Chapter 12
A Mobile Cloud Framework for Context-Aware and Portable Recommender System for Smart Markets

Aftab Khan, Aakash Ahmad, Anis Ur Rahman, and Adel Alkhalil

12.1 Introduction

Smart city systems are an emerging trend that utilize information and communication technologies (ICTs) to offer improved urban services to individuals and collectively refining the lifestyle of societies [11]. In recent years, research and practices have intended to transform the traditional cities and societies into technology and knowledge-driven twenty-first century metropolis [4, 10]. In the context of smart city systems, mobile computing has emerged as a pervasive technology that has empowered its users—with mobility and context-awareness—to accomplish a range of tasks including portable computation as well as location-aware communication [31, 33]. Mobile computing provides the users with mobility-driven and context-aware interfaces to select and utilize the available services including but not limited to smart health, transportation, business, and socialization offered by smart city systems [37].

A. Khan
School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad, Pakistan
e-mail: 13mscsmaftab@seecs.edu.pk

A. Ahmad (✉) · A. Alkhalil
College of Computer Science and Engineering, University of Ha'il, Ha'il, Saudi Arabia
e-mail: a.abbasi@uoh.edu.sa; a.alkalel@uoh.edu.sa

A. U. Rahman
School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad, Pakistan

Department of Information System, Faculty of Computer Science and Information Technology, University of Malaya, Malaysia
e-mail: anis.rahman@seecs.edu.pk

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_12

Mobility or portability is regarded as one of the central features of mobile computing that also provides the foundation for context-aware computing [37]. However, mobility also enforces resource constraints such as limited hardware that affects computation, storage, and energy-related tasks on mobile devices. There is a need for solutions that maintain the balance between mobility and resource availability in the context of mobile computing and smart city systems [2]. In contrast to mobile computing, cloud computing model exploits the "pay-per-use" services model to provide virtually unlimited processing and storage resources [24]. Cloud computing offers the entities or organizations to off-load or deploy their (on-premise) software systems, computations, or storage resources to remote servers by means of cloud-based services [17]. For example, the research in [22] highlights an approach known as cyber-foraging that off loads the computation/storage intensive tasks from a mobile device to (cloud-based) servers in order to enhance computation and energy efficiency of mobile devices. In the context of resource-constrained mobile computing, resource-sufficient cloud computing can be viewed as an opportunistic model that allows mobile devices to compensate their resource poverty by offloading mobile data and computation to cloud servers [34]. Therefore, the unification of mobile and cloud computing can benefit from the mobility and context-awareness of mobile computing, and the computation/storage services of cloud computing to provide systems that are portable and resource sufficient [21].

Research Context In the past, electronic commerce (e-commerce) systems have proven to be useful by digitally offering products and services to international marketplaces. In the current era, business systems/entities heavily rely on reaching their potential customers by recommending them highly customized products and offers. Now with mobile commerce (m-commerce), the use of context information such as age, gender, preferences, and location to offer customer recommendations has gained a significant attention [39]. For example, based on the users' contextual information, any software that provides recommendations such as socialization activities, product and service offerings, and dining options enables human decision support and gives rise to smart systems. Considering a wide-spread adoption of the mobile and cloud computing, there is still a lack of solutions that facilitate its users with a recommendation of their preferences primarily based on their localized context. One of the main challenges for managing and exploiting context-aware recommendations is to identify the contextual factors (such as location and preferences) that influence decisions and actions of people in smart city context [5].

Recommender systems represent a class of software systems that generate meaningful recommendations of interests for their users and empower the users with decision support [28, 32]. Context-aware recommender systems provide dynamic adaptive recommendations to users based on contextual information such as the location, gender, and other preferences of the user [36]. In a smart city context, *smart markets* refer to electronic (virtualized) marketplaces that exploit ICT technologies and infrastructure to enable or enhance digitized commerce. Recommendations and matchmaking between potential customers and business entities are enabled by the relevancy of contextual information that gives rise to the concept of smart markets. Such markets offer personalized offers, products, services, and delivery by business entities to potential customers while minimizing the irrelevant mass publicity. For instance, by calculating user's location, age, and other relevant information, a mobile recommender system acts as a context-sensing and portable computer that can notify the user on the go about his/her events or places of interests. Recently, much research and development is being carried out to support context-aware recommender systems. However, existing mobile recommender systems fall short of context-aware recommendations in smart city systems to support the activities of smart markets [27, 32, 43].

Solution Overview We overview the proposed recommender system based on the illustrations in Fig. 12.1 that also highlights the activities of smart markets. The proposed framework has two layers, namely: (1) *front-end mobile computing layer* and (2) *back-end cloud computing layer*. By acting as the front-end layer of the framework, a mobile device plays two distinct roles that include (1) sensing the user's context (i.e., location, age, and preferences) and (2) providing an interface to display context-aware recommendations. Cloud computing acts as the back-end layer to compensate for limited resources of a mobile device by providing storage

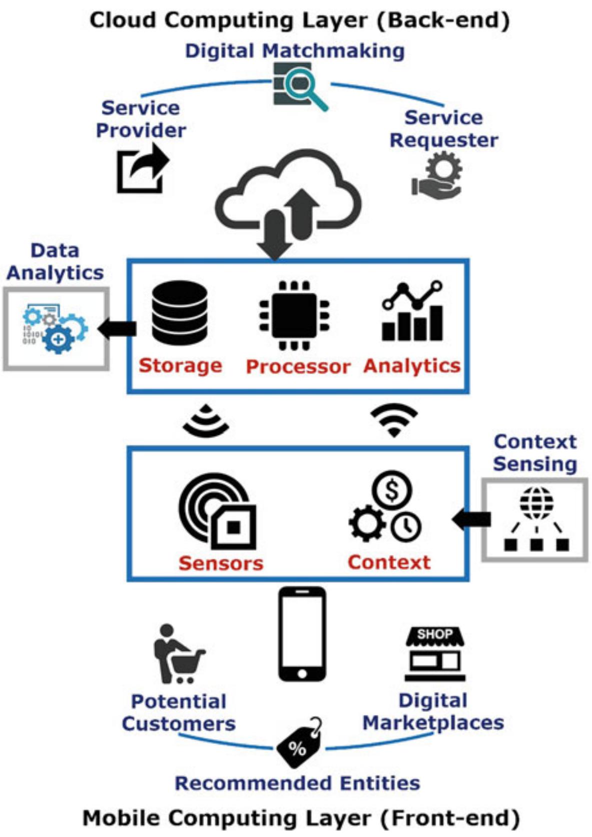


Fig. 12.1 An overview of the proposed mobile cloud recommender system

and computation resources to produce recommendations that are communicated to the mobile device. The proposed solution[1] allows markets, businesses, and transaction-driven entities to communicate with their potential customers in a smart way, i.e., to provide recommendations to the customers based on their localized context of location and preferences. The primary challenge to generate context-aware recommendations lies with the identification of contextual elements such as user's location and preferences from different sources. Another research challenge for recommendation systems is to yield recommendations in real-time fashion for a given user from large and diverse dataset(s) of persons' past preferences. In doing so, there is a need to efficiently utilize the resource-constrained mobile devices that can execute the computation and energy-intensive tasks efficiently.

Proposed Contributions A recent survey on the existing mobile recommender systems highlights that current solutions fall short of context-aware recommendations in a smart market domain [32]. The proposed solution introduces a system architecture, algorithms, enabling technologies, and a prototype for mobile-cloud-based context-aware recommender system. We outline the primary contributions of the proposed solution as:

– Unification of the mobile and cloud computing technologies to provide a framework that supports users' decision support in smart city systems. The framework empowers its users with mobility and context-awareness while processing complex recommendations accurately and efficiently.
– Algorithms and prototype that support automation and user-based customization to provide portable and context-aware matchmaking between potential customers and business entities in smart markets.
– Exploiting mobile cloud computing as state-of-the-art mobile computing technology to alleviate the resource poverty of mobile devices by means of cloud-based resources. The solution supports a class of recommender systems that are context-aware, portable, accurate, and resource efficient.

Section 12.2 presents background details and the related research. Section 12.3 presents the proposed framework and its architecture. Section 12.4 presents the algorithms and tools to implement the framework. Section 12.5 presents the framework evaluation. Section 12.6 presents conclusions and future research.

12.2 Background and Related Research

First, we present the background details about the different types of recommender systems in Sect. 12.2.1. We then discuss the existing research on context-aware recommender systems in Sect. 12.2.2, e-type software recommender systems in

[1]Please note that we use the terms *proposed solution*, *proposed system*, and *proposed framework* interchangeably, all referring to the same concept.

