.4.1 Algorithm(s) for Cloud Based Processing of Recommendations

This section contains pseudo code for the Algorithm(s) used during the process of offline recommendations computation.

Table 4.3 contains a list of variables that help parametrisation of algorithms for offline recommendation's computation. Table 4.4 describe all the utility methods which have been used during the the process of offline recommendation's computation.

Parameter	Description
simSet	The List contains the values of similiarity between different products or users
simProdSet	The List of all products which are more similiar to each other than other ones
simUsersProdSet	The List of all users which are more similiar to each other than other ones
productCounter	Total Number of unique product in similiar-products table
userCounter	Total Number of unique users in similiar-users table

Table 4.3: Parameters for Offline Recommendation Computation

Function(param)	**Return**	**Description**
ComputePearson(testItem,itemSet)	List of Floats	return the Pearson corelation value between each item into itemSet and testItem
getUserSetSimProd(targetProduct)	List	select users who have rated the targetProduct
getProdSetSimProd(userSet,targetProduct)	List	select all other Products rated by the userSet who have rated targetProduct
getSimProd(simSet)	List	select k most similar product form simSet
predictSimProd(SimProdSet,targetProduct)	Void	Compute the weighted average of SimProdSet and predict ratings for the targetProduct and update table
getProdSetSimUsers(currentUser)	List	select items rated by the currentUser
getUserSetSimUsers(productSet)	List	select user which have rated more than one items in productSet
getSimUserProd(simSet)	List	select k most similar product form simSet
predictSimProd(simUsersProdSet,currentUser)	Void	Compute the weighted average of simUsersProdSet and predic ratings for the currentUser's unrated products and update table

Table 4.4: Utility Methods for Offline Recommendation Computation

Algorithm 2: similiarProducts(User-Product rating matrix)

Input : User-Product rating matrix

Output: Output: Boolean [TRUE/FALSE] to indicates wheather all products rating have been updated or not

1: *counter* ← 0

2: *productCounter* ← total Number of products

do

targetProduct ← product[counter] ;

userSet ← getUserSetSimProd(targetProduct) ;

productSet ← getProdSetSimProd(userSet,targetProduct) ;

simSet ← ComputeCosine(targetProduct , productSet) ;

SimProdSet ← getSimProd(simSet) ;

predictSimProd(SimProdSet,targetProduct) ;

while *counter !=productCounter*;

if *counter !=productCounter* **then**

return *False*

else

return *True*

end

Algorithm 3: similiarUsers(User-Product rating matrix)

Input : User-Product rating matrix

Output: Output: Boolean [TRUE/FALSE] to indicates wheather all users rating have been updated or not

1: *counter* ← 0

2: *userCount* ← total Number of users **do**

currentUser ← user[counter] ;

productSet ← getProdSetSimUsers(currentUser) ;

userSet ← getUserSetSimUsers(productSet) ;

simSet ← ComputePearson(user , usertSet) ;

simUsersProdSet ← getSimUserProd(simSet) ;

predictSimProd(simUsersProdSet,currentUser) ;

while *counter != userCount*;

if *counter != userCount* **then**

return *False*

else

return *True*

end

4.5 Summary of Chapter

In this chapter we have described the process of recommendation generation. We have highlighted the role of mobile front end layer for retrieval of user context and push related recommendation on mobile screen. We have also discussed the importance of cloud computing to solve the scalability as well power consumption, computing and storage issues. We have write down the details algorithmic techniques for the unification of mobile and cloud computing layer to generate better recommendation in a scalable and power efficient way. The algorithmic and implementation details help us to evaluate the solution as detailed in the next chapter.

Chapter 5

Evaluation of Recommender System

5.1 Overview of the Chapter

In this chapter, we present the evaluations of the proposed recommender system. We utilise the ISO/IEC model to evaluate the quality of the software. We present the dataset for the evaluation and also highlight the criteria for the evaluation. We present a short discussion about validity threats as well future dimensions for evaluation of the proposed recommender system. The chapter provides the validation of the research and also indicates areas of future improvements and evaluations.

5.2 Context, Objectives and Methodology for Evaluation

5.2.1 Context and Methodology of Evaluation

Recommender system can be evaluated by using different options. It is necessary to specify precisely how the evaluation is exercised to be able to compare results easily. To evaluate the proposed recommender framework we have followed the software product quality standards called ISO/IEC 9126 – 1 [34]. ISO/IEC 9126 – 1 [34] is an international standard to measure the quality attributes and features of a software solution. It is noteworthy that ISO/IEC 9126 – 1 is a theoretical model which consist of set of well structured attributes and features to evaluate software quality and it must be accompanied with concrete evaluation scheme.

5.2.2 Objectives of the Evaluation

The main goal of the evaluation is to validate the quality aspects defined in ISO/IEC 9126 - 1 model including functionality as well as efficiency of the proposed recommender framework. By evaluating these quality features and attributes we can easily validate the research hypothesis as listed in Chapter 1 of report.

5.3 ISO/IEC 9126 Model for Evaluation

To evaluate quality features and characteristics of a software solution or product, International Organization for Standardization (ISO) introduced ISO/IEC 9126 – 1 standard [34] on 2001 [1]. The ISO/IEC 9126 1 quality model investigates six quality features or characteristics i.e. *a) functionality, b) reliability, c) usability, d) efficiency, e) maintainability, and f) portability* for the sake of a software solution or product's evaluation. Moreover, theses six quality attributes are further categorized into 27 sub-categories of software product quality evaluation. For example, the quality characteristic for efficiency consists of five sub-characteristics including *Time behavior, resource utilization, and efficiency compliance.* The in depth discussion of quality sub-characteristics of ISO/IEC 9126 - 1 model is not in the scope of this document and for more details please refer to [34].

5.3.1 Criteria for Evaluation

To evaluate the proposed recommender framework, we only consider two quality characteristics of ISO/IEC 9126 – 1 quality model *Accuracy* and *Efficiency* and their sub characteristics.

5.3.1.1 Accuracy of System Recommendations

The accuracy of the recommender system is of utmost importance in context-aware recommendations. Specifically, we aim to evaluate how accurate the

[1] It is noteworthy that, ISO/IEC 9126–1 was first published in 1991; and later on from the year 2001 to year 2004 ISO published an international standard (ISO/IEC 9126 – 1) as well three technical reports (ISO/IEC 9126 – 2 to ISO/IEC 9126 – 4)

proposed system is to present the results as per the preferences and the context of the user. We aim to evaluate the accuracy of the system by means of

- **Precision of the Recommendations**
- **Recall of the Recommendation System**

The technical details about the precision and recall based evaluations are presented in Section 5.4 .

5.3.1.2 Efficiency of System Recommendations

In terms of the efficiency, we aim to evaluate how efficient if the proposed recommender system. In the context of mobile devices (that are resource constrained computers) and specifically a mobile recommender system the system must efficient. We evaluate the system efficieny in terms of

- **Energy Efficiency of the Recommender System**
- **Computational Efficiency of the Recommender System**

Technical details about the evaluation of system efficiency are discussed in Section 5.5 .

5.3.2 Data Set and Use Cases for Evaluation

For the sake of evaluation I am having a Huawei P8 lite device. Besides, I plan to request my family, friends as well as colleagues to install and use the application.

From their app usage I would be able to build a real data set. We have used sample dataset of superstore sales to evaluate the proposed recommender system [35]. With the help of this dataset we have created a new dataset. We have altered geographic locations (regions / provinces) to include Pakistani locations(regions / cities). We have used the geographic locations (regions / provinces) as location of different stores and add new attributes longitude and latitudes against that geographic location. We have randomly added customer's ratings with scale of (1-5) against every product shipped to that customer. We have created a random password and user id against every user. We have only consider customer name, product name, product category, product sub-category,price and geographic locations (Province and Region) attributes to create a new dataset.

5.3.2.1 Use Cases for Evaluating Recommendation Accuracy

Use cases for accuracy evaluation are decribed in Table 5.1 and Table 5.2.

