

A CONTEXT-AWARE REMINDER SYSTEM BASED ON PUBLISH AND
SUBSCRIBE MODEL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖNDER ÇAĞLAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

JULY 2013

A CONTEXT-AWARE REMINDER SYSTEM BASED ON PUBLISH AND
SUBSCRIBE MODEL

Submitted by **Önder ÇAĞLAR** in partial fulfillment of the requirements for the
degree of **Master of Science in the Department of Information Systems,**
Middle East Technical University by,

Prof. Dr. Nazife Baykal
Director, Informatics Institute

Prof. Dr. Yasemin Yardımcı Çetin
Head of Department, Information Systems

Assoc. Prof. Dr. Altan Koçyiğit
Supervisor, Information Systems, METU

Examining Committee Members

Assist. Prof. Dr. Pekin Erhan Eren
IS, METU

Assoc. Prof. Dr. Altan Koçyiğit
IS, METU

Dr. Nail Çadallı
KAREL A.Ş.

Assist. Prof. Dr. Banu Günel
IS, METU

Assist. Prof. Dr. Alptekin Temizel
WBLS, METU

Date: 18.07.2013

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : ÖNDER ÇAĞLAR

Signature :

ABSTRACT

A CONTEXT-AWARE REMINDER SYSTEM BASED ON PUBLISH AND SUBSCRIBE MODEL

ÇAĞLAR, Önder

M.S., Department of Information Systems

Supervisor: Assoc. Prof. Dr. Altan Koçyiğit

JULY 2013, 55 pages

The usage of computing technologies to build smart and interactive human-centric systems that support our daily life activities is gaining popularity day by day. Pervasive computing seems to be a new computing revolution era in the near future. In this sense, context-aware reminder systems stand out as interesting and stimulating research topic. Such systems are used to remind important events and activities properly at the right time and at the right place. In this context, the system detects the user's environment and makes the recall in the most appropriate way. Reminding the message in the most appropriate way without disturbing the user provides user satisfaction naturally. The major purpose of this study is to develop a context aware reminder system to improve the quality of life for students in any university campus. The system developed in this study is based on publish-subscribe model. In this model, reminder notifiers are called publisher and reminder receivers are called subscriber. A prototype has been developed to demonstrate the applicability of the proposed context-aware reminder system. This prototype system consists of server-side application which is used to create and send a reminder message and an android application which is used to retrieve and display the reminder message.

Keywords: Pervasive/Ubiquitous Computing, Context Aware Reminder
Systems/Applications, Publish - Subscribe Model.

ÖZ

YAYINLA VE ABONE OL MODELİ ÜZERİNE KURULU BİR BAĞLAM FARKINDALIKLI HATIRLATMA SİSTEMİ

ÇAĞLAR, Önder

Yüksek Lisans, Bilişim Sistemleri Bölümü

Tez Yöneticisi: Doç. Dr. Altan KOÇYİĞİT

Temmuz 2013, 55 sayfa

Günlük yaşam aktivitelerimize destek olan akıllı ve etkileşimli insan merkezli sistemlerin inşa edilmesinde yaygın hesaplama teknolojilerinin kullanımı gün geçtikçe popülerlik kazanmaktadır. Yakın gelecekte yaygın hesaplama, hesaplama devrimi için yeni bir hesaplama çağı olacak gibi görünüyor. Bu anlamda, bağlam farkındalıklı hatırlatma sistemleri de ilgi ve dikkat çekici araştırma konusu olarak göze çarpıyor. Bu tarz sistemler, önemli olay ve aktivitelerin doğru zaman, doğru yer ve doğru şekilde kullanıcıya hatırlatılması amacıyla kullanılırlar. Bu bağlamda sistem kullanıcının içinde bulunduğu ortamı algılayarak hatırlatmanın en uygun biçimde yapılmasına olanak sağlar. Hatırlatmaların kullanıcıyı rahatsız etmeden en uygun şekilde yapılması doğal olarak kullanıcı memnuniyeti sağlar. Bu çalışmanın başlıca amacı, bir üniversite kampüsünde öğrencilerin yaşamını kolaylaştıracak bağlam farkındalıklı bir hatırlatma sistemi geliştirmektir. Geliştirilen sistem yayımla-abone ol modeli üzerine kuruludur. Hatırlatıcı bildirenler yayıncı, hatırlatıcı alıcılar ise abone olarak adlandırılmaktadır. Bu çalışmada, bir prototip önerilen bağlam farkındalıklı hatırlatma sisteminin uygulanabilirliğini göstermek

için geliştirilmiştir. Bu prototip sistem, hatırlatıcı mesajı oluşturmak ve göndermek için kullanılan sunucu tarafı uygulamalarından ve hatırlatıcı mesajlarını almak ve görüntülemek için kullanılan bir android uygulamasından oluşmaktadır.

Anahtar Kelimeler: Sürekli/Yaygın Hesaplama, Bağlam Bilinçli Hatırlatma Sistemleri/Uygulamaları, Yayınla - Abone Ol Modeli.

This thesis is dedicated to my family.

ACKNOWLEDGEMENTS

First and foremost I wish to thank my thesis supervisor Assoc. Prof. Dr. Altan Koçyiğit for his support, guidance, patience and trust from initial to the final part of the study. I cannot thank him enough for this great experience.

I would like to thank to Mehmet Koça, Mutlu Koça, Murat Duman, İbrahim Çalışır, and especially to Alpay Karagöz for their time and revisions.

I express my appreciation to Ahmet İlçi and Selim Yayla for their guidance on user tests.

Also I would like to thank to Yaver Cansız for helping me out in figure design.

I cannot thank enough to İlhan Zekerie Yumer for technical help for server side programming and database related issues. I also appreciate Onur Rauf Bingöl and Uğur Dökmeci's help very much for their help in server side programming.

I would like to thank to Mustafa Akkuzu and Ümit Sivri for their guidance in mobile programming.

I am very thankful to Emre Çağlayan and Nilgün Öner for their support.

I would like to thank to Gökçe Türkmendağ for theoretical topics on geographical location related issues.

I would also like to thank to Feride Erdal for her guidance in thesis formatting.

Last but not least, I also want to thank to my friend Hakan Yılmaz for helping me out for the reference section and final formatting.

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ	vi
DEDICATION	viii
ACKNOWLEDGEMENTS	ix
TABLE OF CONTENTS	x
LIST OF TABLES	xii
LIST OF FIGURES	xiii
CHAPTERS	
1 INTRODUCTION	1
2 LITERATURE REVIEW	3
2.1 Current and Traditional Reminder Systems	4
2.1.1 Paper To do lists.....	4
2.1.2 Post-it Notes	4
2.1.3 Human Assistant or Another Person	5
2.1.4 E-mail.....	5
2.1.5 Computer Reminder Software.....	5
2.2 Current Reminder Applications for Mobile Devices	5
2.3 2.3 Context Aware Reminder Systems.....	6
3 PROPOSED MODEL.....	9
3.1 The Basic Features of Proposed Model.....	10
3.1.1 Publish Subscribe Model	10

3.1.2	Reminder Pattern.....	11
3.2	Major Components of Proposed Model	14
3.2.1	Publishers	15
3.2.2	Distribution of Events	16
3.2.3	Subscribers	16
3.3	Typical Scenarios	19
3.3.1	Scenario 1.....	19
3.3.2	Scenario 2.....	19
4	IMPLEMENTATION.....	20
4.1	System Design.....	20
4.1.1	PubSubHubbub Protocol and Google Hub Service	21
4.1.2	Google Cloud Messaging for Android.....	21
4.1.3	Publishers	23
4.1.4	Distribution of Events (Application Server)	32
4.1.5	Subscribers	34
4.2	System Security	45
4.2.2	Login	46
4.3	Evaluation and Test Case (User Experiment)	47
4.3.1	Tasks	47
4.3.2	General Opinions of Participants about the System.....	49
5	CONCLUSION AND FUTURE WORK	52

LIST OF TABLES

Table 3.1: Reminder Template Content	12
Table 3.2: Reminder Pattern Example	14
Table 3.3: Event Content.....	15

LIST OF FIGURES

Figure 3.1: General Structure of Context Aware Reminder System.....	10
Figure 3.2: The Publish and Subscribe Model	11
Figure 3.3: Partial Domain Model of Proposed System	12
Figure 3.4: Location Alert Region (Inside / Outside)	13
Figure 3.5: Deployment Diagram of Context Aware Reminder System	14
Figure 3.6: The flowchart of the subscriber application.	18
Figure 4.1: The general structure of the prototype developed	20
Figure 4.2: How GCM works	22
Figure 4.3: Editor Page	24
Figure 4.4: Define Pattern Page	25
Figure 4.5: Defining Reminder Template	26
Figure 4.6: Sent Events Page	27
Figure 4.7: Cancelled Events Page	28
Figure 4.8: Archived Events Page.....	28
Figure 4.9: Locations Page.....	29
Figure 4.10: Define Location Page	30
Figure 4.11: Users table	30
Figure 4.12: atom_feed_reading Table	31
Figure 4.13: Reminder_Template and Reminder_Pattern Tables.....	31
Figure 4.14: Location Table.....	32
Figure 4.15: Atom_feed_reading Table	33
Figure 4.16: Association Between subscriber, publish, and publishsubscribe Tables	33

Figure 4.17: gcm_users Table	34
Figure 4.18: Home Page of Application and Event List Page	35
Figure 4.19: Event Details Page.....	36
Figure 4.20: Subscriber Page	37
Figure 4.21: Subscription Process Classes.....	38
Figure 4.22: New Event Page	39
Figure 4.23: Define Location Page	40
Figure 4.24: Reminder Pattern Page and Reminder Template Page.....	41
Figure 4.25: Class Diagram for the Subscriber Component	42
Figure 4.26: The flowchart for event triggering	43
Figure 4.27: Notification Example.....	44
Figure 4.28: Registration Form.....	46
Figure 4.29: Login Page.....	47

CHAPTER 1

1 INTRODUCTION

It's a truth that human race enjoys technology and taking advantage of it. However, this situation comes up with some drawbacks such as busy working hours, tra_c jam and so on. These drawbacks when combined with living under stress make people more depressed and amnesiac little by little while trying to manage their lives. To do so people tried tying ropes to fingers, managing to-do lists, using post-it cards and etc. to help them remember the things they really should do until these times.