Sect. 12.2.3, and cloud-based recommender systems in Sect. 12.2.4. A discussion of the different types of recommender systems and their state-of-the-art research helps us to justify the scope and needs for the proposed recommender system. The concepts and terminologies used in this section are utilized throughout the paper.

12.2.1 Types of Recommender Systems

The recommender systems can generally be categorized into five distinct types as illustrated in Fig. 12.2 that includes:

1. **Content-based recommender systems** recommend items to user according to user' preferences and past history [12].
2. **Collaborative filtering recommender systems** recommend the items to user based on the collaborative preferences that are gathered from a diverse set of users [44].
3. **Demographic recommender systems** recommend items to users on the basis of user's personal profile of demography [3].
4. **Knowledge-based recommender systems** recommend items according to either inferences regarding users' taste or particular domain knowledge. These types

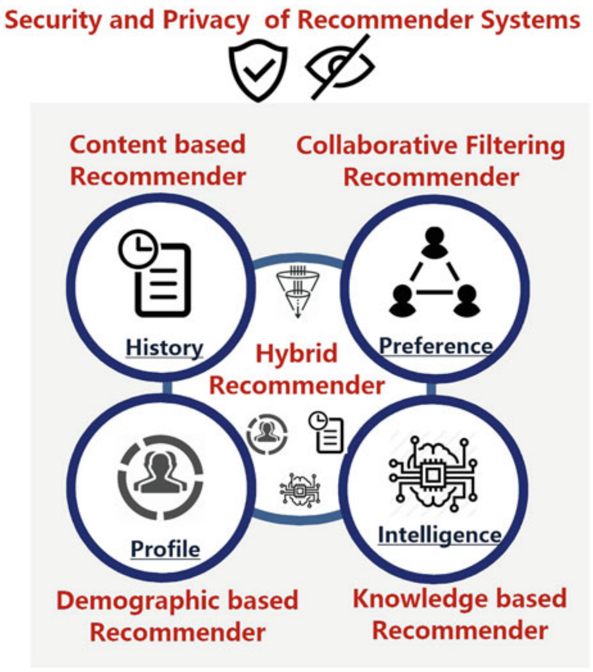

Security and Privacy of Recommender Systems
Content based Recommender
Collaborative Filtering Recommender
History
Preference
Hybrid Recommender
Profile
Intelligence
Demographic based Recommender
Knowledge based Recommender

Fig. 12.2 An overview of the types of recommender systems

of recommender systems exploit past knowledge about how items of potential recommendation fit better according to preferences of the user [8].

5. **Hybrid recommender systems** are based on the intersection of the above mentioned approaches to provide recommendations [38].

In addition to the types of recommender systems mentioned above, we also discuss four major categories or the real world domains where different types of recommender systems have been applied [32]. We discuss the recommender systems in the context of mobile and e-type systems detailed below.

12.2.2 Context-Aware Recommender Systems

Context-aware recommender systems present items to the user(s) according to his/her taste and preferences as well as considering the contextual factors such as location, mood, weather, and day or time.

- *Mobile Recommender Systems for Places of Interest:* In recent years, there have been many efforts to develop recommender systems that can operate in different domains such as tourism, leisure, e-commerce, and mobile-commerce recommendations. For example, Braunhofer et al. [6] have developed a context-aware mobile application STS that suggests user's places of interests by using their mood, weather conditions, and personality traits. They used five factor model [15] along with user past rating to discover user personality traits. One of the primary challenge in developing multi-user mobile information systems lies with the scalability of the solution. By keeping scalability issues in mind, Roberts et al. [29] have designed a high-performance mobile recommender system (Magitti) which is scalable to many users operating at the same time. The system proposes a technique for recommending leisure time activities based on what time of week it is and venues nearby to the users' location by combining multiple recommendation patterns using predefined rules. A three-tier client–server architecture approach has been used to implement 3D based context-aware system [27]. 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 keeps track of user's location and speed via GPS/compass and communication is achieved through binary request–response protocol.
- *Knowledge-Based Recommender System for Movies:* Currently, recommender systems are making use of semantic web technologies to address the challenges of data sources diversity and information overload. For example, a recommendation system named as RecomMetz that recommends movie show times is presented in [8]. In RecomMetz, three different types of contextual factors are studied: *(a) location, (b) crowd, and (c) time* to produce recommendations. The proposed architecture of RecomMetz is based on modules, such as user interface, user check-in subsystem, data repository, context-aware subsystem, and recommendation engine.

Security and Privacy of Recommender Systems Typically, context-aware recommender systems have some privacy issues, such as the right(s) of users to know how, when, and under which circumstances their location as well as identity and other personal information can be accessible to other users or services. To support the privacy preserving mobile recommender systems, a solution named PRECISE [40] has been developed using cloud architecture. The PRECISE allows users to define privacy preserving policies while availing-off recommendation services.

12.2.3 Recommender Systems for E-Type Software

The e-type software refers to the systems such as e-health and e-commerce that exploit the ICT technologies to automate the manual and laborious tasks efficiently. Current recommendation techniques cannot be fully applied to e-type systems. To address this issue, Yang et al. [43] proposed a location-aware recommender system named PR (personal recommender) that fulfills customers' shopping demands with location-based seller offers and publicities. In this solution, on client side there are two components (a) web browser and (b) location manager. Customer requests include its GPS location, while the server side system maintains database for customer history and database having customer profile. 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.

– *Recommendation System for E-Health:* Services related to health 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 online information regarding medical services available for public use. People make use of this information for personal health care management or patient-specific decision making [42].

  The information pertinent to the patients is generally distributed across a huge number of different web sites, so it is hard for patients to explore authentic health care information from large volume of data. It has been found that young people preferred to use mobile devices to download or browse health information [16]. Moreover, Wang et al. [41] have proposed a framework to develop a recommendation service that facilitates users to get relevant health information on mobile devices [35].

12.2.4 Cloud-Based Recommender Systems

Due to the limited battery and computational resources of mobile devices, cloud services provide a great alternative to software services that are configured and executed on the mobile. By taking into consideration the cloud computing based

Table 12.1 A comparison of the relevant existing solutions of mobile recommender systems

Application domain	Type of recommender	Context information	Mobile computing	Cloud computing	Mobile cloud computing	Solution reference
E-commerce	Content based	Location	✓	×	×	[43]
Leisure activities (music)	Content based, collaborative filtering	Weather, location, companion	✓	×	×	[14]
Leisure activities (media items)	Knowledge based, collaborative filtering	Location, time, day	✓	✓	✓	[26]
Leisure activities (movies)	Knowledge based, collaborative filtering	Location, crowd, time	✓	×	×	[8]
Tourism	Collaborative filtering	Location, time	✓	✓	✓	[20]
E-health	Collaborative filtering	User preferences, physiological data	✓	✓	✓	[41]
Tourism	Knowledge based, collaborative filtering	Location	✓	×	×	[25]

solutions, Otebolaku and Andrade [26] have proposed a context-aware recommendation system to recommend relevant cloud-based media particulars to mobile users. The context recognition service hosted inside the cloud is responsible for monitoring, learning, and predicting users' context. To retrieve user context, WiFi, GPS, accelerometer, rotation as well as orientation vector sensors have been used. In this solution, the nearest neighbor (KNN) algorithm has been used to predict the location and activity of the users. Moreover, a web-based recommender system named REJA is developed to address drawbacks of mobile recommender systems [25]. The current approaches detailed above do not provide an optimal solution for the problem of group recommendations as well as cold start and data sparseness problems. To overcome these issues, Khalid et al. [20] have implemented a cloud-based solution named 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 the mixture of ant colony algorithms with social filtering technique to retrieve the most favorable location recommendations based on real-time context such as traffic and weather conditions.

Comparative Summary of Existing Solutions In Table 12.1, we present a comparison-based summary of the existing solutions. For an objective comparison and interpretation of the results, we compare the existing solutions based on six distinct criteria presented in Table 12.1. For example, to interpret the data in Table 12.1, we can summarize that [43] presents a solution that exploits the context-based recommender techniques and uses location as a context to support mobile recommender system for e-commerce activities. We conclude that the solutions for mobile recommender systems have progressed and matured over time. However, there is a need for innovative solutions that address mobile commerce in general and smart markets in particular. In addition, the emerging solutions need to exploit mobile cloud computing as state-of-the-art for mobile computing technology. Mobile cloud computing supports context-aware and mobility-driven recommendations while also maintaining the scalability and elasticity of computational resources.

12.3 Architecture of the Recommendation Framework

In this section, we present the architecture of the proposed framework and its underlying layers that represents a higher-level view and blueprint of the overall system. Based on the presented architecture, we discuss the implementation specific details for the framework later in the paper.