Use Case 01	
Use Case ID:	Ac1
Use Case Name:	Recommendations for Newly Registered user
Actor:	Mobile device user
Description:	If the user register for the first time than according to his provided demography and preferences a list of recommended products within λ distance from current location context of user will be generated and display on mobile screen
Pre-conditions:	1. App must be installed in the mobile device. 2. Modern browser is installed in user's device that has full support for Geolocation API 3. Mobile device has network availability either Wi-Fi network or mobile data
Post-conditions:	User profile data as well as rating to recommended products will be saved in the database for future use
Priority:	High
Frequency of Use:	Repeatedly
Exceptions:	In case of any exception, the previous status must be restored.
Assumptions:	NIL
Notes and Issues:	NIL

Table 5.1: Use Case 01 for Accuracy Evaluation

Use Case 02	
Use Case ID:	Ac2
Use Case Name:	Recommendations for Existing user
Actor:	Mobile device user
Description:	If the already registered user login and request for recommendation than a list of recommended products according to user profile and past history within λ distance from current location context of user will be generated and display on mobile screen
Pre-conditions:	1. App must be installed in the mobile device. 2. Modern browser is installed in user's device that has full support for Geolocation API 3. Mobile device has network availability either Wi-Fi network or mobile data
Post-conditions:	The profile of user will be updated and ratings for the recommended products will be saved into database
Priority:	High
Frequency of Use:	Repeatedly
Exceptions:	In case of any exception, the previous status must be restored.
Assumptions:	NIL
Notes and Issues:	NIL

Table 5.2: Use Case 02 for Accuracy Evaluation

5.3.2.2 Use Cases for Evaluating Recommendation Efficiency

Use cases for efficiency evaluation are decribed in Table 5.3 and Table 5.4.

Use Case 01	
Use Case ID:	Ef1
Use Case Name:	Battery and Power consumption
Actor:	Mobile device
Description:	Battery and power usage must not surpass a specific limit (it should not increase from p %).
Pre-conditions:	1. App must be installed in the mobile device. 2. Modern browser is installed in user's device that has full support for Geolocation API 3. Mobile device has network availability either Wi-Fi network or mobile data
Post-conditions:	NIL
Priority:	High

Table 5.3: Use Case 01 for Efficiency Evaluation

Use Case 02	
Use Case ID:	Ef2
Use Case Name:	Computational Efficiency
Actor:	Mobile device user
Description:	Does the list of recommended items of returned form server and display on mobile screen within Δ time and app does not in a continuous wait state?
Pre-conditions:	1. App must be installed in the mobile device. 2. Modern browser is installed in user's device that has full support for Geolocation API 3. Mobile device has network availability either Wi-Fi network or mobile data
Post-conditions:	NIL
Priority:	High
Frequency of Use:	Repeatedly
Exceptions:	In case of any exception, the previous status must be restored.
Assumptions:	NIL
Notes and Issues:	NIL

Table 5.4: Use Case 02 for Efficiency Evaluation

5.4 Evaluating System Accuracy for Recommendations

The accuracy of the proposed system is observed to be 100 percent when the user allows to exploit her current location by the app. Otherwise app will not recommend nearby products. Another important parameter is the preferences of user. If user does not select her preferences during registration than there are fair chances that the recommendation will not accurate. The app will try to recommend the nearby products according to demographic information of the user in which there are possibilities of wrong recommendations. To measure the effectiveness of the recommendation we have utilized following metrics *1) Precision 2) Recall.* Precision is the ratio of accurate recommendations (tp (true positives)) to the entire count of recommendations (true positives + (fp) false positives). The Mathematical representation of precision is as follow

$$Precision = \frac{tp}{tp + fp} \tag{5.1}$$

Recall is the ratio of accurate recommendations (tp (true positives)) to the total number of generated recommendations (true positives + (fn) false negative) by the recommender. The Mathematical representation of precision is as follow

$$Recall = \frac{tp}{tp + fn} \tag{5.2}$$

We have observed that average precision and recall of system is 80.4% and 64.8 % respectively in case of newly registered users (Cold Start Scenarios). On the other hand average precision and recall of system is 82.6% and 72.6

% respectively in case of already existing users.

5.4.1 Accuracy of Computing Recommendations

We have also generated a list of recommendation manually. We have observed that it takes a lot of time to generate list of recommended items. A sample of record set has been selected from product table as shown into subsect 5.4.1.1 And we select one record of user as shown into subsect 5.4.1.2 to generate recommendation against that user record. Right Now the size a Product table is *8400 rows*. It takes almost 8 minutes to list down only 30 rows from product if we only consider the tags of the product and use record in same city. *A primary drawback of manual method is that we cannot incorporate current location of user as location is retrieved via mobile device by exploiting GPS*. While the proposed recommender take approximately one second to retrieve list of recommended item *sorted by distance*. On the other hand it not practical to generate similarUsers and similar Products tables using Pearson correlation and cosine similarity techniques manually as theoretically it will take months or year to perform these tasks manually. While our offline recommender system take 5-6 hours to generate similarUsers and similar Products tables for the sake of better recommendations.

5.4.1.1 Sample Records from Product Table


```
1 {"_id":{"$oid":"58a585853ad3bd22cc502a17"},"productName":"
    Hewlett-Packard cp1700 [D, PS] Series Color Inkjet Printers
    ","geoLoc":{"type":"Point","coordinates":[73.055522,33.7206
```

```
   18]},"tags":["Technology","Office Machines"],"city":"
   Islamabad","productCat":"Technology","productSubCat":"
   Office Machines","price":500.98,"imageUrl":"https://s3.
   amazonaws.com/uploadedpictures/636229589702545009.Jpeg","
   baseProductId":{"$oid":"58af137f3ad3bd5548204286"},"
   isDelete":false}
2
3
4  {"_id":{"$oid":"58a5857d3ad3bd22cc5028bd"},"productName":"i470"
   ,"geoLoc":{"type":"Point","coordinates":[73.049457,33.70686
   ]},"tags":["Technology","Telephones and Communication"],"
   city":"Islamabad","productCat":"Technology","productSubCat"
   :"Telephones and Communication","price":205.99,"imageUrl":"
   https://s3.amazonaws.com/uploadedpictures/63622934169449078
   2.Jpeg","baseProductId":{"$oid":"58af13763ad3bd554820404e"}
   ,"isDelete":false}
5
6  {"_id":{"$oid":"58a585853ad3bd22cc502a0b"},"productName":"8890"
   ,"geoLoc":{"type":"Point","coordinates":[73.055522,33.72061
   8]},"tags":["Technology","Telephones and Communication"],"
   city":"Islamabad","productCat":"Technology","productSubCat"
   :"Telephones and Communication","price":115.99,"imageUrl":"
   https://s3.amazonaws.com/uploadedpictures/63622958915562036
   2.Jpeg","baseProductId":{"$oid":"58af137f3ad3bd554820426f"}
```

```
,"isDelete":false}
```


5.4.1.2 Sample Records from Users Table


```
{"_id":{"$oid":"58a56e323ad3bd233ca55df1"},"userName":"Muhammed
 MacIntyre","geoAddress":{"type":"Point","coordinates":[33.
729388,73.093146]},"plainAddress":"Islamabad","emailAddress
":"MuhammedMacIntyre@raasapp.testdsmscs.nust.seecs.edu.pk",
"password":"abcd1234","type":"U","ageIndex":3,"
ageGroupDecription":"36-45","gender":"M","genderAccuracy":0
.93,"prefferedDistance":20.0,"discovery":true,"isDelete":
false,"tags":["Office Supplies","Storage And Organization",
"Binders and Binder Accessories","Paper","Furniture","
Chairs And Chairmats","Technology","Computer Peripherals","
Office Machines","Pens And Art Supplies","Rubber Bands"]}
```


We have used a list of simulated users to evaluate the accuracy of Recommender.We have created 5 users having different demographical information as well as different preferences. The users belongs to different age groups and different gender group. Than we have generated personalized recommendation list according to simulated location of the user and recorded recall and precision. The simulated results have been shown in Table 5.5 and Table 5.6 for newly registered users (Cold Start Scenarios) and existing users respectively. Column "Specific Category Total" shows the total number of records about particular topic in the database. Column "Recommended" represent

the count of retrieved records while Column "Relevant" represents the number of records which are relevant to users preferences. The last row in both tables describes the average for each column.

User	Specific Category Total	Recommended	Relevant	Precision	Recall
U1	81	63	45	71%	55%
U2	96	72	60	83%	62%
U1	98	98	84	85%	85%
U2	90	60	50	83%	55%
U1	91	65	52	80%	57%
Avg.	91.2	71.6	58.2	80.4%	64.8%

Table 5.5: Precision and Recall for newly Registered User

User	Specific Category Total	Recommended	Relevant	Precision	Recall
U1	84	96	72	75%	85%
U2	99	77	66	85%	66%
U1	72	63	54	85%	75%
U2	88	66	55	83%	62%
U1	80	70	60	85%	75%
Avg.	84.6	74.4	61.4	82.6%	72.6%

Table 5.6: Precision and Recall for Existing User

5.5 Evaluating System Efficiency for Recommendations

To assess the efficiency of the developed app, we have monitored memory and CPU usage via CPU Monitor app [36]. We have observed that the developed app takes approximately three second to fetch and display list

of recommended items from node server to end user respectively. We have analyzed the consumption of CPU as well as memory in following cases

- When developed app executes in background
- When developed app executes in foreground

The consumption of CPU and RAM when app is being executed in foreground has been shown in Figure 5.1. It is noteworthy that maximum CPU consumption does not exceed 4 percent and the usage of RAM is 157 MB when app is performing tasks in foreground.