The methods mentioned above were utilized as reminders. A reminder can be described as a signal and a description embedded in a special type of message. What is to be remembered in such reminder systems are indicated by using signals like audio-based signals, visual signals, etc. while it is clarified by descriptions. (Dey & Abowd, 2000). For example, an alarm clock provides only the signal while lacking description and an e-mail message is only a description which lacks a signal.

Reminders are delivered when a posted reminder gets triggered due to some event. In timebased reminder systems, reminders are triggered by time. Generally, the users set up an alarm and it gets triggered at a certain time and notifies the user. Another example can be the applications that are similar to the Outlook. In such applications, users set meeting date and meeting time to go off before meeting.

Location-based reminder systems have emerged with the recent development of technology and popularity in mobile device usage in daily life. In these systems, the reminder notifications are triggered with respect to location. As an example, programing the mobile device application by typing the message "remind me to take the garbage away, when I get home" can be shown as a simple location-based reminder. This reminder is triggered when users' geolocation come up to the users' home address recorded in reminder application. In recent years, more useful applications have been started to develop that can replace classical methods.

The usage of mobile devices has been growing progressively thanks to the rapid development in mobile device technology; Latest mobile phones provide a potentially convenient and truly ubiquitous platform for the detection of personal context such as location, as well as the delivery of reminders.

Today most of the smartphones have lively and pleasant user interfaces, embedded sensors and very optimized for surfing on the internet. Sensors in smart mobile devices allow collecting context information that delivers much more individual / personal and precise information with related to the users' environment. With this information, user satisfaction is

achieved by giving a better service to the user. Kim (2004) states: “Life will be much more easy-going if we are reminded more effectively”. In this manner, the reminder systems have a special place in our daily live.

The usage of mobile devices equipped with sensors and computing technologies to build smart and interactive human-centric systems that support our daily life activities are gaining popularity day by day. Pervasive computing which is also known as ubiquitous computing seems to be a new computing revolution era in the near future. In this sense, context-aware reminder systems stand out as an interesting and stimulating research topic.

Kim (2004) states that an effective reminder should have the ability to sense the right contexts like right time, right place and right person for reminding, in addition to the signal and description. Context aware reminder systems are useful to recall significant events and activities to the users by considering users’ context. These systems execute the set reminders by considering the right time and the right place. The system perceives the peripheral cues around the users and warns the users in a proper or predetermined way. Informing the users in a proper way via messages secures user satisfaction since the system performs this action without discomforting them. For this reason, the context-aware reminder systems are helpful and functional.

The major purpose of this study is to propose a context aware reminder system to improve the quality of life for students in a university campus. The proposed system is based on publish-subscribe model. In this model, reminder notifiers are called publishers and reminder receivers are called subscribers. Another important feature of the system proposed is the usage of reminder patterns. A reminder pattern specifies a set of reminder templates for a specific event type. For each event a set of reminders are set for the subscriber according to the reminder pattern associated by the publisher or the subscriber to the event. In this study, a prototype has been developed to demonstrate the applicability of context-aware reminder system proposed. This prototype system consists of server-side application which is used to create and send a reminder event and an android application which is used to retrieve and display the reminder event.

This thesis is structured as follows:

- In chapter 2, Current and traditional reminder systems, current reminder applications for mobile devices and context aware reminder systems are detailed.
- Chapter 3 introduces the conceptual description of a context-aware reminder system based on publish-subscribe model. In this chapter, the basic features of publish subscribe model and advantages of using publish subscribe model is given. Then, the major components of proposed model are explained in detail. Typical usages scenarios are also illustrated in chapter 3.
- Chapter 4 presents the implementation of context aware reminder system.
- Finally, in chapter 5 the conclusions and directions for future works are given.
- CD attached to this thesis contains the source codes of the prototype application.

CHAPTER 2

2 LITERATURE REVIEW

The usage of computing technologies to build smart and interactive human-centric ubiquitous but also minimally intrusive systems that support our daily life activities is raising its popularity in information technology day by day. This phenomenon is first introduced by Mark Weiser in early 1991s during his work at the Xerox Palo Alto Research Center and it has been called ubiquitous (pervasive) computing. In his seminal paper (Weiser, 1991); Mark Weiser expresses the benefits of the ubiquitous computing as an overcoming factor for the problem of information overload and adds that integrating computers seamlessly into the human environment instead of forcing humans to enter their environment will make using a computer refreshing and relaxing. The necessity of much further studies on this vision led researchers to context aware computing. The following years the context aware computing has become a popular research field of ubiquitous computing.

Context-aware systems provide exclusively innovative prospects for application developers and users. Context-Aware systems are described by (Baldauf, 2004) as “being able to adapt their operations to the current context without explicit user intervention and thus aim at increasing usability and effectiveness by taking environmental context into account”. As the studies evolved around this field, many definitions for context-awareness are made. However, the main purpose of context awareness is creating more adaptive, useful and user-friendly computing environment by gathering context data and adapting systems behavior with respect to this data.

Dey and Abowd (1999) define context as any kind of information that can be used to characterize the state of person, place, object or such entities which are relevant to the interaction between a user and an application. Main types of context can be classified into three categories. These are;

1. Physical Environment Context (current location, time, room or body temperature, light level, noise, etc.)
2. Human Context /User Context (identity preference, activities, task requirements, types of user, personal history, daily behavioral patterns etc.)
3. ICT Context or Virtual Environment Context (services available locally or remotely)

Of these context mentioned above, types of context data such as current location, identity, time and activity are more important than others and widely used in systems with respect to others.

Context information may be gathered by placing sensors and creating smart user environments. Retrieving network information, device status, and browsing users' activity

history also helps to gather context information. In the past, using such devices or sensors was not feasible since end users can not have access to that equipment all the time and locating these devices to everywhere was not cost-effective. Nowadays, using mobile devices for gathering context data provides a vital solution. Today, most of the mobile devices like smart phones, laptops or pads come along with built-in equipment or sensors like gyroscope, digital compass or so on. Furthermore, using mobile devices for gathering such context would provide much more individual / personal and precise data. With this vision on mind mobile devices are used for gathering context data at this study.

Current tools and reminder applications are not sufficient because they are lacking of information about context of the user. Herstad (1998) emphasized the importance of contextual information for human-centric applications and tools. In order to create useful, functional and powerful tools, the context information has to be taken into consideration during development process. Using contextual information enables building less obtrusive and more transparent applications. The aim of developing Context Aware Reminder systems is to provide more functional and proactive usage in terms of reminder systems usability. Traditional ways of handling reminders do not meet the expectations because of insufficient use of context. For that reason, developing context aware reminder system is gaining much more importance nowadays.

In this chapter, review of the related literature is given. The first section expresses current and traditional reminder systems. The second section expresses current reminder applications for mobile devices. Finally, the third section expresses context aware reminder systems.

2.1 Current and Traditional Reminder Systems

A reminder is a special type of message consisting of a signal and a description. In such a reminder system, what is to be remembered is indicated by using signals like audio-based signals, visual signals, etc. and what is to be remembered is explained by using descriptions.

For instance, an alarm clock is non-descriptive whereas an icon which provides a few signs for what needs to be remembered is partially descriptive and an e-mail message is fully descriptive (Dey & Abowd, 2000). In the this section, traditional reminders are explained shortly.

2.1.1 Paper To do lists

To do lists are simple reminder notes which are written on small papers in order to remind what should be done. Although they are easy to construct, to do lists are far from being proactive and can be quite outdated, furthermore keeping them up to date can cause overheads.

2.1.2 Post-it Notes

Similar to to-do-lists, but, post-it notes differ as they can be stuck to different location with the aim of the other people to see them. Placed for an intended person's view, post it notes may be viewed by also other people that cause a disadvantage in functionality of them. The uncertainty of whether the post-it note is viewed and reminded what is to be done at the right time and date can be shown as another disadvantage of using post-it notes as a reminder.

2.1.3 Human Assistant or Another Person

Another useful reminder mechanism can be using human support / help. A friend, secretary, human assistant or another person can be relied on to remind important events. This mechanism is the closest one to an ideal reminder. However hiring a person with this purpose is not cost effective and relying on another person is not an exact solution since the person itself can also forget what is to be done. In addition, they cannot be with us all the time.

2.1.4 E-mail

Being informal type of to do list, sending e-mail as a reminder functions for reminding what is to be done. The users can get e-mail from other people or themselves to remind some activity at a later time. These types of reminder messages are still lacking of being proactive and users constantly need to check e-mail box to be aware of descriptions.

2.1.5 Computer Reminder Software

Many operating systems today come along with software which functions as reminders. Microsoft Outlook, Mozilla Thunderbird, Google calendar can be given as examples to this kind of software. Many of them are fully descriptive as they provide audio and visual signals and text descriptions. However; a few of them have location based reminder support. Computer reminder software is similar to alarm clock considering the fact that triggering mechanism works depending on the date and time of event. Immobility of PC makes them suffer from the disadvantage of not reaching end users all the time.

All these reminders above mentioned are respectively basic and far from satisfying the needs of modern time mankind. Moreover, they are lacking of context-aware data which determines the functionality and usage of these reminders.

2.2 Current Reminder Applications for Mobile Devices

When searched for the keyword “reminder” in Google Play, it shows many Android applications. One of the most popular ones is MyCalendar¹. It is a birthday reminder which is synchronized with the Facebook and contacts from the phone. This application transfers the birthdays from Facebook. Moreover, it allows adding birthdays to the contacts that are already on the phone or SIM card memory. The settings of the established reminder can be changed and the users can receive notifications if they want to. Furthermore, the users can send birthday messages to their friends easily.

Another popular reminder is Business Calendar². It has paid and free version. The application can be synchronized with Google Calendar and it has many functionalities. It is used mainly for business events. The application has both graphical and textual screen options for the users. Moreover, the users arrange widgets for month, week, agenda and day views. The users can set repeatable events and they can view days in a month that they want by moving their fingers. Another useful functionality of the application is the multi-selection. The users can delete, move or copy events at the same time.

¹ An application by K-Factor Media.

² An application by Appgenix Software Free.

The most popular note taking application is Evernote³. Computer, mobile phone and tablet device version of this application are available. This application has the ability to record voice reminders, take pictures and generate to-do lists. Moreover, these notes are searchable users have the chance of finding the created notes whenever they want. Users can also share these notes via their mobile phones. Furthermore, the users can synchronize the created notes between the computers and devices they use. These notes can be edited by friends or others if the users have the premium version and if they allow these people to do so.