12.3.1 Architecture and Patterns for the Framework

As per the ISO/IEC/IEEE 42010 standard,[2] architecture of a software intensive systems represents a high-level view of the systems in terms of system components (e.g., computational elements and data stores) and connectors that enable component communication. We follow software architecture-based development of the proposed framework for two reasons detailed below.

1. **System abstraction and quality:** Software architecture abstracts the complex and implementation specific details of the system with higher-level software components and connectors. Software components and connectors help with designing and reasoning about the system functionality and quality prior to its implementation [23].
2. **Reusability of design:** Software architecture patterns can be exploited as proven best practices that support reusability of components and structures in architecture-based development and enhance the quality of the software [7].

[2]ISO/IEC/IEEE 42010 Systems and software engineering—Architecture description is an international standard for architecture descriptions of systems and software.

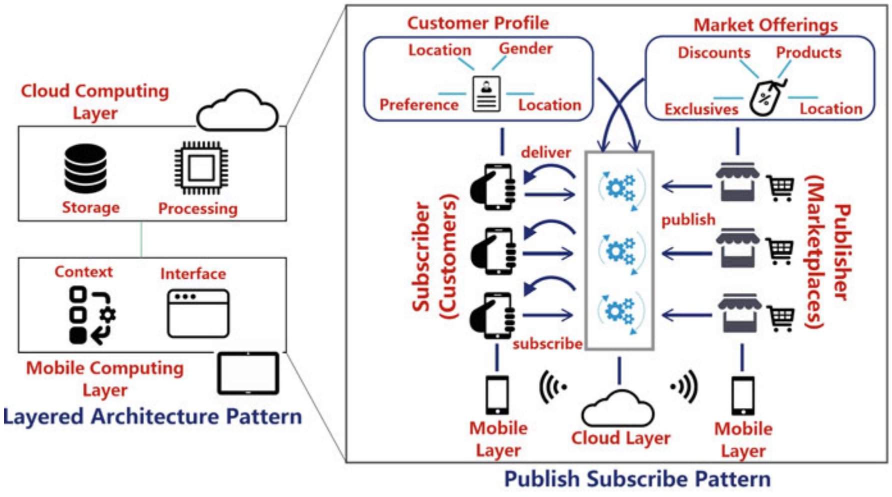


Fig. 12.3 Pattern-based software architecture for recommender system

We present the architecture of the proposed framework in Fig. 12.3. Specifically, we present the software architecture view in terms of two architectural patterns, namely: *layered architecture pattern*, and *publish–subscribe pattern* as illustrated in Fig. 12.3. The layered architecture pattern has helped us to maintain the separation of concerns in terms of mobile and cloud computing layers to engineer and develop the recommender system[18]. In addition, we have also applied the publish–subscribe pattern that helps to maintain the relationship between potential customers and business entities as the requesters and the providers of the product specific contextual information [13]. Based on the illustration in Fig. 12.3, we detail the application of the layered pattern to the proposed software architecture as below.

12.3.2 *Context-Aware Mobile Computing Layer*

The mobile computing layer as the front-end of the system exploits portable and context-aware mobile devices that provides an interactive interface to the potential customers and the market entities to communicate with the system [2, 5]. Using the mobile computing layer the potential customers can specify their preferences such as interest in available discounts, consumer products, and services to enable the matchmaking between potential customers and the market. In addition to the user input and decision support, the mobile device dynamically calculates the contextual information such as user's geographical location along with market offerings to recommend the products/services of the customers' interest.

Mobility of mobile computing inherits a few challenges such as resource poverty that includes limited processor, memory, and available energy to perform complex and computationally intensive tasks. This restriction poses the challenge to the mobile recommender system that must perform complex analytics—as real-time computations—to offer accurate recommendations. There is a need to extend the computation and storage resources of the mobile devices to enable efficient and scalable recommender system.

12.3.3 Computation-Based Cloud Computing Layer

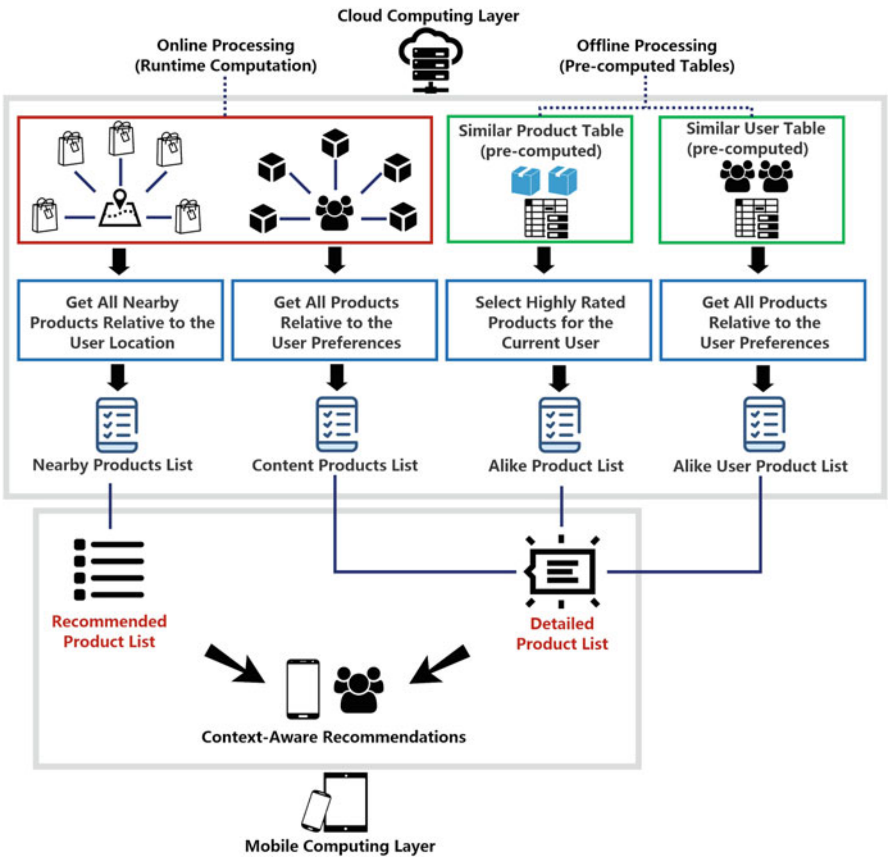
Cloud computing layer as the back-end of the system provides virtually unlimited (pay-per-use) hardware and software resources [24]. Specifically, the cloud layer offers infrastructure, platform, and software as a service to its users. Therefore, in order to compensate for the resource poverty of the context-aware mobile device, we use the software as a service offered by cloud servers that integrates the mobile and cloud computing technologies to generate the recommendations.

Once the mobile device captures user preferences and contextual information, the details are stored on the cloud-based server. The cloud server based on the input from the mobile computing layer computes the most relevant recommendations and communicates them back to the mobile device. The integration and operations of the mobile and cloud computing technologies are enabled by means of continuous availability of the network that enables the inter-layers communication. By using the layered architecture pattern, we distinguish between the two distinct concerns of user interaction and system processing with a systematic implementation of the recommendation system.

As in Fig. 12.3, the publish–subscribe patterns help to manage an effective coordination between the user level inputs and the system level processing. By applying this pattern, the market entities (such as outlets and restaurants) can publish their offerings of products/services to a central repository (such as cloud-based data storage) for their broadcasting. In contrast, the potential customers can subscribe to the published offerings that enable the matchmaking between both parties. The publish–subscribe patterns provide a systematic mediation between markets and potential customers to enable the digital matchmaking.

12.4 Algorithms and Technologies for Framework Implementation

After presenting the software architecture, we now present the details of the architecture-based implementation of the recommender framework. We present the algorithms that represent the data, modularization, and parameterized customization of the proposed framework in Sect. 12.4.1. We discuss the tools and technologies that implement the algorithms to provide the automation and proof-of-the-concept for the recommender system in Sect. 12.4.2.