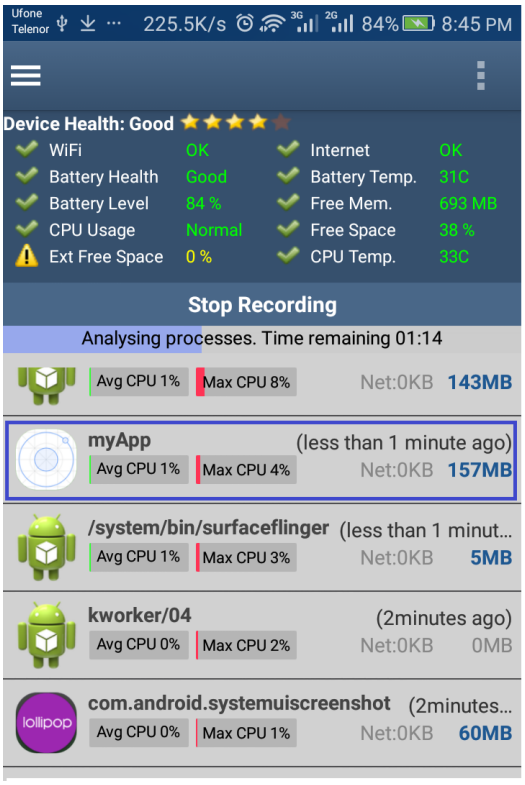

Figure 5.1: CPU and Memory Consumption in Foreground

The consumption of CPU and RAM when app is being executed in background has been shown in figure Figure 5.2 . It has been observed that

maximum CPU usage is 2 % and the consumption of RAM is round about 146 MB when app is performing tasks in background.

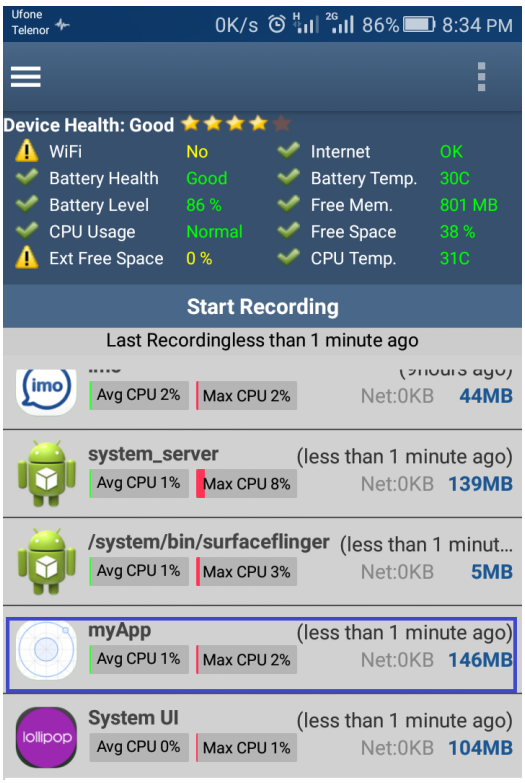

Figure 5.2: CPU and Memory Consumption in Background

We have exploited AccuBattery app [37] to check the battery consumption by the app. We have noticed that battery usage is normal in both scenarios whether app is running in background or being executed on foreground. The battery usage is 0.2 % when app is processing in background. On the other hand it its usage of battery is 0.4 % as show in Figure 5.3 and Figure 5.4 respectively.

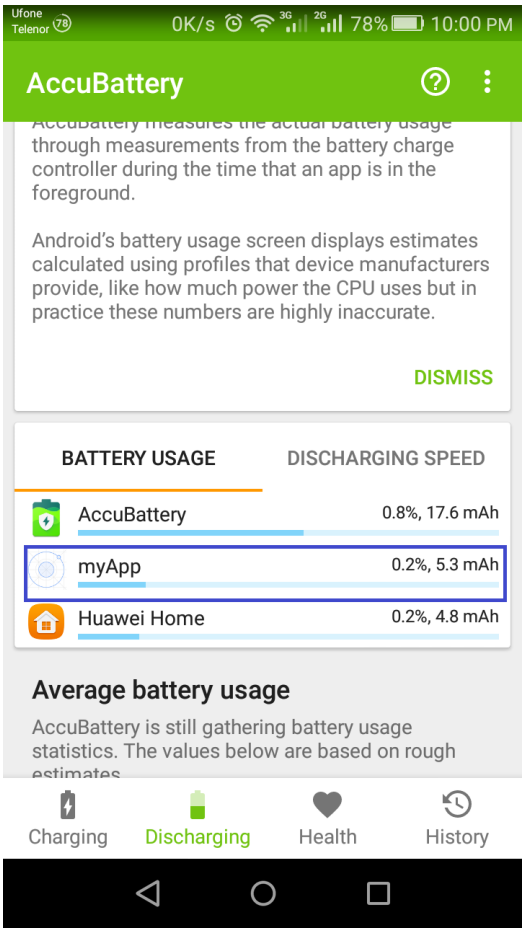

Figure 5.3: Battery Consumption in Background

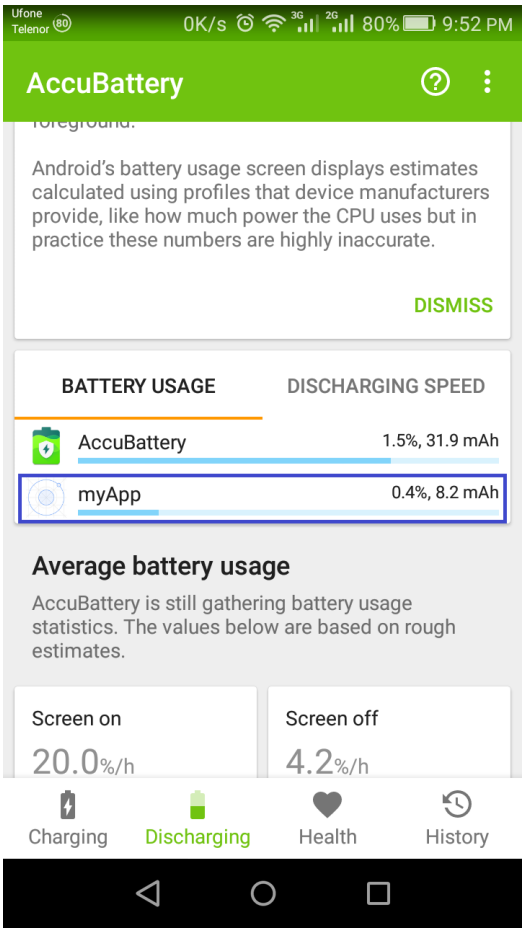

Figure 5.4: Battery Consumption in Foreground

5.6 Validity Threats and Future Evaluations

In this section we describe the validity threats to this research and provide some suggestions to incorporate as future work that can possibly helpful to minimize these threats.

- **Threat I–Availability of Limited Dataset:** The size of data set against which the evaluation has been performed is not big enough. There is a need for more diverse data and real world validations that

define the scope of the future research.

- **Threat II– System Validations in Real Markets Scenarios:** The upfront challanges is an accurate recommender system. We have addressed this challange by providing the algorithms that automate and customise the reommendation process. However, there is a need for more real worl validations in the context of real markets. Due to time constrainsts and availability of data, we plan to do real world validations and fine tuning the prototype in our future work.

5.7 Chapter Summary

We have exploited the ISO/IEC 9126 - 1 quality model to measure the efficiency and accuracy of the recommender system. We have list down the use cases against which effectiveness and efficiency of the proposed system has been evaluated. The technical details regarding evaluation have also been discussed in details. The evaluation results proves that the app uses only a small portion of memory and computational resources and display accurate recommendations to end user. It has also been observed that battery usage by app is normal. Moreover, we have shed light on validity threats to evaluation of research and how those threats can be avoided as part of future work.

Chapter 6

Conclusions and Future Work

6.1 Overview of the Chapter

The focus of this chapter to present the conclusion of our research work by describing the contribution and indicating the direction for future work in the domain of context aware recommendation system. First, we talk about the core contributions of this research. Then highlighted the possible dimensions of future research.