If we consider the reminder applications in the market and the examples given above, it could be said that most of these applications do not interact with the environment of the users. They have one way mechanic interaction with the users. The users set the reminder and when the time comes, the reminder just warns them. However, we have the chance to detect the environment and its elements in order to provide remarkable warnings to the users. The application can detect the location of the users and warn them if they had set a reminder for this location. For instance, let's assume that a user has borrowed a book from the university library and set a reminder to warn him/her when s/he is around the library so as to return the book before the book is overdue. The reminder program can warn the user independent from the location. However, if the reminder application has the perception of location, then it will warn the user when s/he is around the predefined location. Therefore, the reminder would be more functional and useful.

2.3 2.3 Context Aware Reminder Systems

In the past, several studies have been conducted on context-aware reminders. Dey and Abowd (2000) propose a context-aware tool that supports users in sending and receiving reminder by using context toolkit infrastructure which is developed for context aware computing in their early studies.

The comMotion (Marmasse & Schmandt, 2000) is a location-based context aware system which is developed by MIT Media Laboratory. In this project, the locations in user's daily life are associated with users' personal information. For instance, when users come near by the grocery store commotion reminds user his / her shopping list. By using GPS system for position sensing, comMotion learns bit by bit the locations where user frequently been. ComMotion displays the location on a map when a location is frequently detected and lead users to name it. Then users can name such places as school, home, dormitory, etc. or can ignore that location.

Zhou and Chu (2012) introduced a context aware reminder system for elders based on fuzzy linguistic model. In this study, reminder messages are revealed in a proper time and way based on the interrupt degree of users' current activities and the urgent degree of triggered reminder applications. Ho and Intille (2005) stated that the interrupt degree of user's current activity is influenced by many factor and it is very hard for people to exactly define the interrupt degree of a specific activity. For this reason, fuzzy linguistic model is adopted to classify activity interrupt degree into qualitative set rather than into a numerical one. Moreover, it is aimed to schedule and maintain reminders easily for the elderly or their care givers by separating the user activity contexts and contexts utilized to trigger a reminder messages in this study.

³ An application by Evernote Corporation.

The Memory glasses (DeVaul, Clarkson & Pentland, 2000), is an attempt to build an effective short-term memory assistance and reminder system based on wearable sensors. Memory glasses aim to deliver reminder under appropriate circumstances by considering time, location and user's activity. It uses wearable sensors (a camera and a microphone) to determine which activity the user is involved in (entering / leaving the office, etc.) and a reminder that associated with the user's activity is delivered by using audio output. In this study, it is proposed that the awareness of user's activity can be used to decide the when it is the proper time to interfere a user with a reminder.

Sohn, Li and Lee, (2005) designed a location based reminder application (Place-Its) to study people using location-aware reminders in their daily lives. Place-Its is designed as a kind of digital post-it notes. The main feature of Place-Its is that it allows the user to create reminder message and associate it to a physical location. In this study, location is considered as the primary context and application retrieves the cell tower ID from the local GSM network for marking and detecting physical location. Two week exploratory study revealed that location based reminders is useful and a mobile phone was a promising platform for personal ubiquitous computing.

A smart reminder system was developed in order to help elders not to forget complex activities which are called as "Coupling Activities" in the project (Chaminda, Klyuev & Naruse, 2012). The system predicts the coupling activities based on the user's recent behavior, recent location and past activity patterns. Wearable sensors were located on the user's body, on the wrist areas of both hands. The system tries to learn the dynamic behavior patterns with minimum governance of the user. The proposed system is claimed to achieve 80% average accuracy rate for the reminder prediction. The system was tested with four subjects. This system still needs development for activity recognition and location recognition so as to identify the activities more correctly.

Another example of reminder systems for personalized reminder systems was proposed by Kwon and Choi (2008). An integrated methodology was used in order to combine intelligent agent, semantic web technologies and RFID (radio frequency identification) in this example. Need awareness mechanism and associative theory were adopted to create a prototype system. RFID was used for the reminder system and Rescorla-Wagner Model was applied for needs awareness during the implementation of the system. Moreover, ontologies were used to communicate with different web services. However, the proposed system needs more development especially on multiple-criteria decision making in order to examine user's current needs.

Reminder systems are frequently used in medical context. A medication reminder system is proposed to reinforce users' medication compliance (Asai, Orszulak, Myrick, Lee, Coughlin & de Weck). Four types of failures inclined by forgetfulness were identified. Forgetfulness is important for the system since the system needs to consider different types of users' behaviors. The system works with the help of the sensors and actuators. These sensors and actuators are beneficial for the system to recognize different users' behaviors. This contextaware medication reminder system was developed considering the failure types but it is not tested with the actual users.

Mobile phones are useful devices to warn the owners for a situation that could attract their attentions (Li, Sohn, Huang & Griswold, 2008). Getting notifications could be irritating when the users are busy. Therefore, environmental cues are important to transmit information for proximity. A buddy proximity application, PeopleTones, is used in this project for a two-week study. There are three main contributions of this study. The first one

is the algorithm that is used for perceiving proximity. The second contribution is the techniques used to decrease the noise of sensor and power consumption. The last contribution is related with a method for forming inconspicuous environmental cues. Consequently, environmental cues are applicable notification modality for mobile phones implementations that use context awareness.

Personalized reminder systems detect users' needs based on the users' recent context with the help of the recent location and activity (Kwon, Choi & Park, 2005). NAMA (Need Aware Multi-Agent) is a prototype system that takes advantages of need identification methodologies. Agent and semantic web based technologies were used to create a personalized reminder system. NAMA uses Bluetooth technology to determine the recent location of the users via their mobile phones. Moreover, users' recent needs and preferences are considered to direct the users to the appropriate web services. NAMA is a context-aware system that considers the location of the users together with the time, identity and entity. However, the system has some limitations. One of them is not being able to use NAMA on WinCE-based PDAs. Another limitation is the speed of the system. Moreover, NAMA does not provide stability when it uses services via UDDI. On the other hand, NAMA uses combination of mechanisms such as agent, ontology and context-aware systems successfully. This combination enables NAMA to provide both personalized and proactive services.

In general, reminders are useful messages and help people to remember information without having to count on their memory alone. Nowadays, they also found a place in our digital life but they have not changed much in time. Most of them still use time to trigger reminder message. Therefore, developing a context aware reminder system is a step forward in reminders and provides user satisfaction as expected.

CHAPTER 3

3 PROPOSED MODEL

In this chapter, the conceptual description of a context-aware reminder system based on Publish Subscribe model is given. The proposed model aims to remind important events and activities at the right time and at the right place. In this context, the system detects the users' environment and makes the recall in the most appropriate way. Reminding the event in the most appropriate way without disturbing the user provides more functional and proactive usage in terms of reminder systems efficacy.

The main contributions of the proposed model are the usage of publish subscribe model and the usage of reminder patterns. In Publish Subscribe model, publishers are the senders of events and subscribers are the receivers of events. Subscribers show interest in one or more topics and only get events that are of interest. The main information flow is in the way from publishers to subscribers and publishers do not directly communicate with subscribers. Publish Subscribe model provides loose coupling between the subscribers and publishers. In this way, a publisher does not need to know all of its subscribers. Scalability is another important feature of Publish Subscribe model. A multitude publishers may send information / notifications to large number of subscribers.

In addition, the proposed model offers "Reminder Patterns" which are used by smart mobile devices in order to notify user more conveniently. Reminder pattern can have multiple reminders (alarms) and they determine when and where to prompt reminder application. If certain predefined situation happens (context info such as time, location are satisfied) then the corresponding reminding service is activated as an action.

The smart mobile devices are also used in the proposed model for retrieving and delivering the events to end users. The smart mobile devices provide a potentially convenient and truly ubiquitous platform for the detection of personal context such as location, as well as the delivery of reminders.

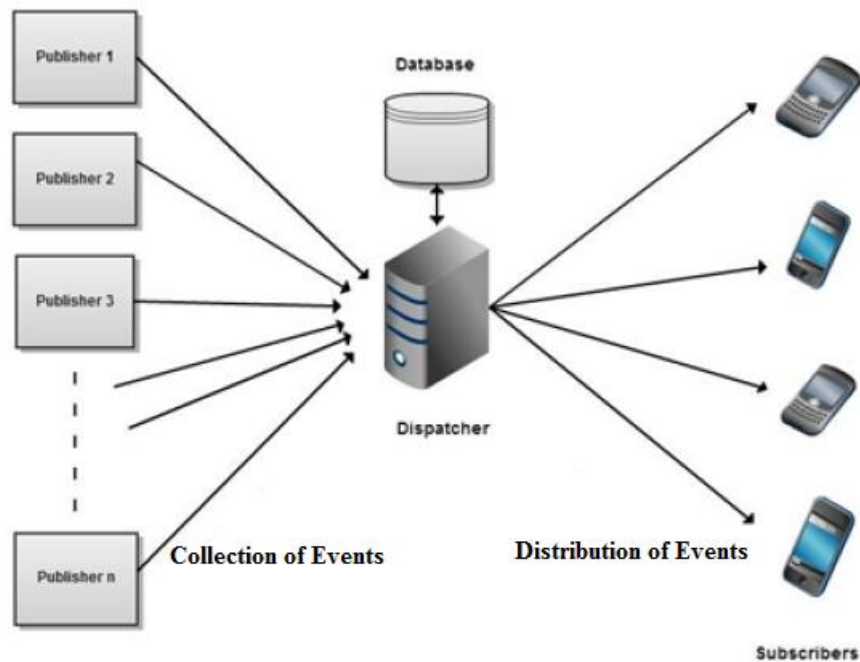


Figure 3.1: General Structure of Context Aware Reminder System

The general structure of proposed model is illustrated in Figure 3.1. In the proposed model, events from publishers are collected in the Dispatcher unit. Then, the events are distributed to the subscribed users. After receiving the events, subscriber application interprets the events and notifies users with respect to associated reminder pattern.

This chapter consists of three main subsections. First section explains the basic features of proposed model. Major Components of Proposed Model is given in the second section. Finally, typical scenarios are illustrated in the last section to give a clear overview of context aware reminder system.

3.1 The Basic Features of Proposed Model

The major contributions of the proposed model are that it uses the Publish Subscribe model and it utilizes Reminder Patterns. The following two sections explain the basic features of the Publish and Subscribe model and Reminder Patterns in detail.