Cloud Computing Layer
Online Processing (Runtime Computation)
Offline Processing (Pre-computed Tables)
Similar Product Table (pre-computed)
Similar User Table (pre-computed)
Get All Nearby Products Relative to the User Location
Get All Products Relative to the User Preferences
Select Highly Rated Products for the Current User
Get All Products Relative to the User Preferences
Nearby Products List
Content Products List
Alike Product List
Alike User Product List
Recommended Product List
Detailed Product List
Context-Aware Recommendations
Mobile Computing Layer

Fig. 12.4 An overview of the context-aware recommendation algorithms

12.4.1 Algorithms for Recommender System

We present the algorithms for the recommender system guided by the illustrations in Fig. 12.4. First the *recommend-products* algorithm is executed to collect the details of the nearby products (as per users' location and proximity details). The recommend-products algorithm is referred to as the online processing as it dynamically calculates the relevant products and offers based on users' location and preferences each time the recommender system is executed. The next two algorithms *similar-users-product-ratings* and *similar-product-ratings* support complementary functionality to find similar products and users. These two algorithms are precomputed, normally as part of the off-line processing. The technical details of these algorithms as in Fig. 12.4 are provided below (Table 12.2).

In the context of the existing recommendation systems (cf. Sect. 12.2, Fig. 12.2) We have adopted the hybrid recommendation system approach. This approach uti-

Table 12.2 Utility methods of algorithms to generate context-aware recommendations

Method (parameter)	Returns	Description
FIND-SIMILAR-USERS (id)	List	Get products recommended to similar users having high scores corresponding to active user
FIND-SIMILAR-PRODUCTS (user id)	List	List products from similar-products table having high scores corresponding to active user
GET-LOCAL-PRODUCTS (user location)	List	List all available products near to active user's current location
GET-PRODUCT-PREFERENCES (user id)	List	List all available products based on active user's preferences
GET-TOP-PRODUCTS (similar product list)	List	Get list of top k products from similar-products table rated highly corresponding to active user
GET-TOP-USERS-PRODUCTS (similar user list)	List	Get list of top k products from similar-users table rated highly corresponding to active user

Table 12.3 Parameters of algorithms to generate context-aware recommendation

Parameter	Description
P_{local}	List of products located in the vicinity of the active user's current location
$P_{preferences}$	List of products falling in similar category of active user's preferences
$P_{similar}$	List of products from similar products list rated highly by the active user
$P_{similar-user}$	List of products from similar-users table rated highly for the active user
$P_{detailed}$	Combined product list of three lists that includes "content product list," "alike product list," and "alike user-product list"
$P_{recommended}$	Contains common products in both detailed and nearby product lists

lizes the context information along with content filtering technique and collaborative filtering algorithm to generate best possible recommendations. The utility function of the implemented system is as follows:

$$\text{Context} \times \text{User} \times \text{Product} \rightarrow \text{Recommendation}$$

All the algorithmic execution and data storage take place at the back-end cloud server. Mobile devices only act as portable and context-aware user interfaces to provide some input (user location and preferences) and output (context-aware recommendations) as in Fig. 12.4. The unification of the mobile and cloud computing helps with a portable, context-aware recommender system with necessary computation and storage resources.

Table 12.3 presents a list of variables that support the parameterization of algorithms for online recommendation's generation. Also, Table 12.2 highlights all the utility methods that are used during the process of online recommendation generation.

Algorithm 1: Recommend-Products

- **Input:** Active user's id (uid), geolocation of the user (loc), and preferences of the user (preferences).
- **Process:** Based on the active user's location all locally available products and offerings are selected and compiled as a list (P_{local}—Line 2, Algorithm 1). This list is used to retrieve a detailed product list as per the user preferences ($P_{detailed}$—Line 4). A procedure runs to find similar products using preferences of similar users. Once all duplicated data is removed, a recommendation is returned based on top k items that rated highly for the active user ($P_{recommended}$—Line 5) in Algorithm 1. The tables comprising similar users and products are precomputed using off-line algorithms described later.
- **Output:** A list of recommended products $P_{recommended}$.

Algorithm 1 Recommend-products algorithm

```
Require: current user: uid, geolocation: loc, user preferences: $P_{preference}$
 1: $P_{local} \leftarrow$ GET-LOCAL-PRODUCTS($loc$)
 2: $P_{preference} \leftarrow$ GET-PRODUCT-PREFERENCES($uid$)
 3: $P_{similar} \leftarrow$ FIND-SIMILAR-PRODUCTS($uid$)
 4: $P_{detailed} \leftarrow P_{local} \cap (P_{preference} \cup P_{similar})$
 5: $P_{recommended} \leftarrow$ GET-TOP-PRODUCTS($P$)
Ensure: $P_{recommended}$ a list of recommended products
```

Algorithm 2: Similar-Users-Product-Ratings

- **Input:** User-product rating matrix ($\mathcal{U}sers$).
- **Process:** The process illustrated in Algorithm 2 picks all products that are not rated by the current user (ρ—Line 3, Algorithm 2). In the next step, a locality-based criterion based on the user location is used to find all neighboring users with available product ratings ($P_{similar-user}$—Line 4). Only users within the same demographic category are considered while compiling the neighboring users' set. In the end, ratings for the target product are calculated based on a weighted average of neighboring users' rating ($P_{predicted}$—Line 5). The resulting ratings are updated to the user-product rating matrix.
- **Output:** A list of predicted product ratings $P_{predicted}$.

Algorithm 3: Similar-Product-Ratings Algorithm

- **Input:** Product rating matrix ($\mathcal{P}roducts$).
- **Process:** First all similar users who have not rated a target product are selected ($U_{similar}$—Line 1, Algorithm 3). In the next step, all other products rated by those users are searched. Cosine similarity is used to calculate product similarity

Algorithm 2 Similar-users-product-ratings algorithm

```
Require: user-product rating matrix: ($\mathcal{U}sers$)
 1: Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$
 for each: $u \in \mathcal{U}sers$
 2:   $P_{user} \leftarrow$ GET-PRODUCTS($u$)
 3:   $U_{similar} \leftarrow$ GET-SIMILAR-USERS($P_{user}$)
 4:   $\rho \leftarrow$ PEARSON($u, U_{similar}$)
 5:   $P_{similar-user} \leftarrow$ GET-SIMILAR-PRODUCTS($\rho$)
 6:   $P_{predicted} \leftarrow$ PREDICT-PRODUCTS($P_{similar-user}, u$)
Ensure: $P_{predicted}$ a list of predicted product ratings
```

of the target product to other products (ρ—Line 3). This results in a subset of most similar products ($P_{similar}$—Line 4). Subsequently, a rating for the target product is predicted using the ratings of the similar products. The predicted rating is updated against the target product in the user-product rating matrix ($P_{predicted}$—Line 4).

– **Output:** A list of predicted product ratings $P_{predicted}$.

Algorithm 3 Similar-product-ratings algorithm

```
Require: user-product rating-matrix
 for each: $p \in \mathcal{P}roducts$
 1:   $U_{similar} \leftarrow$ GET-SIMILAR-USERS($p$)
 2:   $P_{similar} \leftarrow$ GET-SIMILAR-PRODUCTS($U, p$)
 3:   $\rho \leftarrow$ COSINE($p, P$)
 4:   $P_{similar} \leftarrow$ GET-SIMILAR-PRODUCTS($\rho$)
 5:   $P_{predicted} \leftarrow$ PREDICT-RATING($P_{similar}$)
Ensure: $P_{predicted}$ a list of predicted product ratings
```

12.4.2 Tools and Technologies for Framework Implementation

After presenting the algorithms, we now discuss the tools and technologies used to implement the framework. The framework implementation represents a prototype based proof-of-the-concept for the proposed solution. We have used the architecture from Fig. 12.3 to implement the framework. An overview of the integrated tools and technologies to implement the mobile computing and cloud computing layers is provided in Fig. 12.5.

We have exploited the Amazon cloud services for storage and computing efficiency. From a technical perspective, we have deployed a virtual server on Amazon cloud called Amazon EC2 instance[3] and set up Red Hat Linux operating system over that instance. We have developed server side application using Node.js[4]

[3]Amazon EC2: https://aws.amazon.com/ec2/.

[4]Node.js: https://nodejs.org/en/.

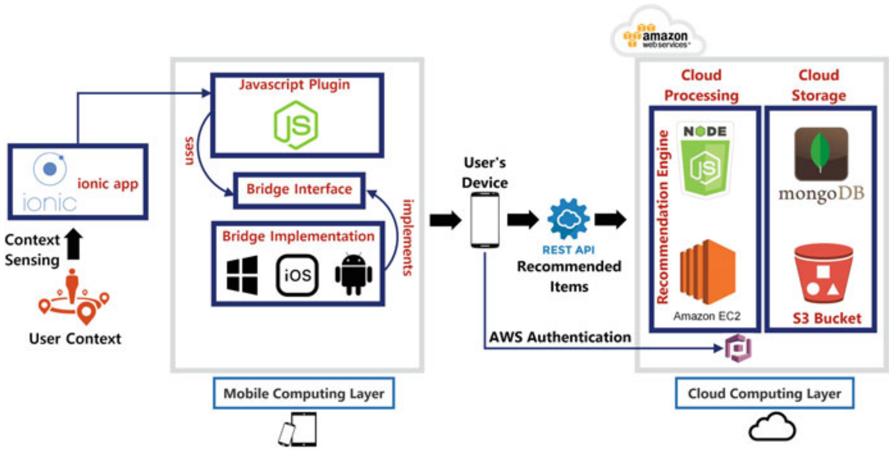


Fig. 12.5 Overview of the tools and technologies to implement framework

and set up Node.js web server on Amazon EC2 Instance. For the sake of efficient data retrieval we have used Mongo DB.[5] We installed MongoDB on Amazon EC2 Instance. Recommendation related data is managed by Amazon S3 storage services.[6]

The recommendation engine is written in python language[7] installed over Amazon EC2. 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 user's recommendations. The algorithmic details have already been discussed in the previous section.