6.2 Summary of Research Contributions

Mobile Computing as a pervasive technology has gained much popularity in current era in masses as well as research community. Mobile cloud computing as state of art mobile computing is a special focus of current research in this domian. As with other research area mobile cloud computing has a plethora of challenges that researchers are currently facing. We list down the

contributions of this research work as:

- We performed a systematic literature survey to discover and categorize the available evidence about context aware recommender systems and describe a comparison of existing research in this domain to highlight its potential, limitations and future work indication. Chapter 2 can be considered as a core contribution in knowledge base of literature.

- Existing mobile recommender systems fall short of context-aware recommendations in a smart market domain. In this research work context-aware recommender system has been developed for effective match making between sellers and customers.

- Mobile Cloud Computing (MCC) needs to be exploited for context-aware recommendations in the context of smart markets. This research work has introduced a system architecture with enabling technologies and implementation platform to develop a mobile-cloud context aware recommender system.For the sake of enery and computational efficiency we have utilized the concept of cloud computing. Overall, the solution integrates the idea of mobile computing with cloud computing.

6.3 Dimensions of Future Research

We believe that our research work will help research community in advancing the research state-of-the-art in the domain of mobile recommender systems. Following are some possible directions as future work of this research:

- We have consider current location, buying power, gender and age group of user as context in this research work. Other contextual factors such as weather conditions and week of the day etc. can also be exploited for better recommendations.

- The data size used during research work was limited. There is need of diverse and big data set to generate better recommendation and evaluation of the recommender system. The collection of data from representative sources is time-taking activity and it requires months or years for the gathering of representative data [38] [39]. The task of huge datasets collection and evaluate the performance and accuracy of system using such data set indicates the dimensions of the future research.

- The evaluation of the developed app has been performed against android based devices only. The future work is to evaluate against iOS based devices.

- We have utilized cosine based similiarity and pearson correlation for computation of item based and user based similiarity respectively. There also exists many other techniques and ways to compute similiarity. Other similiarity techniques can also be exploited to see which produce better results.

- We have utilized traditional methods to implement the proposed app in the domain of smart markets. We aim to apply deep learning model along with machine learning algorithms as part of future work. Al-

though recently deep learning techniques [40] have been utilized in multiple domain of machine learning, such as recognition of audio , natural language processing [41, 42, 43] and computer vision. However, deep learning techniques have not been examined well in the area of recommender systems yet. Deep learning can be applied to improve recommendation quality as future research work.

Bibliography

[1] W. M. da Silva, A. Alvaro, G. H. Tomas, R. A. Afonso, K. L. Dias, and V. C. Garcia, "Smart cities software architectures: a survey," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 1722–1727.

[2] M. Satyanarayanan, "Mobile computing: the next decade," in *Proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond*. ACM, 2010, p. 5.

[3] F. Sanchez, M. Barrilero, S. Uribe, F. Alvarez, A. Tena, and J. M. Menendez, "Social and content hybrid image recommender system for mobile social networks," *Mobile Networks and Applications*, vol. 17, no. 6, pp. 782–795, 2012.

[4] P. Mell and T. Grance, "The nist definition of cloud computing," *Communications of the ACM*, vol. 53, no. 6, p. 50, 2010.

[5] S. Kitanov and T. Janevski, "State of the art: Mobile cloud computing," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2014 Sixth International Conference on*. IEEE, 2014, pp. 153–158.

[6] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci, "Context relevance assessment and exploitation in mobile recommender systems," *Personal and Ubiquitous Computing*, vol. 16, no. 5, pp. 507–526, 2012.

[7] M. Szczerbak, F. Toutain, A. Bouabdallah, and J.-M. Bonnin, "Collaborative context experience in a phonebook," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on.* IEEE, 2012, pp. 1275–1281.

[8] Y. Mo, J. Chen, X. Xie, C. Luo, and L. T. Yang, "Cloud-based mobile multimedia recommendation system with user behavior information," *IEEE Systems Journal*, vol. 8, no. 1, pp. 184–193, 2014.

[9] M. Roberts, N. Ducheneaut, B. Begole, K. Partridge, B. Price, V. Bellotti, A. Walendowski, and P. Rasmussen, "Scalable architecture for context-aware activity-detecting mobile recommendation systems," in *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a.* IEEE, 2008, pp. 1–6.

[10] T. Wang and L. Liu, "Privacy-aware mobile services over road networks," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 1042–1053, 2009.

[11] M. Ali, J. M. Zain, M. F. Zolkipli, and G. Badshah, "Mobile cloud computing & mobile battery augmentation techniques: A survey," in *Research and Development (SCOReD), 2014 IEEE Student Conference on.* IEEE, 2014, pp. 1–6.

[12] G. A. Lewis, P. Lago, and G. Procaccianti, "Architecture strategies for cyber-foraging: Preliminary results from a systematic literature review," in *European Conference on Software Architecture*. Springer, 2014, pp. 154–169.

[13] P. Stuedi, I. Mohomed, and D. Terry, "Wherestore: Location-based data storage for mobile devices interacting with the cloud," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*. ACM, 2010, p. 1.

[14] A. H. Celdrán, M. G. Pérez, F. J. G. Clemente, and G. M. Pérez, "Design of a recommender system based on users behavior and collaborative location and tracking," *Journal of Computational Science*, vol. 12, pp. 83–94, 2016.

[15] L. O. Colombo-Mendoza, R. Valencia-García, A. Rodríguez-González, G. Alor-Hernández, and J. J. Samper-Zapater, "Recommetz: A context-aware knowledge-based mobile recommender system for movie showtimes," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1202–1222, 2015.

[16] J. Zhou, K. Athukorala, E. Gilman, J. Riekki, and M. Ylianttila, "Cloud architecture for dynamic service composition," *International Journal of Grid and High Performance Computing (IJGHPC)*, vol. 4, no. 2, pp. 17–31, 2012.

[17] S. Sharma and D. Kaur, "Location based context aware recommender system through user defined rules," in *Computing, Communication &*

Automation (ICCCA), 2015 International Conference on. IEEE, 2015, pp. 257–261.

[18] M. Braunhofer, M. Elahi, and F. Ricci, "Sts: A context-aware mobile recommender system for places of interest." in *UMAP Workshops.* Citeseer, 2014.

[19] S. D. Gosling, P. J. Rentfrow, and W. B. Swann, "A very brief measure of the big-five personality domains," *Journal of Research in personality*, vol. 37, no. 6, pp. 504–528, 2003.

[20] A. Moradeyo Otebolaku and M. T. Andrade, "Supporting context-aware cloud-based media recommendations for smartphones," in *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference.* IEEE, 2014, pp. 109–116.

[21] J. M. Noguera, M. J. Barranco, R. J. Segura, and L. MartíNez, "A mobile 3d-gis hybrid recommender system for tourism," *Information Sciences*, vol. 215, pp. 37–52, 2012.

[22] J. Postel, "Rfc 793: Transmission control protocol, september 1981," *Status: Standard*, vol. 88, 2003.

[23] O. Khalid, M. U. S. Khan, S. U. Khan, and A. Y. Zomaya, "Omnisuggest: A ubiquitous cloud-based context-aware recommendation system for mobile social networks," *Services Computing, IEEE Transactions*, vol. 7, no. 3, pp. 401–414, 2014.

[24] W.-S. Yang, H.-C. Cheng, and J.-B. Dia, "A location-aware recommender system for mobile shopping environments," *Expert Systems with Applications*, vol. 34, pp. 437–445, 2008.

[25] M. Wiesner and D. Pfeifer, "Health recommender systems: concepts, requirements, technical basics and challenges," *International journal of environmental research and public health*, vol. 11, no. 3, pp. 2580–2607, 2014.

[26] L. Hasman, "An introduction to consumer health apps for the iphone," *Journal of Consumer Health On the Internet*, vol. 15, no. 4, pp. 322–329, 2011.

[27] S.-L. Wang, Y. L. Chen, A. M.-H. Kuo, H.-M. Chen, and Y. S. Shiu, "Design and evaluation of a cloud-based mobile health information recommendation system on wireless sensor networks," *Computers & Electrical Engineering*, vol. 49, pp. 221–235, 2016.

[28] J.-H. Su, B.-W. Wang, C.-Y. Hsiao, and V. S. Tseng, "Personalized rough-set-based recommendation by integrating multiple contents and collaborative information," *Information Sciences*, vol. 180, no. 1, pp. 113–131, 2010.