3.1.1 Publish Subscribe Model

Publish and Subscribe Model is described as “a well-established communications paradigm that allows any number of publishers to communicate with any number of subscribers asynchronously and anonymously via an event channel. (Houlding, 2000) In publish and subscribe model, subscriber consumes information which is produced by publishers. The

main information flow is in the way from publishers to subscribers. The Figure 3.2 illustrates the Publish Subscribe model.



Figure 3.2: The Publish and Subscribe Model

The publishers often do not directly communicate with subscribers. This communication is performed by an indirect manner via publish / subscribe middleware. Publish Subscribe middleware is a logical intermediary between publisher and subscriber. Publish subscribe middleware is also called as “Notification Service”. Publish Subscribe Middleware;

- collects all the information/notifications from publishers.
- keeps all the information with related to all the subscribers that expressed interest in such notifications.
- dispatches all the published information/notifications to the right subscribers.

Loose coupling is one of the main features of Publish Subscribe model. Publishers and subscribers can exchange information without knowing of each other with this feature.

Scalability is another important feature of publish / subscribe model. Multiple publishers may send information / notifications to multiple subscribers. Publish Subscribe model provides a better scalability than traditional client / server models.

The usage of Publish Subscribe model in the proposed model provides anonymous, many-to-many and asynchronous communication. Publishers are loosely coupled to mobile client subscribers without knowing of their existence and when the size of the system grows, both publishers and subscriber still interact only with the Publish Subscribe middleware.

3.1.2 Reminder Pattern

The proposed model offers “Reminder Patterns” in order to notify users more appropriately. Reminder Pattern can contain multiple reminder templates that define when and where to trigger reminders. If a certain predefined situation occurs (context info such as time, location are satisfied) then the corresponding reminding service is activated as an action.

Event can be created by users based on to their needs or wishes. One event can have only one reminder pattern. On the other hand, one reminder pattern consists of one or more reminder templates. The reminder templates in one reminder pattern can be governed by both publisher and subscriber. Simply, a reminder pattern can be described as a group of reminder template collection.

The key concepts of the proposed model are Event, Reminder Pattern, Reminder Template and Reminder. Figure 3.3 shows the domain model of proposed system.

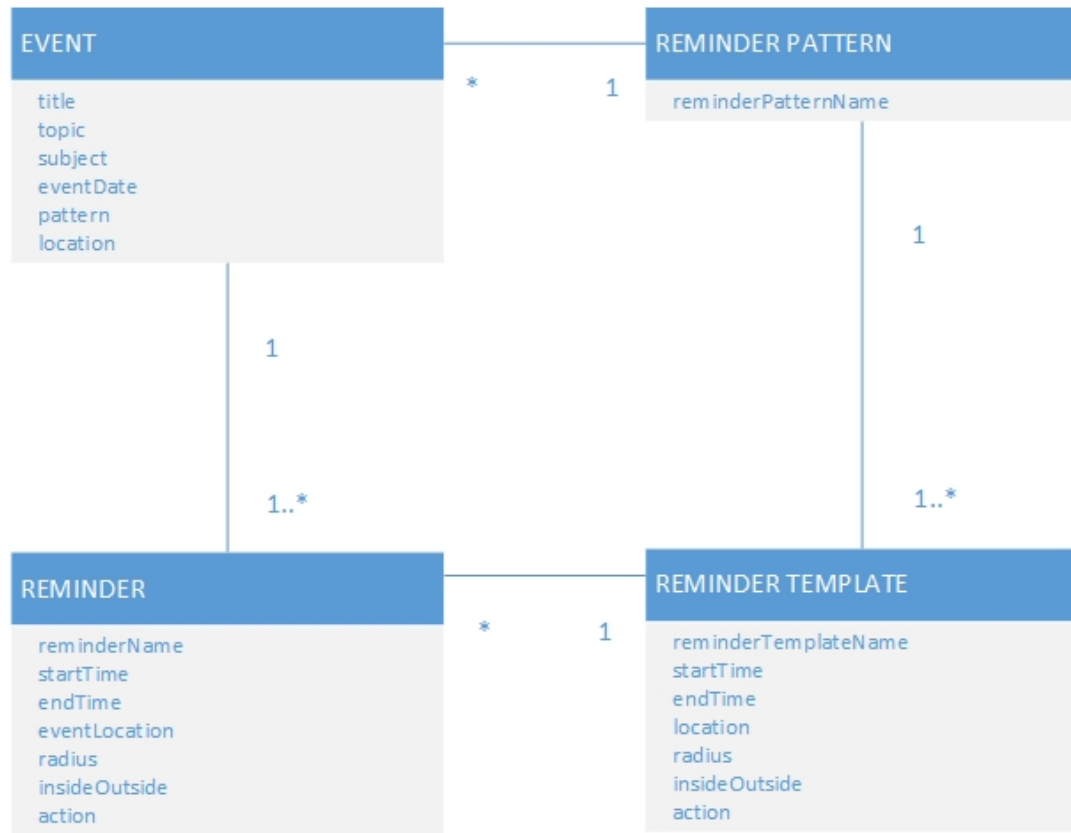


Figure 3.3: Partial Domain Model of Proposed System

User sets the following parameters while creating reminder templates.

Table 3.1: Reminder Template Content

Reminder Template	
<i>Time Interval</i>	The valid time range. It is used while triggering the reminder.
<i>Location</i>	It is used while triggering the location-based reminder.
<i>Radius</i>	Reminder Start Distance. Location-based reminder is triggered with respect to this distance.
<i>Inside / Outside</i>	It defines whether the user is inside or outside the area of location point.
<i>Action</i>	It defines how a reminder will respond. E.g. Sound, Vibrate.

If user does not set time interval parameter for a reminder template then reminder becomes a location-based reminder and the mobile application ignores time control. When location condition is satisfied then, the reminder is triggered and user is notified. If user does not set location parameter for a reminder template then reminder becomes a time-based reminder and the mobile application ignores location control. When time condition is satisfied then, the reminder is triggered and user is notified. When both location and time interval parameters are set by user, smart mobile application first checks time control and in given specific time interval, application starts to compare current location position with the event location. If the conditions are satisfied, reminder is fired and user is notified with prearranged action.

The location which is defined for a reminder template represents a location name such as Home, Work or Event Location. The location name such as Home and Work are saved earlier while user forms favorite location point.

Location alert can be triggered when the user is inside or outside the predefined location area. Location point is the central point of the alert region and it is given by the specific coordinates in terms of latitude and longitude. Inside or outside defines whether the event is fired when the user is inside or outside the area of location point. When the user chooses inside parameter, the reminder is triggered inside the alert region. On the other hand, if the user chooses outside parameter, the reminder is triggered outside the alert region. Location alert region is illustrated in Figure 3.4.

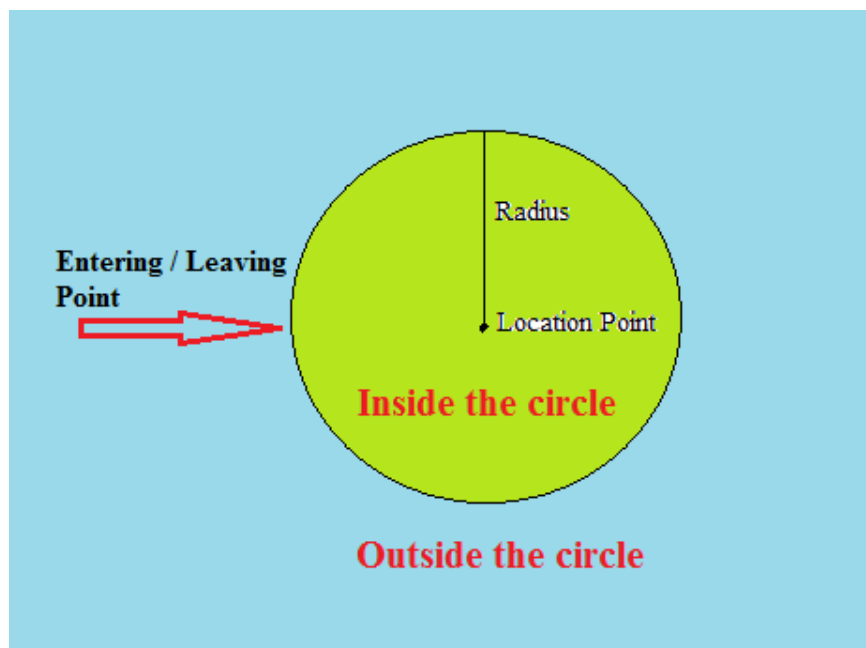


Figure 3.4: Location Alert Region (Inside / Outside)

Users can edit event in accordance with their wishes. While cancelling or setting new reminder alarms, the event-specific alarm information is obtained from built-in database. When user makes changes to saved event, firstly the event-specific prearranged alarms are cancelled and related alarm information is deleted from database. Secondly the new alarms are set with respect to new parameters. The new event specific alarm information is also stored in database for subsequent changes or update process of reminder alarms.

Table 3.2: Reminder Pattern Example

	Reminder Pattern	Event Date	Event Location	Reminder Start Time	Reminder Start Distance
1	Homework Pattern	18.01.2013	School	60 mins.	-
2	Cafeteria Pattern	19.01.2013	Cafeteria	-	500m
3	Library Pattern	21.01.2013	Home	1 day	100m

A reminder pattern example is given in table 3.3. According to this pattern, when user walks around cafeteria equal or under 500m distance, the user is reminded with cafeteria menu. In the same way, homework pattern will alert user 60 minutes before the homework deadline. In summary, when the conditions are met that is specified with reminder pattern, and then application alert the user. The notice of reminders can be a voice, a vibration or a visual according to user request.

3.2 Major Components of Proposed Model

The proposed model consists of three main components. The first one is publishers which are also known as reminder notifiers or information providers. The second one is the dispatcher, performing duties such as distribution of published events to subscriber, keeping all the information with related to all the subscribers etc. and the third one is subscribers which are also known as reminder receivers or an information consumers. The main components of proposed model are shown in Figure 3.5.