12.4.3 Implementing Context-Aware Mobile Computing Layer

From an implementation point of view, at the mobile computing layers (i.e., context-aware user interface) we have exploited HTML5[8] technologies to support multiple mobile platforms. Moreover, the reason to choose a platform independent technology for the mobile layer is that the framework carries out all performance intensive tasks over cloud layer. The core responsibility of the mobile layer is to retrieve the current location of the user and calculate users' preferences to (1)

[5]MongoDB: https://www.mongodb.com/.

[6]Amazon S3: https://aws.amazon.com/s3/.

[7]Python: https://www.python.org/.

[8]HTML 5—World Wide Web Consortium: https://www.w3.org/TR/html5/.

send them to cloud end of the system, and (2) display the recommended products to the 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. The user of the framework needs to be registered to system to get any recommendations. User information is stored into user table/collection in MongoDB installed over Amazon EC2. After the user gets logged into the system, user's current location is retrieved. The current location of the active user and preferences are sent to Node JS server via Restful API.

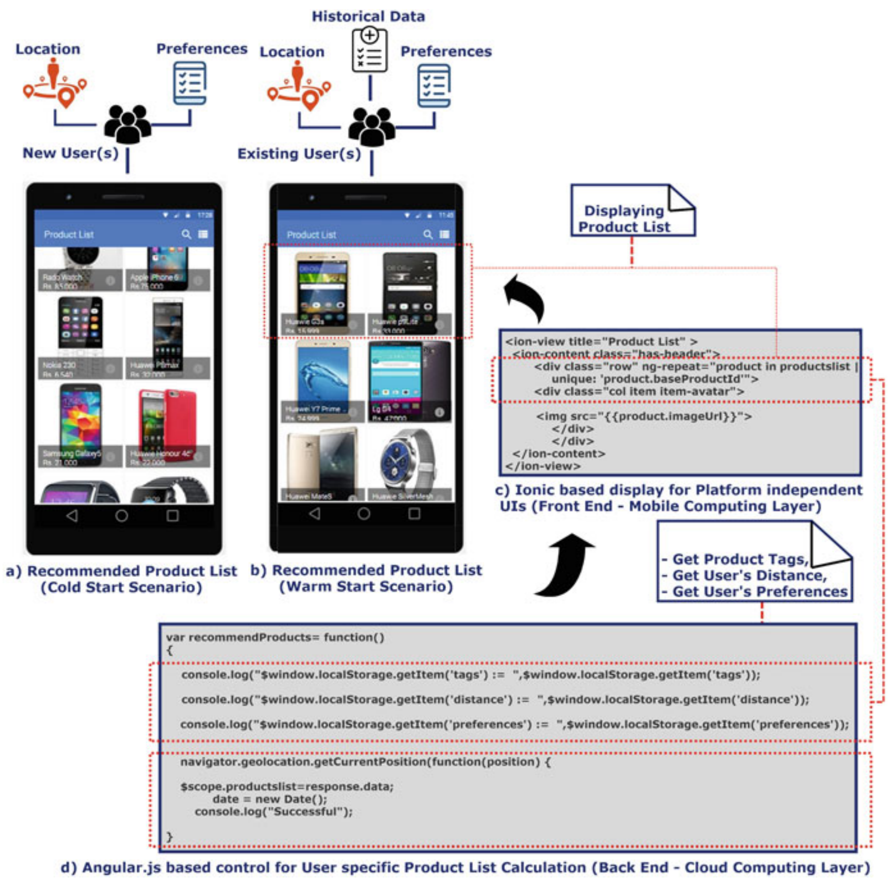
12.4.4 Implementing Processing Based Cloud Computing Layer

This section describes the methodological details to implement cloud end of the recommender system. It is vital to mention that all computational and storage work for recommendation generation is performed over cloud layer using node JS server and MongoDB that are deployed on Amazon EC2 instance. Another benefit gained by cloud layer is off-line processing performed by python based recommendation engine. The purpose of off-line processing is to overcome the problems of scalability and performance.

To implement off-line processing we have used collaborative filtering algorithm [8, 25, 44]. We have utilized the table user-product rating matrix to apply above mentioned techniques and generate precomputed tables of similar users as well as similar products (cf. Algorithms 2 and 3). Python based recommendation engine runs these algorithms to refresh precomputed tables on daily basis based on the time when there is a minimalistic use of the framework. In order to compute item based similarities we have utilized cosine-based similarity (cf. Algorithm 3—Line 3) and for the sake of user based similarities' computation, we have utilized Pearson correlation method (cf. Algorithm 2—Line 3). We have unified the items based collaborative filtering technique with user based collaborative filtering technique during off-line processing.

Figure 12.6 presents an overview of the context-based recommendation of the products' list to the users. As highlighted in Fig. 12.6, there are two types of users, namely: (1) new users and (2) existing users that lead to two scenarios for the recommendations that are detailed below.

– **Cold Start Scenario** represents the situation when a new user utilizes the framework for context-aware recommendations as illustrated in Fig. 12.6a. Since the framework has no prior information about the user's context, any computations and recommendations by the framework are cold start. In the cold start scenario, the framework gathers user's location and preferences to generate the recommendations.
– **Warm Start Scenario** represents the situation when an existing user utilizes the framework to get the recommendations as illustrated in Fig. 12.6b. In this


Location
Preferences
New User(s)
Historical Data
Location
Preferences
Existing User(s)
Product List
Displaying Product List
a) Recommended Product List (Cold Start Scenario)
b) Recommended Product List (Warm Start Scenario)
c) Ionic based display for Platform independent UIs (Front End - Mobile Computing Layer)
- Get Product Tags,
- Get User's Distance,
- Get User's Preferences
d) Angular.js based control for User specific Product List Calculation (Back End - Cloud Computing Layer)

```
<ion-view title="Product List" >
 <ion-content class="has-header">
      <div class="row" ng-repeat="product in productslist |
          unique: 'product.baseProductId'">
      <div class="col item item-avatar">

       <img src="{{product.imageUrl}}">
          </div>
          </div>
 </ion-content>
</ion-view>
```

```
var recommendProducts= function()
{
  console.log("$window.localStorage.getItem('tags') :=  ",$window.localStorage.getItem('tags'));

  console.log("$window.localStorage.getItem('distance') :=  ",$window.localStorage.getItem('distance'));

  console.log("$window.localStorage.getItem('preferences') :=  ",$window.localStorage.getItem('preferences'));

  navigator.geolocation.getCurrentPosition(function(position) {

  $scope.productslist=response.data;
       date = new Date();
    console.log("Successful");

}
```

Fig. 12.6 Overview of the product recommendations sample implementation logic

situation, the framework has prior contextual information about the user that helps the framework's accuracy of recommendations as a warm start. In the warm start scenario, the framework gathers user's location, preferences as well as user's historical data (e.g., past preferences, items/points of interests, and time/date) to generate the context-aware recommendations. These recommendations ensure a digital matchmaking in a virtualized context of smart markets involving potential customers and business entities.

We also highlight the sample code that executes at the mobile and cloud computing layers of the framework to generate the context-aware recommendations in Fig. 12.6. Figure 12.6c presents the partial view of the code to display the recommender product list to the users. For the display of the recommendation list, we have used ionic framework to support a platform independent code/technology. Figure 12.6d presents the back-end logic that gathers the user's contextual information to generate the recommendation list. The logic is executed at the cloud-based server. We have used the Angular JS to compute the contextual information.

12.5 Qualitative Evaluation of the Framework

We now present the results for the framework evaluation. Specifically, we discuss the dataset(s) used in Sect. 12.5.1 to evaluate the accuracy and efficiency of the framework presented in Sect. 12.5.2. Finally, we also present some threats to the validity of the evaluation results detailed in Sect. 12.5.3.