[29] C. D. Wolfgang Beer, Walter Hargassner and S. Herramhof, "General framework for context-aware recommendation of social events," 2013.

[30] M. del Carmen Rodr?guez-Hern andez and S. Ilarri, "Towards a context-aware mobile recommendation architecture," in *Mobile Web Information Systems*. Springer, 2014, pp. 56–70.

[31] H. J. La and S. D. Kim, "A conceptual framework for provisioning context-aware mobile cloud services," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference*. IEEE, 2010, pp. 466–473.

[32] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[33] T. Segaran, *Programming collective intelligence: building smart web 2.0 applications*. " O'Reilly Media, Inc.", 2007.

[34] H.-W. Jung, S.-G. Kim, and C.-S. Chung, "Measuring software product quality: A survey of iso/iec 9126," *IEEE software*, vol. 21, no. 5, pp. 88–92, 2004.

[35] "Sample - superstore sales," https://community.tableau.com/docs/DOC-1236, last modified by: Micheal Martin.

[36] "Cpumonitor(version-6.54)," https://play.google.com/store/apps.

[37] "Accbattery(version-1.1.7)," https://play.google.com/store/apps.

[38] J. Buckley, T. Mens, M. Zenger, A. Rashid, and G. Kniesel, "Towards a taxonomy of software change," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 17, no. 5, pp. 309–332, 2005.

[39] H. Kagdi, M. L. Collard, and J. I. Maletic, "A survey and taxonomy of approaches for mining software repositories in the context of software evolution," *Journal of software maintenance and evolution: Research and practice*, vol. 19, no. 2, pp. 77–131, 2007.

[40] L. Zheng, "A survey and critique of deep learning on recommender systems," 2016.

[41] W.-t. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering." in *ACL (2)*. Citeseer, 2014, pp. 643–648.

[42] F. Meng, Z. Lu, M. Wang, H. Li, W. Jiang, and Q. Liu, "Encoding source language with convolutional neural network for machine translation," *arXiv preprint arXiv:1503.01838*, 2015.

[43] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[44] "Determine the gender of a first name," https://genderize.io/, powered by: DISQUS.

[45] "Google custom search," https://developers.google.com/custom-search/, powered by: Google Developers.

Appendix A

Source Code

A.1 Source Code Offline Recommendation

```
import os
from pymongo import MongoClient
from bson.objectid import ObjectId
from math import sqrt, radians,pow
import datetime
client = MongoClient('localhost:27017')
db = client.mydb

#Start Of SimiliarUser Method

def similiarUserSet(userRecord):
    simList = []
```

```
    for u in db.raasUsers.find():
        if u['gender'] == userRecord['gender'] and
           u['plainAddress'] == userRecord['plainAddress'] and
           u['ageIndex'] == userRecord['ageIndex']:
           simList.append(u['_id'])

    return simList;

def sim_Pearson(u1,u2):
    # Get the list of mutually rated items
    plist1 = []
    plist2 = []
    si = {}
    for k, v in u1.items():
        plist1.append(k)
    for k, v in u2.items():
        plist2.append(k)

    for item in plist1:
        if item in plist2: si[item] = 1

    n = len(si)
    if n == 0: return 0
    sum1=0
    sum2=0
    sum1Sq=0
```

```
    sum2Sq=0
    pSum=0
    for item in si:
        sum1=sum1+u1[item]
        sum2 = sum2 + u2[item]
        sum1Sq=sum1Sq+pow(u1[item], 2)
        sum2Sq = sum2Sq + pow(u2[item], 2)
        pSum = pSum + (u1[item]*u2[item])
    num = pSum - ((sum1 * sum2 / n))
    den = sqrt((sum1Sq - pow(sum1, 2) / n) * (sum2Sq - pow(sum2, 2)
        / n))
    if den == 0: return 0
    r = num / den
   return r


count=0


for u in db.raasUsers.find(no_cursor_timeout=True):
    print(u['_id'])
    count+=1
    userRecord=db.raasUsers.find_one({"_id": ObjectId(u['_id'])})
    userRatedProductsList=[]


    similiarUsersIDs = similiarUserSet(userRecord)
    ratingdict1 = {}
    ratingdict2 = {}
```

```
    simUserParentRatingDict = {}
    for r in
        db.raasRatings.find({"user_Id":ObjectId(u['_id'])},no_cursor_timeout=True):
        ratingdict2[r['baseProductId']] = r['rating']
        userRatedProductsList.append(r['baseProductId'])


    similiaritySum=0
    similiaritySumDict={}
    ratingDictioanry={}
    for simUID in similiarUsersIDs:
        ratingdict1={}
        for r in db.raasRatings.find({"user_Id": ObjectId(simUID)}):
  ratingdict1[r['baseProductId']] = r['rating']
        similiarityScore=sim_Pearson(ratingdict1, ratingdict2)
        simUserChildRatingDict = {}
        if similiarityScore <= 0 or similiarityScore==1:
            continue
        print("similiarityScore",similiarityScore)
        similiaritySum+=similiarityScore
        localRating = db.raasRatings.find({"user_Id":
            ObjectId(simUID)},no_cursor_timeout=True)
        for r in localRating:
            if r['baseProductId'] not in userRatedProductsList:
                if r['baseProductId'] not in ratingDictioanry:
                    ratingDictioanry[r['baseProductId']]
                        =similiarityScore * r['rating']
```

```
                if r['baseProductId'] not in similiaritySumDict:
                    similiaritySumDict[r['baseProductId']] =
                        similiarityScore
                else:
                    similiaritySumDict[r['baseProductId']]
                        =similiaritySumDict[r['baseProductId']] +
                        similiarityScore
            else:
                ratingDictioanry[r['baseProductId']] =
                    ratingDictioanry[r['baseProductId']]+(similiarityScore
                    * r['rating'])
                if r['baseProductId'] not in similiaritySumDict:
                    similiaritySumDict[r['baseProductId']] =
                        similiarityScore
                else:
                    similiaritySumDict[r['baseProductId']]
                        =similiaritySumDict[r['baseProductId']] +
                        similiarityScore


    for k, v in ratingDictioanry.items():
        print(k,v)
        normalizedRating= ratingDictioanry[k]/similiaritySumDict[k]
        db.similiarUsers.insert_one({"user_Id": ObjectId(u['_id']),
            "product_Id": k, "rating": normalizedRating})
#End Of SimiliarUser Method
```

```
#Start Of SimiliarProduct Method

count=0
for u in db.raasUsers.find(no_cursor_timeout=True):

    count+=1
   userRecord = db.raasUsers.find_one({"_id":
       ObjectId(u['_id'])},no_cursor_timeout=True)
    userRatedProductsList=[]
    userUnRatedProductsList = []



    ratedProductsSum=0
ratedDict={}

    for r in
        db.raasRatings.find({"user_Id":ObjectId(userRecord['_id'])}):
        ratedDict[r['baseProductId']] = r['rating']
        userRatedProductsList.append(r['baseProductId'])



    similiaritySum = 0

    loopLimitCounter=0
    for unrated in db.raasBaseProducts.find(no_cursor_timeout=True):
```

```
        similiaritySum=0
        loopLimitCounter=loopLimitCounter+1
        if loopLimitCounter==50:
            break
        if unrated['_id'] not in userRatedProductsList:
            similiaritySum = 0
            #userUnRatedProductsList.append(unrated['_id'])
            ratedProductsMultiPly = 0
            for item in userRatedProductsList:
                simRecord=db.ProdComputedSim.find_one({ '$and': [ {
                    "product_Id": ObjectId(item) },{
                    "simProduct_Id": ObjectId(unrated['_id'])} ] })
                print(ObjectId(item))
                print( ObjectId(unrated['_id']))
                if simRecord is not None:
                    similiaritySum=similiaritySum+simRecord['simScore']
                    ratedProductsMultiPly+=simRecord['simScore']*
                  ratedDict[ObjectId(item)]
            print(unrated['_id'])
            if similiaritySum!=0:
                normalizedRating=ratedProductsMultiPly/similiaritySum
                db.similiarProducts.insert_one({"user_Id":
                    ObjectId(u['_id']), "product_Id":
                    unrated['_id'], "rating": normalizedRating})

#End Of SimiliarProduct Method
```

```
#Start Of Pre-Computed similiarity of products for
    similiarProducts Method
def sim_distance(u1,u2):

    plist1 = []
    plist2 = []
    si = {}
    for k, v in u1.items():
        plist1.append(k)
    for k, v in u2.items():
        plist2.append(k)

    for item in plist1:
        if item in plist2: si[item] = 1

    n = len(si)
    if n == 0: return 0
    sum_of_squares=0


    for key in u1:
             if key in u2:
                 sum_of_squares = sum_of_squares + sum(([pow
                     (u1[key] - u2[key], 2)]))
```

```
    return 1/(1+sum_of_squares)


count=0
dictParent={}
dictChild={}
print("start time:= ",datetime.datetime.now())
for pSuper in db.raasBaseProducts.find(no_cursor_timeout=True):
    dictParent = {}
    for rParent in db.raasRatings.find({"baseProductId":
        ObjectId(pSuper['_id'])},no_cursor_timeout=True):
        dictParent[rParent['user_Id']] = rParent['rating']



    for pSub in db.raasBaseProducts.find({ "_id": { '$ne':
        pSuper['_id'] } },no_cursor_timeout=True ):
        dictChild = {}
        for rChild in db.raasRatings.find({"baseProductId":
            ObjectId(pSub['_id'])},no_cursor_timeout=True):
            dictChild[rChild['user_Id']] = rChild['rating']
        similiarityScore = sim_distance(dictParent, dictChild)
        count = count + 1
        print("row count := ",count)
        if(similiarityScore>0):
            db.ProdComputedSim.insert_one({"product_Id":ObjectId(pSuper['_id']),
                "simProduct_Id":ObjectId(pSub['_id']), "simScore":
                similiarityScore})
```

```
        print(pSub['_id'])
        print(similiarityScore)

#End Of Pre-Computed similiarity of products
```