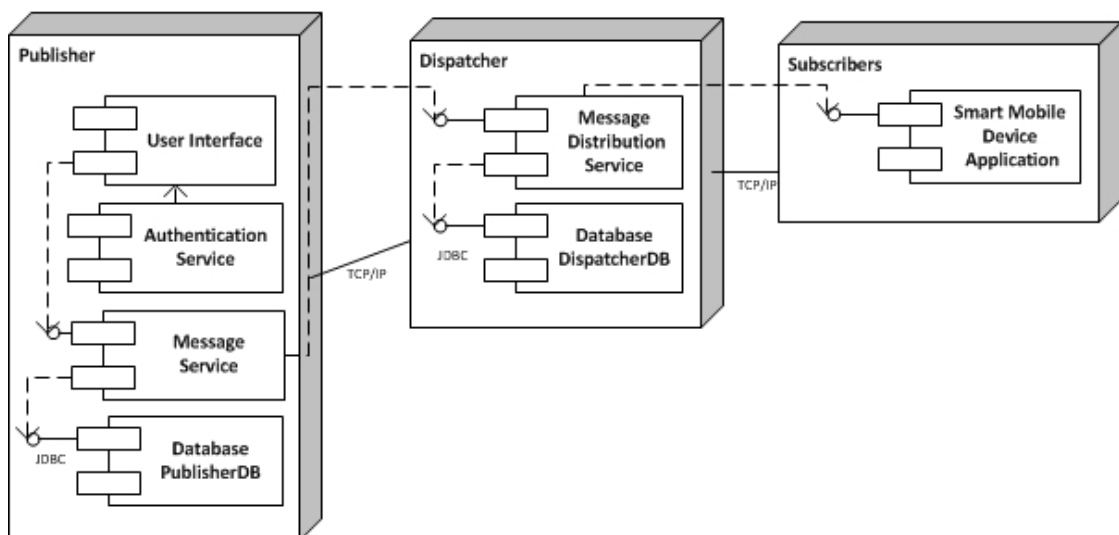


Figure 3.5: Deployment Diagram of Context Aware Reminder System

In the following, a detailed description of these components will be given.

3.2.1 Publishers

In a nutshell, first component of the proposed model is Publisher and it performs duties such as creating events, showing sent, cancelled or archived events separately and canceling a created event.

Publisher consists of a publisher application and a database. Publisher application is used for creating events. By means of a web application, a user can easily access to system via the Internet and create events. Authentication Service and Message Service are the two subcomponents of the publisher application. Database is used for storing and archiving events.

Proposed model implements an authentication service for verifying the identity of user. The users need to be authenticated for accessing message service. If the user access the system for the first time needs to sign up for context aware reminder system.

Message service is the main unit of the Publisher component. Its main functions are to create and send event to dispatcher component. The event is created easily via the message service and stored for control purposes. Event content has generic structure that can be used by several different types of publishers. Event content is detailed in Table 3.2.;

Table 3.3: Event Content

Message Field	Explanation
<i>Event Title</i>	The title of event.
<i>Author Name</i>	Publisher Name.
<i>Author E-Mail</i>	Publisher E-mail.
<i>Topic</i>	The topic of the event.
<i>Subject</i>	Short explanation of event.
<i>Event Date</i>	Date is used in reminder pattern while triggering the reminder.
<i>Reminder Pattern</i>	Specific pattern. Reminder is triggered with respect to this pattern by considering event date and location.
<i>Event Location</i>	Place is used in event while triggering the reminder.
<i>Event UUID</i>	Specific ID is used to uniquely identify the event.

The most significant feature of proposed model is to notify user according to “Reminder Pattern” which is sent by the publisher. There are default reminder patterns according to the

publisher specification such as Homework reminder pattern, Library reminder pattern, Cafeteria reminder pattern etc. Reminder pattern determines when and where to prompt reminder application.

The sent event can be canceled in case the user sends incorrect event. Message service provides cancellation function under the sent events. Users can access sent events and can cancel an event based on their wishes. Cancellation information is sent to subscribers and event is deleted from event list of subscribers.

3.2.2 Distribution of Events

The second component of the proposed model is dispatcher. Dispatcher consists of an application server and a database. Application server is used for receiving events from Publishers and delivering the published events to Subscribers. Message Distribution Service is a sub-component of the application server. Database is used for storing events and storing information of registered Publishers and Subscribers.

Message Distribution Service is the basic unit of the Dispatcher component. Its main functions are receiving events from Publishers and sending the published events to the correct Subscribers. The primary characteristics of dispatcher are listed as follows:

- It collects events notified by publishers and stores them in a database.
- It handles all parts of queuing of events and sending to target mobile device application.
- The mobile device application does not need to be running to receive events. The dispatcher wakes up the application when the events arrive.
- It passes events to the right application and smart mobile device application has full control of how to handle it.
- It can either send events to a single device or to multiple mobile devices simultaneously. (Multicast event)
- Users send subscription request by using mobile application. Dispatcher handles requests and store subscription information in order to use when sending events.
- It stores all publishers and subscribers' information in database.
- The topics to be subscribed are queried from the dispatcher and they are presented to the user's choice by the mobile device application.

In substance, Dispatcher can be considered as a bridge between publishers and subscribers and it helps us to deliver the published events to subscribers' smart mobile devices.

3.2.3 Subscribers

The third component of the proposed model is Subscriber. Subscriber consists of hardware and software. As hardware, Smart mobile devices are used in proposed model for delivering events to user at the right time and at the right place. As software, mobile application is developed for receiving events and notifying users.

3.2.3.1 Hardware

Proposed model targets to build a context aware reminder system and in this context, smart mobile devices provide a potentially suitable and fittingly context aware platform for the discovery of personal context especially location as well as the delivery of reminders.

Today most of the users take their smartphones with them almost everywhere. Location awareness is one of the important features of mobile applications. Sensors in mobile devices assist to gather context information that provide much more individual / personal and precise information with related to the users. Knowing context information and using this information cleverly bring a better service to the user.

The smart mobile device which is used in proposed model must have a GPS navigation unit and a wireless adapter. GPS sensor is basically used to know where the user is presently located and wireless adapter is mainly used for Internet access and data exchange. More than that, wireless adapter can also be used to determine current location of user. Basically, the usage of GPS sensor and wireless adapter makes smart mobile application location aware and location-based reminders can be set with this feature.

3.2.3.2 Software

Application which is running on the smart mobile device is the main unit of the Subscriber component.

Users need to register first in order to use the mobile application. After the registration process, the user performs all tasks via this application. As the first step, user needs to subscribe to a topic; in order to receive event with related that topic. After subscribing on a topic, user gets events for this topic. The events are received automatically by application and user can check events whenever they want. Reminder alarms are set with respect to reminder pattern when the event is received. The information of registered alarms is stored for subsequent changes or update process of reminder alarms.

The reminder alarms which are set when the event is received are triggered when the event-related conditions are satisfied. If user edits the event to change the reminder pattern, the previously set reminder alarms are cancelled and new reminder alarms are set with respect to new pattern. The primitive flowchart of the smart mobile device application is illustrated in Figure 3.6.

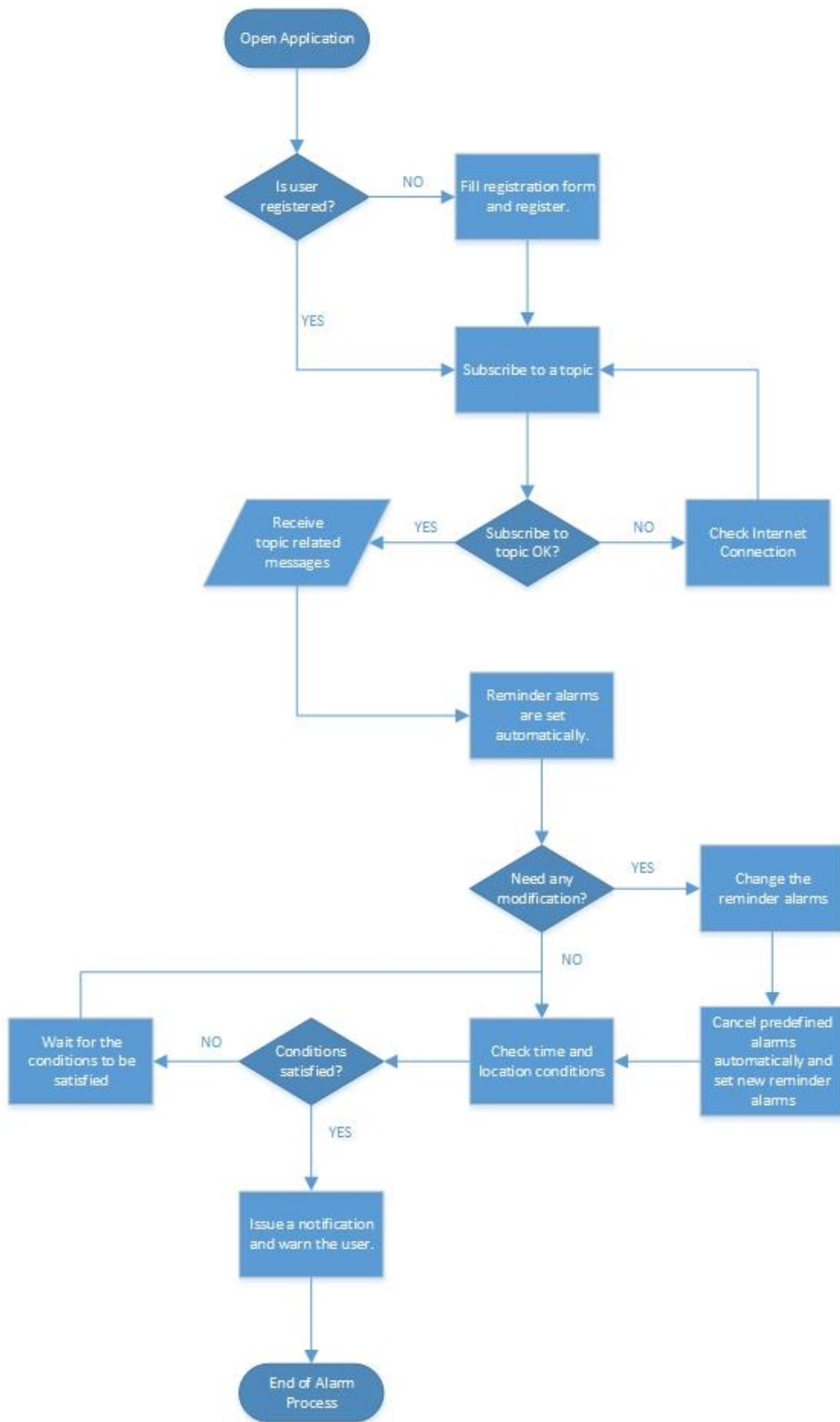


Figure 3.6: The flowchart of the subscriber application.