12.5.1 Platform, Metrics, and Dataset for Evaluation

– **Platform and Tools Used for Evaluation:** All evaluations were performed using Huawei P8 Lite smart-phone on the client side (i.e., mobile computing layer). On the other hand, on the server side, a Red Hat Enterprise Linux OS system with Node.js, MongoDB, and Python installed was used (i.e., cloud computing layer). The proposed recommender framework is evaluated based on two main criteria: (1) accuracy of the recommendations that are generated by the cloud server and (2) efficiency of the resource utilization by the mobile device in terms of CPU and power consumption. Memory consumption issues are not considered due to the fact that all the data storage takes place at the cloud-based server.
– **Metrics Used for Evaluation:** To evaluate software quality features of the developed recommender system, we have used ISO/IEC 9126-1 software quality standard [19] introduced by the International Organization for Standardization (ISO).[9] The standardized model investigates six quality features that are categorized into 27 sub-categories. Using an established model to evaluate the quality of the framework can help us to avoid any bias and guides feature based evaluation of the framework. To evaluate the proposed recommender framework, we only considered two quality features: *accuracy*, *efficiency*, and their sub-features.
– **Dataset Used for Framework Evaluation:** We used the dataset of superstore sales to evaluate the proposed recommender system. The dataset contains real items of product offerings and is publicly available at [30]. The selected dataset provides us with realistic data and scenarios to avoid any bias or limitations of the evaluation. We slightly modified the dataset to accommodate geographic locations (regions/provinces) corresponding to our needs that provides the foundation to evaluate the framework in a real context, as per the needs of the

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

local users and markets. For framework evaluation, we consider attributes of *customer name*, *product name*, *product category* and *sub-category*, *price*, and *geographic location* to propose a recommender system.

12.5.2 Results for Framework Evaluation

The result shows higher precision rates for the proposed system corresponding to good recommendations. Moreover, the proposed system demonstrated an efficient CPU consumption and memory usage.

Accuracy of Framework's Recommendations

In order to quantify and measure the accuracy of the framework's recommendations, we have used two metrics, namely: (a) *recommendation precision* as a measure of the accuracy of the recommendations made, and (b) *recommendation recall* measures the proportion of correct recommendations out of the total recommendations made by the framework. Mathematically, the metrics are defined as

$$\text{precision} = \frac{TP}{TP + FP}\ ; \text{recall} = \frac{TP}{TP + FN}$$

where

$$\begin{cases} TP : \text{ true positives} \\ FP : \text{ false positives} \\ TN : \text{ true negatives} \\ FN : \text{ false negatives} \end{cases}$$

We present the results of measuring the framework's accuracy based on the data in Table 12.4. Moreover, we provide an illustrative comparison of the evaluations and trials on the framework to measure its accuracy in Fig. 12.7. As highlighted in Table 12.4 and Fig. 12.7, there are two scenarios, namely: cold start and warm start recommendations that have been detailed earlier.

We performed the trials with 5 distinct user groups (UG), where each group had on average 10 people with varying age and gender groups along with distinct preferences to reduce any bias in the recommendation trials. The users' groups were asked to specify their preferences and let the framework provide them with context-aware recommendations. Based on the data in Table 12.4 and its visualization in Fig. 12.7, we observed that average precision and recall of the proposed system were 80.4% and 64.8%, respectively, in case of newly registered users (cold start scenarios). On the other hand, average precision and recall of the system were 82.6% and 72.6%, respectively, in case of already existing users (warm start scenarios).

Table 12.4 Precision and recall for newly registered users and existing user

	User group	Specific category total	Recommended	Relevant	Precision	Recall
Cold start scenario	UG1	81	63	45	71%	55%
	UG2	96	72	60	83%	62%
	UG3	98	98	84	85%	85%
	UG4	90	60	50	83%	55%
	UG5	91	65	52	80%	57%
	Avg.	91.2	71.6	58.2	80.4%	64.8%
Warm start scenario	UG1	84	96	72	75%	85%
	UG2	99	77	66	85%	66%
	UG3	72	63	54	85%	75%
	UG4	88	66	55	83%	62%
	UG5	80	70	60	85%	75%
	Avg.	84.6	74.4	61.4	82.6%	72.6%

Note: Column "specific category total" shows the total number of records related to a particular topic in the database. Column "recommended" represents the count of retrieved records, while column "relevant" represents the number of records relevant to user preferences. The last row in both tables describes the average for each column

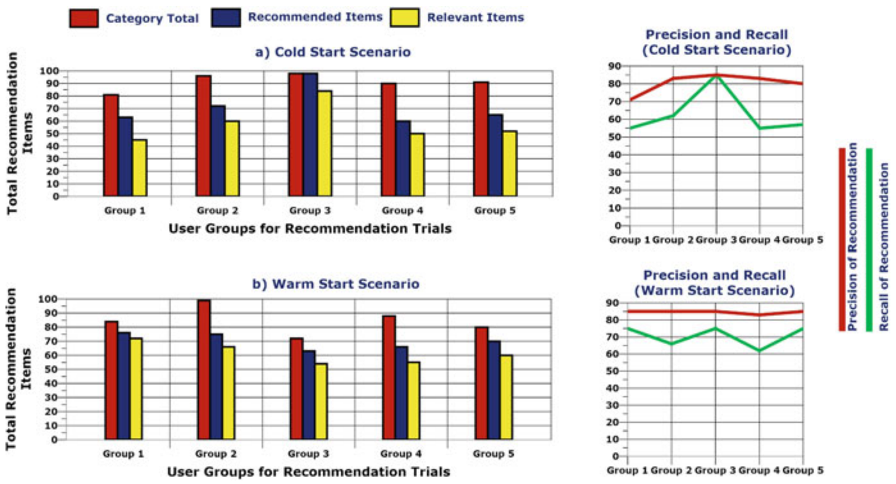


Fig. 12.7 Overview of the results for evaluating framework's accuracy for recommendation

A recommendation list was generated based on the location of the user as in Fig. 12.6. The recommendation list was also used to record and evaluate the precision and recall of recommendations provided by the system. The results have been shown in Table 12.4 for newly registered users and existing users, respectively.

Efficiency of the Framework

We now measure the computational and energy efficiency of the proposed recommender system. The data is offloaded to cloud-based server, therefore, evaluating the memory or storage efficiency of the mobile computing layer is out of the scope here. To assess and evaluate the efficiency of the recommender framework, we monitored its memory and CPU usage using CPU monitor [9]. An overview of the framework's processing and power efficiency monitoring is illustrated in Fig. 12.8. In Fig. 12.8, to measure the efficiency, we need to consider two execution modes of the framework:

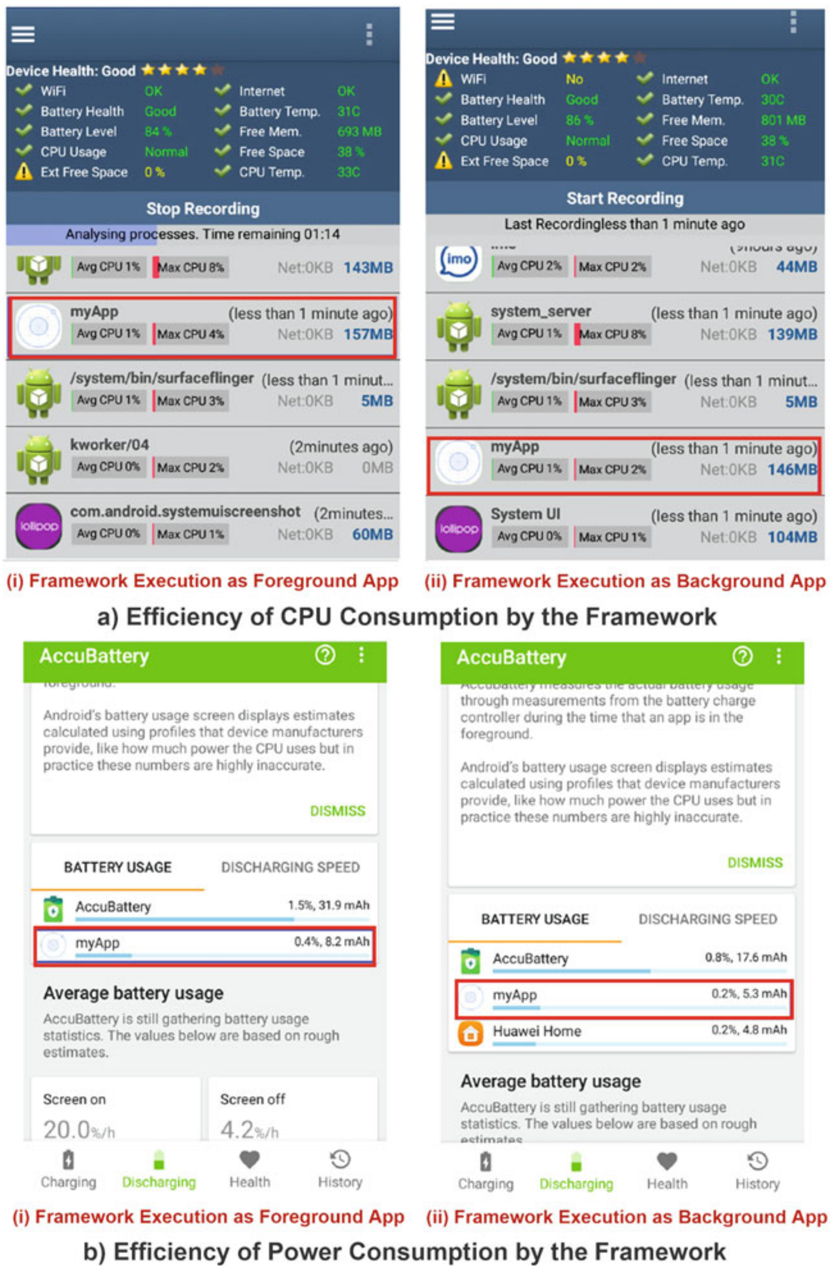
- **Framework Execution as a Foreground App** represents the scenario when the framework is active and fully executional during the recommendations process.
- **Framework Execution as a Background App** represents the scenario when the framework is only running in the background for context calculation but not operational for the users' recommendations.