A.2 Source Code Online Recommendation

```
module.exports.getUserProducts=function (req, res,next) {
         var tags=req.body.tags;
         var distance=req.body.distance;
         var userId=req.body.userId;
         console.log(userId)
         //distance=50;
         distance*=1000;
         console.log(distance);
         var lat=req.body.lat;
         var lng=req.body.lng;
         var searchTags=[];
         var tagString=tags+'';
         console.log("tags string...",tagString);
         var splitTags=tagString.split(",");
         for (var x = 0; x < splitTags.length; x++) {
       searchTags.push(splitTags[x]);
```

```
}
    db.raasProducts.find({ "geoLoc" : { "$near" :{
        "$geometry" :{ "type" : "Point" ,"coordinates" :
        [lng,lat] } ,"$maxDistance":distance} }
              ,"isDelete":false },function(err,nearByProdList){
       var refinedContentList = [];
       var finalList=[]
   var refinedSimProdList = [];
   var refinedSimUserList = [];

       if(err || nearByProdList.length==0){
          console.log("Error");
         console.log(err);
          res.send(null);
      }
       else{
      db.raasProducts.find({ "geoLoc" : { "$near" :{
          "$geometry" :{ "type" : "Point" ,"coordinates" :
          [lng,lat] } ,"$maxDistance":distance} },
      "tags": { "$in":searchTags},"isDelete":false
          },function(err,allContentList){

      if(err ){
          console.log("Error");
         console.log(err);
      }
```

```
        else{
           console.log("refinedContentList.length=:
              ",refinedContentList.length);
           if(allContentList.length>0){
      for(var i = 0, l = allContentList.length; i < l; i++) {
      refinedContentList.push(allContentList[i]);
      if(refinedContentList.length==10){
         break;
      }}
              for (var i=0; i < refinedContentList.length; i++) {
                    finalList.push( refinedContentList[i] );
                    }

db.similiarProducts.find({user_Id:ObjectId(userId)}).sort({rating:-1},
   function(err,simProdList){
        if(err){
           res.json(finalList);
        }
        else{
        if(simProdList.length>0){ //If length greater than 0
            than loop otherwise useless
              var baseProductIDexp =null;
        for(var i = 0, l = simProdList.length-1; i < l; i++) {
              console.log("for simProdList
                 length",simProdList.length);
```

```
                console.log("refinedContentList
                    :=",refinedContentList.length);
                if(refinedSimProdList.length==10){
                   break;
                }


         baseProductIDexp = new RegExp('"' +
             simProdList[i].product_Id + '"');
    if(!baseProductIDexp.test(JSON.stringify(refinedContentList))
        &&baseProductIDexp.test(JSON.stringify(nearByProdList)))
       {
    var filteredData = _.find(nearByProdList, function(elt){
        return elt.baseProductId
        ==simProdList[i].product_Id.toString(); });
    refinedSimProdList.push(filteredData);}


    }
    for (var i=0; i < refinedSimProdList.length; i++) {
                    finalList.push( refinedSimProdList[i] );
                    } }
db.similiarUsers.find({user_Id:ObjectId(userId)}).sort({rating:-1},
function(err,simUserList){
      if(err){
         res.json(finalList);
      }
      else{
```

```
        if(simUserList.length>0){ //If length greater than 0
            than loop otherwise useless
            baseProductIDexp =null;
        for(var i = 0, l = simUserList.length; i < l; i++) {

           if(refinedSimUserList.length==10){
              break;
           }

  baseProductIDexp = new RegExp('"' +
      simUserList[i].product_Id + '"');
         console.log(simUserList[i].product_Id);
if(!baseProductIDexp.test(JSON.stringify(finalList)) &&
    baseProductIDexp.test(JSON.stringify(nearByProdList)))
{

   var filteredData = _.find(nearByProdList, function(elt){
    return elt.baseProductId
       ==simUserList[i].product_Id.toString(); });
   refinedSimUserList.push(filteredData);

}}}
      for (var i=0; i < refinedSimUserList.length; i++) {
                finalList.push( refinedSimUserList[i] );
                }
```

```
if(finalList.length==10){
        for(var i = 10, l =20; i < l; i++) {
  finalList.push(allContentList[i]);
        }}
     if(finalList.length==20){
        for(var i = 20, l =30; i < l; i++) {
  finalList.push(allContentList[i]);
        }}
     res.json(finalList);}})}


        })
 }
    else{res.json(null);  }
     }})} })}
```

Appendix B

DataSet Detail

We have generated three tables after the processing of dataset of super-store sales to evaluate the proposed recommender system [35].These table are Users,Products and Ratings. Figure B.1 depicts an abstract overview of the whole story. The mentioned dataset was in excel sheet format. To Process dataset we have exploited visual studio IDE and C# language. We have utilized *Microsoft.Office.Interop.Excel* library provided by Microsoft to read data from excel sheet.

To commincate with mongoDB from visual studio environment we have used two libraries *a) MongoDB.Bson b) MongoDB.Driver* provided by MongoDB ecosystem. We have used the genderize API [44] to determine the gender of the customer. This API [44] takes the first name of user and return the gender against that name. To process JSON response by genderize API [44] we have utilized Newtonsoft.Json library.We have downloaded image of the products using google custom search API [45]. We have sent the product name as query to API [45] and it returned the image of the product. We

have uploaded the product image on amazon S3 and update image URL in product table accordingly. To call Google API from C# code we have used Google.Apis.Customsearch.v1 library and to upload image upon S3 bucket we have exploited Amazon.S3.Transfer library. To calculate the customer buying power we have used two column namely Unit Price and Order Quantity. We have multplied the values of these columns and store data of row having maximum value against each customer into MongoDb raasUsers Table.

The benefit of processing sale dataset is to prepare dataset which can be easily consumed by recommendation engine. Another major benefits of processing is to shape dataset according to the local market of Pakistan. The primary advantage of processing dataset is that MongoDB is much better chocie as compared to excel database to build recommender systems.

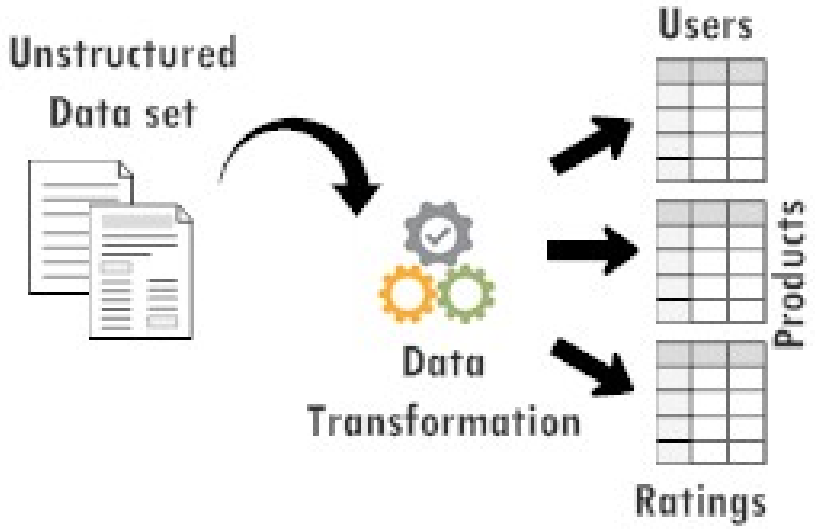

Figure B.1: Overview Of Dataset Processing

B.1 Sample of Superstore Sales Data Set

Overall Dataset contains 8400 rows and 21 columns. We have used just seven cloumns from the provided data set already discussed in *Data Set and Use Cases for Evaluation section* of Chapter 5. Table B.1 describes the sample subset of sales [35] dataset. As mentioned before that Sales dataset is in excel sheet, we have just list down a sample data against only those seven columns which have been exploited to generate datasets for our proposed system.