Events are sent to the smart mobile device whenever new event is available instead of polling by mobile application to dispatcher. This enhances user experience and saves a lot of battery power.

There are some default reminder patterns which are defined by publishers. In addition, the system enables user to change and add new reminder pattern for their interests.

3.3 Typical Scenarios

In this section, typical usage scenarios of our proposed system is given.

3.3.1 Scenario 1

Hakan does not want to miss the deadline for his homework or project.

Hakan subscribes to the publisher of his homework or project and he gets reminder messages when there is an announcement with related to his homework and projects. When there is an announcement or change in the status of lecture, the reminder application running on mobile device notifies him before the lecture date and time. For example, if the deadline for submissions of a project is changed, the reminder application should notify Hakan before the deadline.

3.3.2 Scenario 2

Burcu wants to get notification message when her lecture is cancelled or postponed.

Burcu subscribes to the publisher of her lessons and she gets reminder messages when there is an announcement. The reminder application running on mobile device notifies Burcu before the lecture date and time.

CHAPTER 4

4 IMPLEMENTATION

In this chapter, a prototype system has been developed to demonstrate the applicability of context-aware reminder system proposed Chapter 3. This prototype system consists of server-side components which are used to create and send a reminder event and an android application which is used to retrieve and display the reminder event.

4.1 System Design

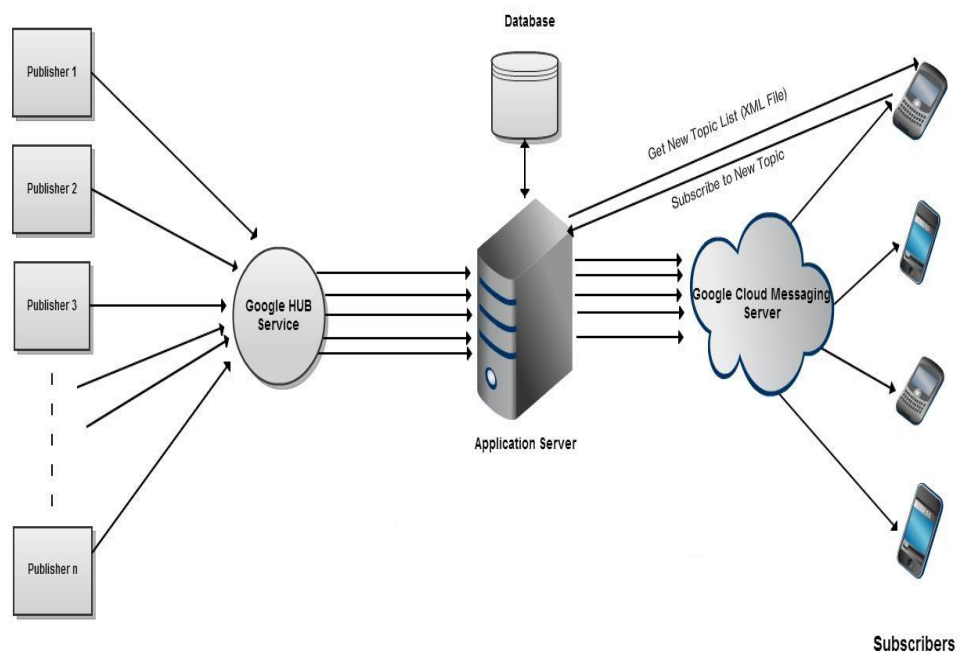


Figure 4.1: The general structure of the prototype developed

The prototype consists of three main components and two contributing components. The main components are Publishers, Subscribers and the dispatcher. The contributing components are Google Hub Service and Google Cloud Messaging Service. The general structure of prototype is shown in Figure 4.1.

4.1.1 PubSubHubbub Protocol and Google Hub Service

In this prototype implementation, Pubsubhubbub (“Pubsubhubbub”, n.d.) is used as the dispatcher. Pubsubhubbub is simple, open, server to server publish / subscribe protocol that extends Atom and RSS for data feeds. The main contribution of Pubsubhubbub is to provide near-instant notifications when a feed is updated. In classic approach, a client periodically polls the feed server at some arbitrary interval. Basically, Atom and RSS update notifications are pushed to subscribers through Pubsubhubbub rather than needing clients to poll whole feeds.

Briefly, this protocol works as follows:

- A feed URL (topic URL, publisher URL) declares its Hub server in its Atom XML file (RSS).
- A server that’s interested in a topic (subscriber server) subscribes to the Topic URL by means of Topic URL's declared Hub.
- The publisher software pings Hub when the publisher next updates the Topic URL.
- The hub efficiently fetches the published feed and multicasts the updated messages to all registered server.

4.1.2 Google Cloud Messaging for Android

In this study, we used Google Cloud Messaging for Android (“Google Cloud Messaging for Android”, n.d.) to send events from Application server to subscribers’ Android-powered device. GCM is a service that allows sending data from web servers to Android devices, and vice versa. The service handles everything to deliver the messages send to the Android application, and receive the messages send from the 3rd-party application servers. GCM does not incur any fee no matter how big the messaging traffic is. It can send messages to a single device or to multiple mobile devices simultaneously.

Contrary to the classical message delivery system “pulling”, checking for new messages once in a while, GCM uses “push” system which delivers messages in real time and saves battery.

Primary characteristics of Google Cloud Messaging are:

- 3rd-party application servers can send messages to their Android applications.
- GCM Cloud Connection Server enables receiving upstream messages.
- Android application does not need to be running in order to receive the message. Android will wake up the application if the application is set up properly.
- GCM passes raw data into the Android application. There is no built in user interface to handle the message.

- Requires Android 2.2 or higher and Google Play Store installed.
- Uses existing connection for Google services.

The lifecycle of Google cloud messaging is described in Figure 4.2 and the explanation is given shortly below the figure.

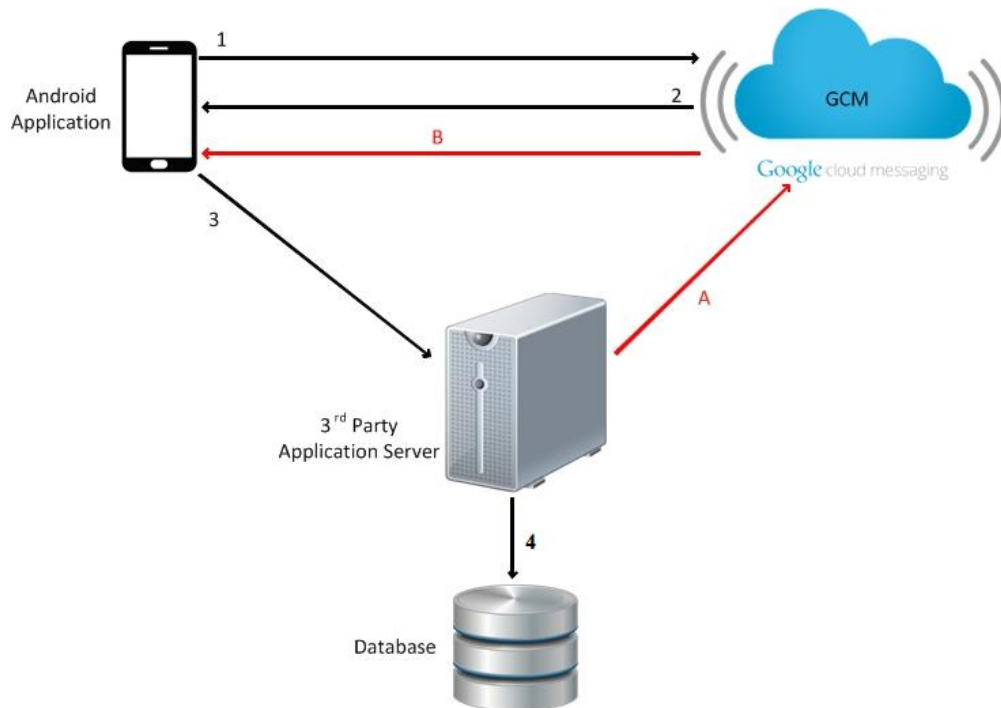


Figure 4.2: How GCM works

- (1) The first time the android application running on smart mobile device registers to GCM server by firing off a registration intent. Registration intent consists of the “sender ID”, and the “Android application ID”.
 - (2) Google cloud messaging service sends “registration ID” to android device after successful registration.
 - (3) Android device will send “registration ID” to 3rd-party application server upon receiving it.
 - (4) 3rd-party application server store Registration ID in the database for future usage.
- (A)** The server sends a message to Google cloud messaging service with “registration ID” which is stored in the database, when the push notification is required.
- (B)** By using device “registration ID” Google cloud messaging service will deliver the message to related Android powered mobile device.

4.1.3 Publishers

The main tasks of Publisher component are creating events, showing sent, cancelled or archived events separately and canceling a created event. Publisher consists of a web server and a database. In publisher implementation, PHP language (PHP: Hypertext Preprocessor) is used for server side programming. PHP language is preferred because it is free and widely-used and it is a strong tool for developing active Web pages. MySQL database is used for storing information.

Authentication Service and Message Service are the two sub-components of the web server. Authentication service is used to ensure the security of the system and it allows only authorized users to login to the system. Message service is used to handle everything related with events such as

- Event Creation,
- Defining Reminder Pattern and Reminder Templates,
- Displaying Sent Events,
- Displaying Cancelled Event and Displaying Event Archive,
- Displaying Defined Locations,
- Defining New Location.

4.1.3.1 Event Creation

A simple editor page is created to help publishers create an event. With this editor, publishers can easily create an event. For event creation several PHP pages is used. The most important of these are onder.php and d2xml.php. The onder.php page is used to display editor page and it includes suggest_topic.php, suggest_pattern.php, suggest_location.php, and connect_metu.mobi_2.php (source codes can be found in the CD attached). The suggest_topic.php ensures to get a certain topic which is belonging to a specific user. The suggest_pattern.php allows users to select reminder pattern from defined reminder patterns. The suggest_location.php helps users to select defined locations by using JQuery and autocomplete. The connect_metu.mobi_2.php opens and closes connection to database and mainly handle database connections.

Figure 4.3: Editor Page

The Figure 4.3 shows the editor page. Event Date defines the date on which event reminders expire. Location defines the event location is where the proximity alert is set with respect to it. Subject and Event Title are filled according to the event properties.