We observed that the proposed application took approximately 3 s to fetch and display a list of recommended items acquired from the node server to the end user. In the former mode, CPU consumption did not exceed 4% and RAM used was 157 MB. In the latter mode, maximum CPU usage remained 2% and RAM usage was $\sim$146 MB. Furthermore, to measure battery consumption of the mobile application, we used AccuBattery [1]. The usage was normal in both execution modes, 0.2% and 0.4% in background and foreground modes, respectively.

12.5.3 Threats to the Validity of Framework

After presenting the framework evaluation, we also highlight some threats to the validity of the proposed research and solution. The threats also highlight the possible future work to optimize the proposed solution.

- **Threat I—Availability of the Diverse Dataset:** A possible threat to the validity relates to the availability of a diverse set of data. Diversity of data refers comprehensiveness of the user related information (i.e., gender, social and national background, emotion, etc.) along with time, day, and other environmental conditions to further evaluate the framework. The proposed algorithms provide parameterized customization of the solution. However, the availability of the diverse dataset can help us to further evaluate the accuracy and efficiency of the proposed and developed framework.
- **Threat II—Real Use-Cases from Smart Markets:** Smart recommender systems in general and smart markets in particular are relatively innovative concepts and lack any historical data. Unlike the more conventional recommender systems, the available usage scenarios for smart markets are limited. Moreover, the unification of the mobile and cloud computing technologies requires historical data and use-case for a more rigorous evaluation of the framework.


(i) Framework Execution as Foreground App
(ii) Framework Execution as Background App
a) Efficiency of CPU Consumption by the Framework
(i) Framework Execution as Foreground App
(ii) Framework Execution as Background App
b) Efficiency of Power Consumption by the Framework

(a)(i)

Device Health: Good ★★★★

WiFi	OK	Internet	OK
Battery Health	Good	Battery Temp.	31C
Battery Level	84 %	Free Mem.	693 MB
CPU Usage	Normal	Free Space	38 %
Ext Free Space	0 %	CPU Temp.	33C

Stop Recording

Analysing processes. Time remaining 01:14

Avg CPU 1% Max CPU 8% Net:0KB 143MB

myApp (less than 1 minute ago) Avg CPU 1% Max CPU 4% Net:0KB 157MB

/system/bin/surfaceflinger (less than 1 minut... Avg CPU 1% Max CPU 3% Net:0KB 5MB

kworker/04 (2minutes ago) Avg CPU 0% Max CPU 2% Net:0KB 0MB

com.android.systemuiscreenshot (2minutes... Avg CPU 0% Max CPU 1% Net:0KB 60MB

(a)(ii)

Device Health: Good ★★★★

WiFi	No	Internet	OK
Battery Health	Good	Battery Temp.	30C
Battery Level	86 %	Free Mem.	801 MB
CPU Usage	Normal	Free Space	38 %
Ext Free Space	0 %	CPU Temp.	31C

Start Recording

Last Recordingless than 1 minute ago

imo Avg CPU 2% Max CPU 2% Net:0KB 44MB

system_server (less than 1 minute ago) Avg CPU 1% Max CPU 8% Net:0KB 139MB

/system/bin/surfaceflinger (less than 1 minut... Avg CPU 1% Max CPU 3% Net:0KB 5MB

myApp (less than 1 minute ago) Avg CPU 1% Max CPU 2% Net:0KB 146MB

System UI (less than 1 minute ago) Avg CPU 0% Max CPU 1% Net:0KB 104MB

(b)(i)

AccuBattery

Android's battery usage screen displays estimates calculated using profiles that device manufacturers provide, like how much power the CPU uses but in practice these numbers are highly inaccurate.

DISMISS

BATTERY USAGE DISCHARGING SPEED

AccuBattery 1.5%, 31.9 mAh

myApp 0.4%, 8.2 mAh

Average battery usage

AccuBattery is still gathering battery usage statistics. The values below are based on rough estimates.

Screen on 20.0%/h

Screen off 4.2%/h

Charging Discharging Health History

(b)(ii)

AccuBattery

through measurements from the battery charge controller during the time that an app is in the foreground.

Android's battery usage screen displays estimates calculated using profiles that device manufacturers provide, like how much power the CPU uses but in practice these numbers are highly inaccurate.

DISMISS

BATTERY USAGE DISCHARGING SPEED

AccuBattery 0.8%, 17.6 mAh

myApp 0.2%, 5.3 mAh

Huawei Home 0.2%, 4.8 mAh

Average battery usage

AccuBattery is still gathering battery usage statistics. The values below are based on rough

Charging Discharging Health History

Fig. 12.8 Overview of the results for evaluating framework's efficiency

- **Threat III—24 × 7 Connectivity for Framework:** From a technical perspective, the solution exploits mobile devices as context-sensitive and portable user interface. The resource poverty of the mobile devices is alleviated with a continuous connectivity and processing at the cloud-based server. Therefore, a fundamental requirement to the success of the framework is a continuous network connection between a mobile device and the cloud-based server. In case of poor connectivity, the accuracy and performance of the framework can be affected.

12.6 Conclusions and Future Research

In this paper, we exploit the mobile cloud computing as state-of-the-art mobile computing technology to develop a context-sensitive and portable recommender system. The recommender system aims to support an efficient and context-driven matchmaking between potential customers (based on their shopping preferences) and relevant business entities (based on their products/service offerings). The recommender system supports the activities of the smart markets, i.e., virtualized and context-aware markets and/or shopping arena. Smart markets aim to facilitate the customers with recommendations and supporting business entities to maximize the outreach of their products and offerings. The proposed recommender system advances the state-of-the-art for recommender systems by exploiting a layered architecture that unifies the mobile computing and cloud computing technology layers. The system supports smart city systems in general and focuses specifically on smart markets.

Contributions and Outcomes The primary contributions of the solution lies with the proposed architecture and its underlying algorithms to sense contextual information from the users. The contextual information is matched with the best market offerings to facilitate the users with decision support based on contextual recommendations. We have used the publish–subscribe architectural pattern to reuse knowledge and best practices and customized it to enable an effective matchmaking and communication. The architectural model and pattern used have also helped us to model and develop mobile computing layer (front-end context-sensitive user interface) that relies on cloud computing layer (back-end data processor and storage) to alleviate the resource poverty of the mobile devices. In short, the proposed research presents an architecture, patterns, algorithms, enabling technologies, and implementation platform to develop a recommender system for smart markets.

Evaluations and Limitations We have developed and evaluated the prototype as a proof-of-the-concept for recommender system and its underlying algorithms. The prototype supports automation, user intervention, and customization during the recommendation process. We have used the ISO/IEC-9126-1 model to evaluate the quality in terms of accuracy and efficiency of the recommender system. To support a formal approach, we have utilized cosine-based similarity and Pearson correlation for computation of context-aware recommendations. The evaluation results suggest that the framework supports high accuracy for recommendations and facilitates computation and energy efficient mobile computing. We have also highlighted some threats to the validity of the research.