Customer Name	**Product Name**	**Product Category**	**Product Sub-Category**	**Unit Price**	**Order Quatity**	**Region**
Muhammed MacIntyre	Eldon Base for stackable storage shelf, platinum	Office Supplies	Storage & Organization	38.94	6	Nunavut
Barry French	1.7 Cubic Foot Compact Cube Office Refrigerators	Office Supplies	Appliances	208.16	49	Nunavut
Allen Golden	Binney And Smith Crayola Metallic Colored Pencils, 8-Color Set	Technology	Pens And Art Supplies	4.3	27	Ontario
Robert Barroso	Hon 4700 Series Mobuis Mid-Back Task Chairs with Adjustable Arms	Furniture	Chairs And Chairmats	355.98	30	Atlantic
Laurel Elliston	Xerox 1968	Office Supplies	Papers	6.68	19	Quebec

Table B.1: Sample Sales Data Set

B.2 Source Code to Process Dataset

Here is the source code which takse superstore sales dataset [35] as input and generate three tables Users,Products and Ratings as output .

```
public static void processData()
 {
     // Utility Method To Store Excel Data Into MongoDB
     // Authored By : Aftab Khan
    xlApp = new Excel.Application();
     xlWorkBook = xlApp.Workbooks.Open(@"D:\Sales.xls", 0,
         true, 5, "", "", true,Excel.XlPlatform.xlWindows,
         "\t", false, false, 0, true, 1, 0);
     xlWorkSheet =
         (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
     List<string> usersName = new List<string>();
     Dictionary<string, string> userCitiesDict = new
         Dictionary<string, string>();
     for (rCnt = 2; rCnt <= rw; rCnt++)
     {
         if (range.Cells[rCnt, 12] != null &&
             range.Cells[rCnt, 12].Value2 != null)
         {
             userName = (string)(range.Cells[rCnt, 12] as
                 Excel.Range).Text;
             if (!usersName.Contains(userName.Trim()))
```

```
                {
                    usersName.Add(userName.Trim());
                }
                if (!userCitiesDict.ContainsKey(userName.Trim()))
                {
                    userCitiesDict.Add(userName.Trim(),
                        productCity.Trim());
                }
            }
        }

        //Call InsertUser Method to Fill Users Table
    insertUsers(userCitiesDict);

      //Code to insert Product One By One
     for (rCnt = 2; rCnt <= rw; rCnt++)
        {
            if (range.Cells[rCnt, 12] != null &&
                range.Cells[rCnt, 12].Value2 != null)
            {
         price = (string)(range.Cells[rCnt, 10] as
             Excel.Range).Text;
                productCity = (string)(range.Cells[rCnt, 13] as
                    Excel.Range).Text;
                productRegion = (string)(range.Cells[rCnt, 14]
                    as Excel.Range).Text;
```

```
                productCat = (string)(range.Cells[rCnt, 16] as
                    Excel.Range).Text;
                productSubCat = (string)(range.Cells[rCnt, 17]
                    as Excel.Range).Text;
                productName = (string)(range.Cells[rCnt, 18] as
                    Excel.Range).Text;
                //Insert Product Into Mongo DB
                insertProdut(userName, productName, productCat,
                    productSubCat, productCity, productRegion,
                    productPrice);
            }
        }

    }

// Method to Insert Users
 private static void insertUsers(Dictionary<string, string>
    userCitiesDict)
    {
          foreach (KeyValuePair<string, string> entry in
              userCitiesDict)
        {
            userName = entry.Key;
            mappingResult resultSet =
                mappedResultSet(entry.Value);
            string[] splittedName = userName.Split(' ');
```

```
Gender = GetGender(splittedName[0]);//Gender API Call
Random rnd = new Random();
ageGroupIndex = rnd.Next(1, 5);
lat = resultSet.Latitude;
lng = resultSet.Longitude;
PlainAddress = resultSet.City;
switch (ageGroupIndex)
{
    case 1:
        ageGroupDecription = AgeRanges.range18_25;
        break;
    case 2:
        ageGroupDecription = AgeRanges.range26_35;
        break;
    case 3:
        ageGroupDecription = AgeRanges.range36_45;
        break;
}

MongoClient client = new MongoClient();
var documnt = new BsonDocument
       {
          {"userName",userName},
          {"geoAddress", GeoJson.Point(new
              GeoJson2DCoordinates(lat,lng)).ToBsonDocument()},
           {"plainAddress",PlainAddress},
```

```
                        {"emailAddress",EmailAddress},
                        {"password",password},
                         {"type",type},
                         {"ageIndex",ageGroupIndex},
                          {"ageGroupDecription",ageGroupDecription},
                          {"gender",Gender},
                          {"genderAccuracy",genderAccuracy.AccuracyProbabilty},
                          { "tags", new BsonArray {}},
                          {
                              "prefferedDistance",PrefferedDistance},
                          { "discovery",discovery},
                          { "isDelete",isDelete}
 };
              collection.InsertOne(documnt);

              //End Foreach Loop
          }
          //End Insertion

      }

//Method to Insert Product into MongoDB
  private static void insertProdut(string userName, string
      productName, string productCat, string ProductSubCat, string
      productCity, string productRegion, double productPrice)
        {
```

```
mappingResult resultSet =
    mappedResultSet(productCity,productRegion);


 MongoClient client = new MongoClient();
 var db = client.GetDatabase("mydb");
//Insert Product
 var documnt = new BsonDocument
                        { {"productName",productName},
                              {"geoLoc", GeoJson.Point(new
                              GeoJson2DCoordinates
                               (resultSet.Latitude,
                               resultSet.Longitude)).
                              ToBsonDocument()},
                              { "tags", new BsonArray
                              {productCat,ProductSubCat}},
                              {"city",resultSet.City},
                               {"productCat",productCat},
                               {"productSubCat",ProductSubCat},
                              {"price",productPrice},
                              {"imageUrl",imageUrl}};
  productcollection.InsertOne(documnt);
 var product_Id = documnt.GetElement("_id").Value;
 //update User Preffered Tags Values
 var updateUserTagsArray =
     Builders<BsonDocument>.Update.AddToSet( "tags", new
     BsonArray {productCat,ProductSubCat});
```

```
            var resultPush =
                usercollection.UpdateOne(userNamefilter,
                updateUserTagsArray);

            //Insert ratings
            var documntRatings = new BsonDocument
                                    {
                                      {"user_Id",user_ObjectId},
                                      {"product_Id",product_Id},
                                      {"rating",rating},
                                      {"isComputedRating","N"},

          };
            userProductscollection.InsertOne(documntRatings);
            //End Product Insertion
        }
```

B.3 Sample Of Processed Dataset

Here is sample records from all three tables generated after processing of superstore Sales dataset [35].