In the background, the editor page creates Atom feed file by using `d2xml.php` and notifies Hub Server about updates by posting Atom feed URL to the Hub; Hub pulls the feed again to find new entries. When Hub receives new message it posts event to the Subscriber's endpoint Application Server URL (for this implementation it is `callback.php`). The `d2xml.php` includes `publisher.php` and `UUID.php`. The `UUID.php` is used to create an UUID number and this number is included in event to ensure the uniqueness of each event. The `publisher.php` contains Hub URL address and Topic URL address. When an event is submitted, `publisher.php` notifies Hub Server about updates.

4.1.3.2 Defining Reminder Pattern and Reminder Templates

Publishers can use “define pattern” menu to display previously defined reminder patterns and the elements of the reminder patterns which are reminder templates. Furthermore, publishers can define new reminder patterns and new reminder templates and they can also delete previously defined patterns and templates by using “define pattern” menu. Define pattern page is illustrated in Figure 4.4 and Defining Reminder Template menu is illustrated in Figure 4.5.

Home Create Event Define Pattern Sent Events Cancelled Events Archieve Locations Define Location

Logout

Defining Reminder Pattern

Reminder Pattern Name	Show	Delete
Library	Show	Delete
Lecture	Show	Delete
Homework	Show	Delete

New Pattern Name:

Submit

Defining Reminder Template

Reminder Template Name	Start Time	End Time	Radius	Location	Position	Action	Delete
Remind Me	0 Day, 3 Hour, 0 Minute Ago.	0 Day, 2 Hour, 0 Minute Ago.	5000	Event Location	Inside	Notification	delete

Figure 4.4: Define Pattern Page

The reminder_pattern.php is used to handle all reminder pattern operations and it includes insert_reminders.php and delete_reminders.php pages. The insert_reminder.php page steps in when new reminder patterns or reminder template is added and it stores the required values to the database. The delete_reminder.php page allows publishers to delete both reminder pattern and reminder templates from database.

Home Create Event Define Pattern Sent Events Cancelled Events Achieve Locations Define Location Logout

Defining Reminder Pattern

Reminder Pattern Name	Show	Delete
Library	Show	Delete
Lecture	Show	Delete
Homework	Show	Delete

The Reminder Pattern is added. :)

New Pattern Name:

Defining Reminder Template

Reminder Template Name:

Start Time: Day Hour Minute

End Time: Day Hour Minute

Radius: meters

Location:

Inside / Outside:

Action:

Figure 4.5: Defining Reminder Template

4.1.3.3 Displaying Sent Events

Publishers can display events sent before. Such events are ordered by event creation date. Sent Events page is illustrated in Figure 4.6.

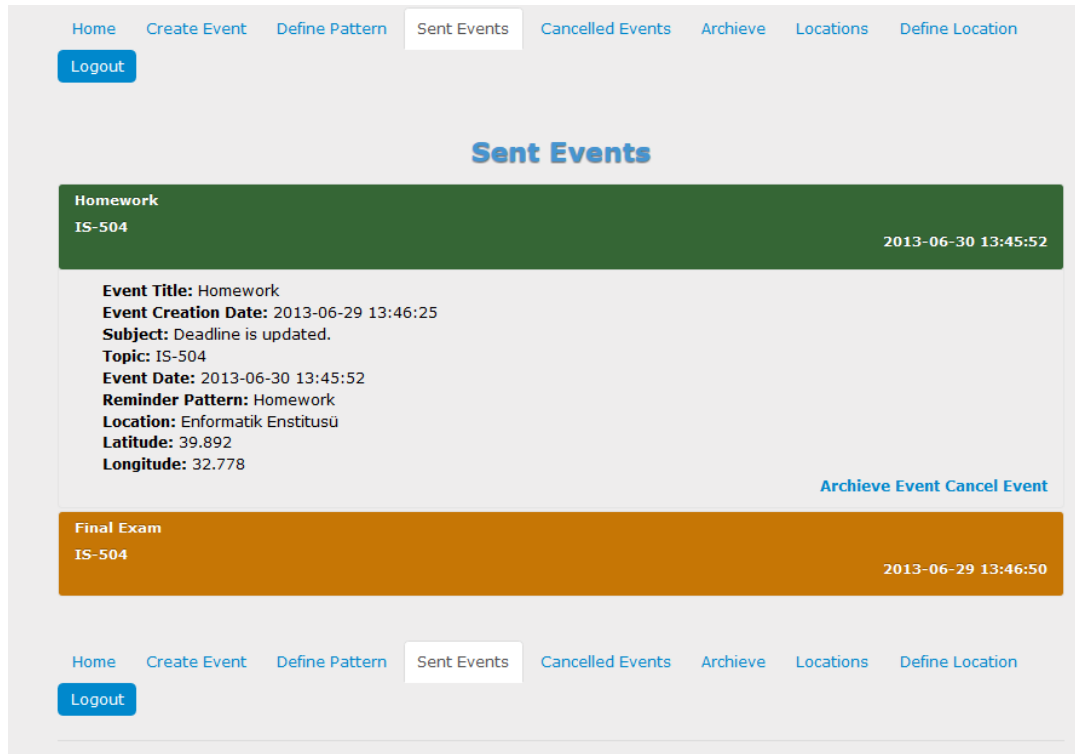


Figure 4.6: Sent Events Page

Publishers can cancel an event or send it to the archive. It is sufficient to click on the “Cancel Event” link to cancel a sent event and to click on the “Archieve Event” link to send an event to Archieve. In cancellation operation, cancel request is also sent to subscribers and after that event is deleted and related reminders are cancelled automatically at subscribers side.

The `show_stored_reminder_messages.php` is used to display sent events and it includes `cancel_message.php` and `archive_send.php`. The `archive_send.php` is used to send an event to archive and in the background; it updates event condition as archived at the database. The `cancel_message.php` handles all event cancel operations and it updates event condition as cancelled at the database.

4.1.3.4 Displaying Cancelled Event and Archived Events

Publishers can also display cancelled events and archived events whenever they want. Cancelled events are ordered by event cancellation date and archived events are ordered by event archive date. Cancelled Events page is illustrated in Figure 4.7 and Archived Events page is illustrated in Figure 4.8.

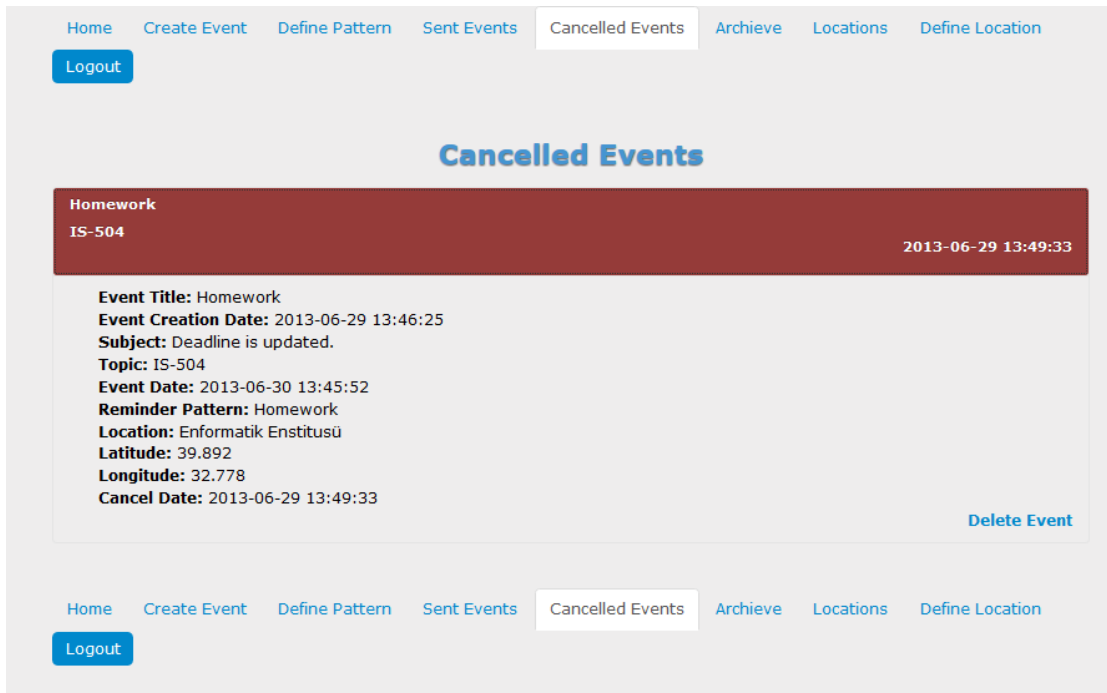


Figure 4.7: Cancelled Events Page

Publishers can delete an event from Cancelled Events or Archived Events. It is sufficient to click on the “Delete Event” link to delete an event.

The `show_cancelled_messages.php` is used to display cancelled events and the `show_archieved_messages.php` is used to display archived events and they include `delete_message.php`. The `delete_message.php` deletes related events from database.