Future Work In future, we mainly focus on extending the types of recommendations and its application to other domains of the smart city systems such as crowd-sensed recommendations. Also, the privacy of user's context, their preferences, and information are also of central importance as part of the future work. From the functional perspective, we aim to explore other contextual factors, such as weather conditions, a week of the day, etc., to further optimize and extend the types of recommendations.

References

1. Accbattery(version-1.1.7). https://play.google.com/store/apps
2. Ali, M., Zain, J.M., Zolkipli M.F., Badshah, G.: Mobile cloud computing & mobile battery augmentation techniques: a survey. In: 2014 IEEE Student Conference on Research and Development (SCOReD), pp. 1–6. IEEE, Piscataway (2014)
3. Al-Shamri, M.Y.H.: User profiling approaches for demographic recommender systems. Knowl.-Based Syst. **100**, 175–187 (2016)
4. Bakıcı, T., Almirall, E., Wareham, J.: A smart city initiative: the case of Barcelona. J. Knowl. Econ. **4**(2), 135–148 (Jun 2013)
5. Baltrunas, L., Ludwig, B., Peer, S., Ricci, F.: Context relevance assessment and exploitation in mobile recommender systems. Pers. Ubiquit. Comput. **16**(5), 507–526 (2012)
6. Braunhofer, M., Elahi, M., Ricci, F.: Sts: a context-aware mobile recommender system for places of interest. In: UMAP Workshops. Citeseer, 2014.
7. Buschmann, F., Henney, K., Schimdt, D.: Pattern-Oriented Software Architecture: On Patterns and Pattern Language, Vol. 5. John Wiley & Sons, Hoboken (2007)
8. Colombo-Mendoza, L.O., Valencia-Garcia, R., Rodriguez-Gonzalez, A., Alor-Hernandez, G., Samper-Zapater, J.J.: Recommetz: a context-aware knowledge-based mobile recommender system for movie showtimes. Expert Syst. Appl. **42**(3), 1202–1222 (2015)
9. Cpumonitor(version-6.54). https://play.google.com/store/apps
10. Dameri, R.P.: Smart City and Digital City Implementation: Two Best Practices in Europe, pp. 109–154. Springer, Berlin (2017)
11. da Silva, W.M., Alvaro, A., Tomas, G.H.R.P., Afonso, R.A., Dias, K.L., Garcia, V.C.: Smart cities software architectures: a survey. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 1722–1727. ACM, New York (2013)
12. de Gemmis, M., Lops, P., Musto, C., Narducci, F., Semeraro, G.: Semantics-aware content-based recommender systems. In: Recommender Systems Handbook, pp. 119–159. Springer, Boston (2015)
13. Demers, A., Gehrke, J., Hong, M., Riedewald, M., Walker, W.: Towards expressive publish/-subscribe systems. In: EDBT, vol. 6, pp. 627–644. Springer, Berlin (2006)
14. Derwein, C., Beer, W., Hargassner, W., Herramhof, S.: General Framework for Context-Aware Recommendation of Social Events. IARIA, Vienna (2013)
15. Gosling, S.D., Rentfrow, P.J., Swann, W.B.: A very brief measure of the big-five personality domains. J. Res. Pers. **37**(6), 504–528 (2003)
16. Hasman, L.: An introduction to consumer health apps for the iphone. J. Consum. Health Internet **15**(4), 322–329 (2011)

17. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud migration research: a systematic review. IEEE Trans. Cloud Comput. **1**(2), 142–157 (2013)
18. Jones, N.C., Meter, R.V., Fowler, A.G., McMahon, P.L., Kim, J., Ladd, T.D., Yamamoto, Y.: Layered architecture for quantum computing. Phys. Rev. X **2**(3), 031007 (2012)
19. Jung, H.-W., Kim, S.-G., Chung, C.-S.: Measuring software product quality: a survey of ISO/IEC 9126. IEEE Softw. **21**(5), 88–92 (2004)
20. Khalid, O., Khan, M.U.S., Khan, S.U., Zomaya, A.Y.: OmniSuggest: a ubiquitous cloud-based context-aware recommendation system for mobile social networks. IEEE Trans. Serv. Comput. **7**(3), 401–414 (2014)
21. Kitanov, S., Janevski, T.: State of the art: mobile cloud computing. In: 2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), pp. 153–158. IEEE, Piscataway (2014)
22. Lewis, G.A., Lago, P., Procaccianti, G.: Architecture strategies for cyber-foraging: preliminary results from a systematic literature review. In: European Conference on Software Architecture, pp. 154–169. Springer, Berlin (2014)
23. Medvidovic, N., Taylor, R.N.: Software architecture: foundations, theory, and practice. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 2, pp. 471–472. ACM, New York (2010)
24. Mell, P., Grance, T.: The NIST definition of cloud computing. Commun. ACM **53**(6), 50 (2010)
25. Noguera, J.M., Barranco, M.J., Segura, R.J., MartíNez, L.: A mobile 3D-GIS hybrid recommender system for tourism. Inf. Sci. **215**, 37–52 (2012)
26. Otebolaku, A.M., Andrade, M.T.: Supporting context-aware cloud-based media recommendations for smartphones. In: 2014 2nd IEEE International Conference Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 109–116. IEEE, Piscataway (2014)
27. Postel, J.: RFC 793: transmission control protocol, September 1981. Status: Standard **88** (2003)
28. Rappaz, J., Vladarean, M.-L., McAuley, J., Catasta, M.: Bartering books to beers: a recommender system for exchange platforms. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, pp. 505–514. ACM, New York (2017)
29. Roberts, M., Ducheneaut, N., Begole, B., Partridge, K., Price, B., Bellotti, V., Walendowski, A., Rasmussen, P.: Scalable architecture for context-aware activity-detecting mobile recommendation systems. In: 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008, pp. 1–6. IEEE, Piscataway (2008)
30. Sample—superstore sales. https://community.tableau.com/docs/DOC-1236. Last modified by: Micheal Martin
31. Sanchez, F., Barrilero, M., Uribe, S., Alvarez, F., Tena, A., Menendez, J.M.: Social and content hybrid image recommender system for mobile social networks. Mob. Netw. Appl. **17**(6), 782–795 (2012)
32. Sassi, I.B., Mellouli, S., Yahia, S.B.: Context-aware recommender systems in mobile environment: on the road of future research. Inf. Syst. **72**, 27–61 (2017)
33. Satyanarayanan, M.: Mobile computing: the next decade. In: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, p. 5. ACM, New York (2010)
34. Stuedi, P., Mohomed, I., Terry, D.: 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, p. 1. ACM, New York (2010)
35. Su, J.-H., Wang, B.-W., Hsiao, C.-Y., Tseng, V.S.: Personalized rough-set-based recommendation by integrating multiple contents and collaborative information. Inf. Sci. **180**(1), 113–131 (2010)
36. Szczerbak, M., Toutain, F., Bouabdallah, A., Bonnin, J.-M.: Collaborative context experience in a phonebook. In: 2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 1275–1281. IEEE, Piscataway (2012)
37. Taleb, T., Dutta, S., Ksentini, A., Iqbal, M., Flinck, H.: Mobile edge computing potential in making cities smarter. IEEE Commun. Mag. **55**(3), 38–43 (2017)

38. Tarus, J.K., Niu, Z., Kalui, D.: A hybrid recommender system for e-learning based on context awareness and sequential pattern mining. Soft. Comput. 1–13 (2017)
39. Turban, E., Whiteside, J., King, D., Outland, J.: Mobile Commerce and the Internet of Things, pp. 167–199. Springer, Berlin (2017)
40. Wang, T., Liu, L.: Privacy-aware mobile services over road networks. Proc. VLDB Endowment **2**(1), 1042–1053 (2009)
41. Wang, S.-L., Chen, Y.L., Kuo, A.M.-H., Chen, H.-M., Shiu, Y.S.: Design and evaluation of a cloud-based mobile health information recommendation system on wireless sensor networks. Comput. Electr. Eng. **49**, 221–235 (2016)
42. Wiesner, M., Pfeifer, D.: Health recommender systems: concepts, requirements, technical basics and challenges. Int. J. Environ. Res. Public Health **11**(3), 2580–2607 (2014)
43. Yang, W.-S., Cheng, H.-C., Dia, J.-B.: A location-aware recommender system for mobile shopping environments. Expert Syst. Appl. **34**, 437–445 (2008)
44. Yang, B., Lei, Y., Liu, J., Li, W.: Social collaborative filtering by trust. IEEE Trans. Pattern Anal. Mach. Intell. **39**(8), 1633–1647 (2017)