B.3.1 Sample Records from Users Table


```
1 {"_id":{"$oid":"58a56e803ad3bd233ca55e03"},"userName":"Jack
    Garza","geoAddress":{"type":"Point","coordinates":[33.72938
```

```
8,73.093146]},"plainAddress":"Islamabad","emailAddress":"
JackGarza@raasapp.testdsmscs.nust.seecs.edu.pk","password":
"abcd1234","type":"U","ageIndex":1,"ageGroupDecription":"18
-25","gender":"M","genderAccuracy":0.99,"prefferedDistance"
:20.0,"discovery":true,"isDelete":false,"tags":["Technology
","Computer Peripherals","Office Supplies","Pens And Art
Supplies","Furniture","Office Furnishings","Rubber Bands","
Telephones and Communication","Tables","Binders and Binder
Accessories"]}
{"_id":{"$oid":"58a56e813ad3bd233ca55e04"},"userName":"Julia
West","geoAddress":{"type":"Point","coordinates":[33.729388
,73.093146]},"plainAddress":"Islamabad","emailAddress":"
JuliaWest@raasapp.testdsmscs.nust.seecs.edu.pk","password":
"abcd1234","type":"U","ageIndex":4,"ageGroupDecription":"46
-60","gender":"F","genderAccuracy":0.99,"prefferedDistance"
:20.0,"discovery":true,"isDelete":false,"tags":["Furniture"
,"Chairs And Chairmats","Office Furnishings","Bookcases"]}
{"_id":{"$oid":"58a56e813ad3bd233ca55e05"},"userName":"Eugene
Barchas","geoAddress":{"type":"Point","coordinates":[33.729
388,73.093146]},"plainAddress":"Islamabad","emailAddress":"
EugeneBarchas@raasapp.testdsmscs.nust.seecs.edu.pk","
password":"abcd1234","type":"U","ageIndex":3,"
ageGroupDecription":"36-45","gender":"M","genderAccuracy":0
.98,"prefferedDistance":20.0,"discovery":true,"isDelete":
```

```
false,"tags":["Office Supplies","Binders and Binder
Accessories","Furniture","Chairs And Chairmats","Paper","
Technology","Telephones and Communication","Office Machines
","Computer Peripherals"]}
{"_id":{"$oid":"58a56e823ad3bd233ca55e06"},"userName":"Edward
Hooks","geoAddress":{"type":"Point","coordinates":[33.72938
8,73.093146]},"plainAddress":"Islamabad","emailAddress":"
EdwardHooks@raasapp.testdsmscs.nust.seecs.edu.pk","password
":"abcd1234","type":"U","ageIndex":4,"ageGroupDecription":"
46-60","gender":"M","genderAccuracy":1.0,"prefferedDistance
":20.0,"discovery":true,"isDelete":false,"tags":["Furniture
","Chairs And Chairmats","Office Supplies","Binders and
Binder Accessories","Technology","Office Machines","Paper",
"Tables"]}
{"_id":{"$oid":"58a56e823ad3bd233ca55e07"},"userName":"Brad
Eason","geoAddress":{"type":"Point","coordinates":[33.72938
8,73.093146]},"plainAddress":"Islamabad","emailAddress":"
BradEason@raasapp.testdsmscs.nust.seecs.edu.pk","password":
"abcd1234","type":"U","ageIndex":4,"ageGroupDecription":"46
-60","gender":"M","genderAccuracy":0.99,"prefferedDistance"
:20.0,"discovery":true,"isDelete":false,"tags":["Office
Supplies","Binders and Binder Accessories","Furniture","
Office Furnishings","Labels","Storage And Organization","
Paper","Bookcases","Pens And Art Supplies"]}
```

```
6 {"_id":{"$oid":"58a56e833ad3bd233ca55e08"},"userName":"Nicole
   Hansen","geoAddress":{"type":"Point","coordinates":[33.7293
   88,73.093146]},"plainAddress":"Islamabad","emailAddress":"
   NicoleHansen@raasapp.testdsmscs.nust.seecs.edu.pk","
   password":"abcd1234","type":"U","ageIndex":2,"
   ageGroupDecription":"26-35","gender":"F","genderAccuracy":1
   .0,"prefferedDistance":20.0,"discovery":true,"isDelete":
   false,"tags":["Office Supplies","Paper","Storage And
   Organization","Furniture","Chairs And Chairmats","Tables","
   Pens And Art Supplies","Appliances","Office Furnishings"]}
```


B.3.2 Sample Records from Products Table


```
1 {"_id":{"$oid":"58a585843ad3bd22cc502a05"},"productName":"Epson
    DFX5000+ Dot Matrix Printer","geoLoc":{"type":"Point","
   coordinates":[73.055522,33.720618]},"tags":["Technology","
   Office Machines"],"city":"Islamabad","productCat":"
   Technology","productSubCat":"Office Machines","price":1500.
   97,"imageUrl":"https://s3.amazonaws.com/uploadedpictures/63
   6229341295101308.Jpeg","baseProductId":{"$oid":"58af13763ad
   3bd5548204043"},"isDelete":false}
2 {"_id":{"$oid":"58a585843ad3bd22cc502a07"},"productName":"Hon 2
   090Pillow Soft Series Mid Back Swivel/Tilt Chairs","geoLoc"
   :{"type":"Point","coordinates":[73.055522,33.720618]},"tags
```

```
":["Furniture","Chairs And Chairmats"],"city":"Islamabad","
productCat":"Furniture","productSubCat":"Chairs And
Chairmats","price":280.98,"imageUrl":"https://s3.amazonaws.
com/uploadedpictures/636229525409714905.Jpeg","
baseProductId":{"$oid":"58af137b3ad3bd554820419a"},"
isDelete":false}
{"_id":{"$oid":"58a585843ad3bd22cc502a09"},"productName":"
Hewlett-Packard cp1700 [D, PS] Series Color Inkjet Printers
","geoLoc":{"type":"Point","coordinates":[73.055522,33.7206
18]},"tags":["Technology","Office Machines"],"city":"
Islamabad","productCat":"Technology","productSubCat":"
Office Machines","price":500.98,"imageUrl":"https://s3.
amazonaws.com/uploadedpictures/636229589702545009.Jpeg","
baseProductId":{"$oid":"58af137f3ad3bd5548204286"},"
isDelete":false}
{"_id":{"$oid":"58a585853ad3bd22cc502a0b"},"productName":"8890"
,"geoLoc":{"type":"Point","coordinates":[73.055522,33.72061
8]},"tags":["Technology","Telephones and Communication"],"
city":"Islamabad","productCat":"Technology","productSubCat"
:"Telephones and Communication","price":115.99,"imageUrl":"
https://s3.amazonaws.com/uploadedpictures/63622958915562036
2.Jpeg","baseProductId":{"$oid":"58af137f3ad3bd554820426f"}
,"isDelete":false}
```

```
5 {"_id":{"$oid":"58a585853ad3bd22cc502a0d"},"productName":"
   Tennsco Snap-Together Open Shelving Units, Starter Sets and
    Add-On Units","geoLoc":{"type":"Point","coordinates":[73.0
   55522,33.720618]},"tags":["Office Supplies","Storage And
   Organization"],"city":"Islamabad","productCat":"Office
   Supplies","productSubCat":"Storage And Organization","price
   ":279.48,"imageUrl":"https://s3.amazonaws.com/
   uploadedpictures/636228824577573874.Jpeg","baseProductId":{
   "$oid":"58af13743ad3bd5548203fcb"},"isDelete":false}
6 {"_id":{"$oid":"58a585853ad3bd22cc502a0f"},"productName":"
   Gyration Ultra Cordless Optical Suite","geoLoc":{"type":"
   Point","coordinates":[73.055522,33.720618]},"tags":["
   Technology","Computer Peripherals"],"city":"Islamabad","
   productCat":"Technology","productSubCat":"Computer
   Peripherals","price":100.97,"imageUrl":"https://s3.
   amazonaws.com/uploadedpictures/636229668410803810.Jpeg","
   baseProductId":{"$oid":"58af13803ad3bd55482042b2"},"
   isDelete":false}
```


B.3.3 Sample Records from Ratings Table


```
1 {"_id":{"$oid":"58a415153ad3bd15a0f0549f"},"user_Id":{"$oid":"5
   8a3f92e3ad3bd2fc4ab6f37"},"product_Id":{"$oid":"58a414fa3ad
   3bd15a0f0549e"},"rating":5,"isComputedRating":"N"}
```

```
{"_id":{"$oid":"58a415153ad3bd15a0f054a1"},"user_Id":{"$oid":"5
   8a3f9ef3ad3bd2fc4ab6f38"},"product_Id":{"$oid":"58a415153ad
   3bd15a0f054a0"},"rating":2,"isComputedRating":"N"}
{"_id":{"$oid":"58a415153ad3bd15a0f054a3"},"user_Id":{"$oid":"5
   8a3f9ef3ad3bd2fc4ab6f38"},"product_Id":{"$oid":"58a415153ad
   3bd15a0f054a2"},"rating":5,"isComputedRating":"N"}
{"_id":{"$oid":"58a415163ad3bd15a0f054a5"},"user_Id":{"$oid":"5
   8a3f9f03ad3bd2fc4ab6f39"},"product_Id":{"$oid":"58a415163ad
   3bd15a0f054a4"},"rating":4,"isComputedRating":"N"}
{"_id":{"$oid":"58a415163ad3bd15a0f054a7"},"user_Id":{"$oid":"5
   8a3f9f03ad3bd2fc4ab6f3a"},"product_Id":{"$oid":"58a415163ad
   3bd15a0f054a6"},"rating":5,"isComputedRating":"N"}
```