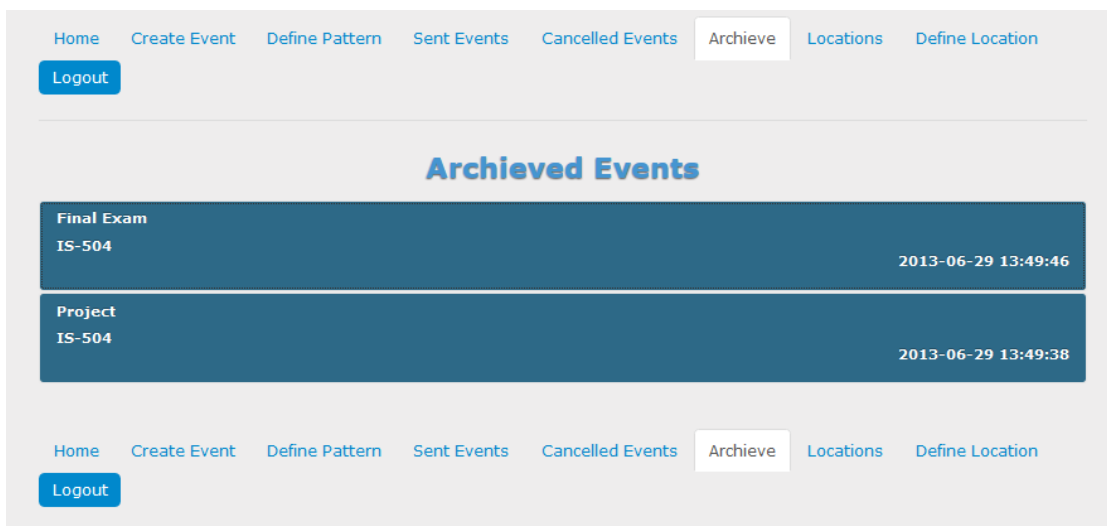
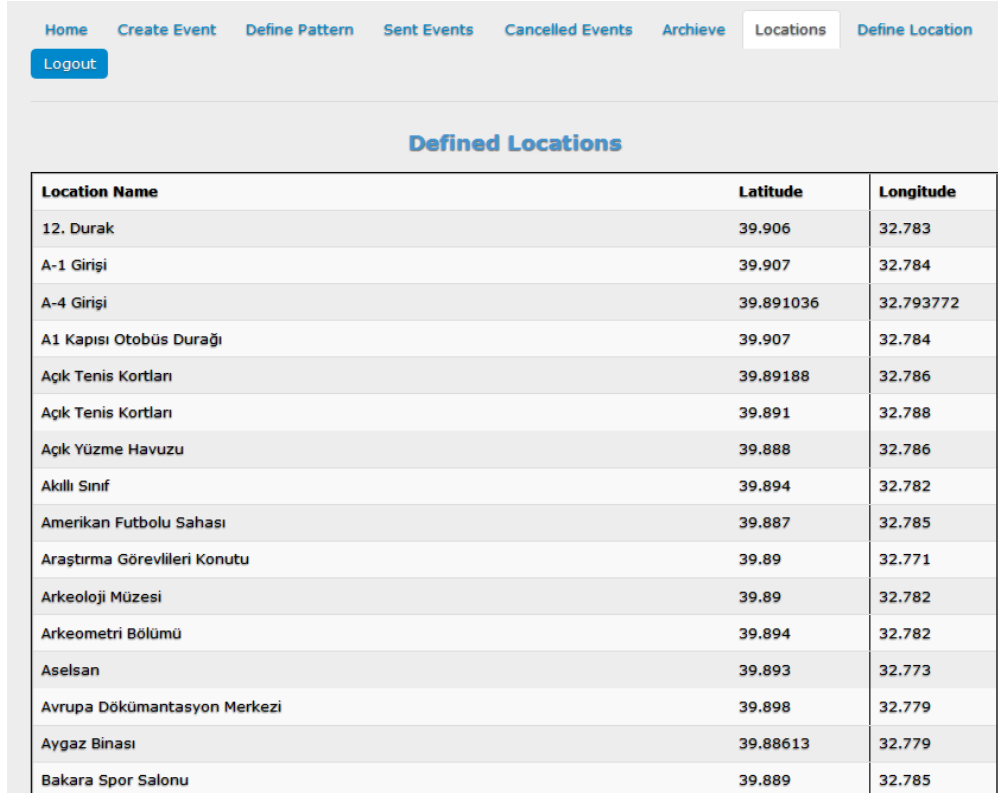


Figure 4.8: Archived Events Page

4.1.3.5 Displaying Defined Locations

The coordinates of the main points of METU are stored in database. Publishers can display defined locations by just clicking on the “Locations” link. Locations are ordered by location names. These locations can be used while creating event. Location page is illustrated in Figure 4.9.



Location Name	Latitude	Longitude
12. Durak	39.906	32.783
A-1 Girişi	39.907	32.784
A-4 Girişi	39.891036	32.793772
A1 Kapısı Otobüs Durağı	39.907	32.784
Açık Tenis Kortları	39.89188	32.786
Açık Tenis Kortları	39.891	32.788
Açık Yüzme Havuzu	39.888	32.786
Akıllı Sınıf	39.894	32.782
Amerikan Futbolu Sahası	39.887	32.785
Araştırma Görevlileri Konutu	39.89	32.771
Arkeoloji Müzesi	39.89	32.782
Arkeometri Bölümü	39.894	32.782
Aselsan	39.893	32.773
Avrupa Dökümantasyon Merkezi	39.898	32.779
Aygaz Binası	39.88613	32.779
Bakara Spor Salonu	39.889	32.785

Figure 4.9: Locations Page

The location_show_detail.php is used to display locations and it includes the connect_metu.mobi_2.php to handle database connections.

4.1.3.6 Defining New Location

Publishers can define new locations by just clicking on “Define Location” link. Location Name, Latitude and Longitude values are required to define locations. Define Location page is illustrated in Figure 4.10.

Figure 4.10: Define Location Page

The `location_add.php` is used for defining location. This page gets required data and stores them into database.

4.1.3.7 Database Structure of Publisher

Publisher component's database contains five tables. These are Users, `atom_feed_reading`, `Reminder_Template`, `Reminder_Pattern`, `Locations`. Users table stores account information for each user. The most important fields are email, username, and password. Username and password is used for authentication, and email is used when it is needed to change the password. The Users table is illustrated in Figure 4.11.

Users	
<code>id_user</code>	: int
<code>name</code>	: varchar(128)
<code>email</code>	: varchar(64)
<code>username</code>	: varchar(16)
<code>password</code>	: varchar(32)
<code>confirmcode</code>	: varchar(32)

Figure 4.11: Users table

Atom feed reading table is used to store Events created by the publisher. Here username is the publisher's username. Cancel UUID is randomly created by the `UUID.php` page for later use, to be used to send cancelation message. Archive indicates the three states of the event: 0 has no meaning and it is default, 1 represent that the event is canceled, 2 means that the event is "archived". The `atom_feed_reading` table is illustrated in Figure 4.12.

atom_feed_reading
id: int
feed_id: varchar(255)
feed_title: varchar(255)
feed_updated: datetime
feed_subtitle: varchar(255)
feed_author_name: varchar(255)
feed_author_email: varchar(255)
entry_id: varchar(255)
entry_title: varchar(255)
entry_updated: datetime
entry_category: varchar(255)
entry_source: varchar(255)
entry_topic: varchar(255)
entry_info1: varchar(255)
entry_info2: varchar(255)
entry_info3: varchar(255)
entry_info4: varchar(255)
entry_info5: varchar(255)
entry_info6: varchar(255)
entry_event_date: datetime
entry_reminder_pattern: varchar(255)
entry_receiver: varchar(255)
entry_location: varchar(255)
cancel_uuid: varchar(500)
username: varchar(50)
achieve: tinyint
achieve_date: datetime

Figure 4.12: atom_feed_reading Table

Reminder specifications are stored in Reminder_Template table such as Start and End Time of reminder, Radius etc. One Reminder pattern can include many reminder templates. The *Reminder_Template* and *Reminder_Pattern* tables are illustrated in Figure 4.13.

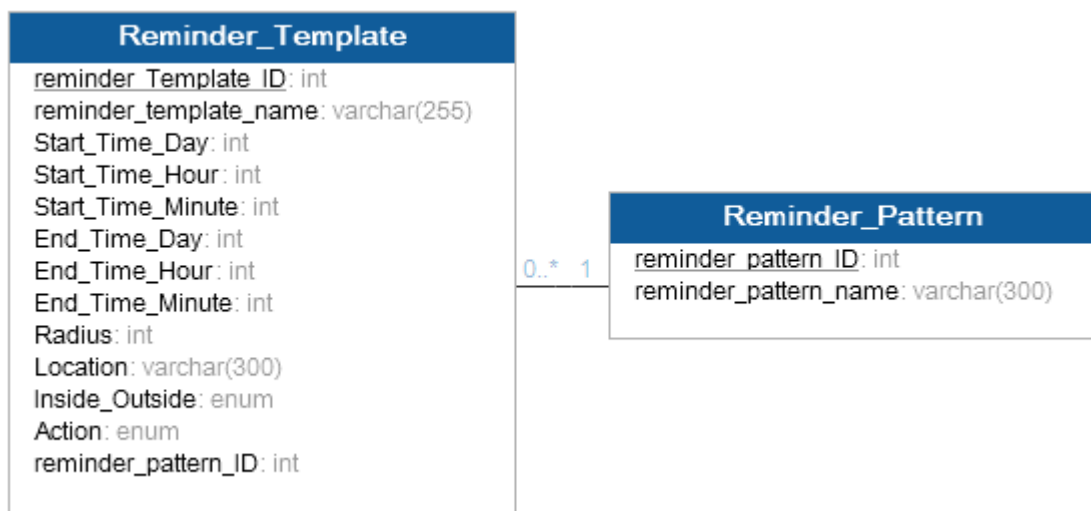


Figure 4.13: Reminder_Template and Reminder_Pattern Tables

In Locations table, event locations are predefined. The table has longitude, latitude, and name of the location fields. The longitude and latitude values are stored as “Double” data type.

Locations	
<u>map_info_id</u>	int
location_name	varchar(255)
longitude	double
latitude	double

Figure 4.14: Location Table

4.1.4 Distribution of Events (Application Server)

The main mission of Application server is to receive events from Publishers and deliver the published events to Subscribers. In this prototype implementation, we used PHP as server side programming and MySQL as a database which is used for storing events and storing information of registered Publishers and Subscribers.

Message Distribution Service is a sub-component of the application server and it steps in when one publisher send an event. Hub Service is notified about updates of the topic URL when an event is sends by a publisher. After that, Hub Service fetches the published feed and multicasts events to the registered application server. The *callback.php* receives the messages from the PubHusbSubbub server in XML format. This message is parsed into raw text with the help of XMLparser. The raw texts are recorded to associated 24 fields of the *atom_feed_reading* table. *atom_feed_reading* Table is illustrated in Figure 4.15.

atom_feed_reading
id: int
feed_id: varchar(255)
feed_title: varchar(255)
feed_updated: datetime
feed_subtitle: varchar(255)
feed_author_name: varchar(255)
feed_author_email: varchar(255)
entry_id: varchar(255)
entry_title: varchar(255)
entry_updated: datetime
entry_category: varchar(255)
entry_source: varchar(255)
entry_topic: varchar(255)
entry_info1: varchar(255)
entry_info2: varchar(255)
entry_info3: varchar(255)
entry_info4: varchar(255)
entry_info5: varchar(255)
entry_info6: varchar(255)
entry_event_date: datetime
entry_reminder_pattern: varchar(255)
entry_receiver: varchar(255)
entry_location: varchar(255)
cancel_uuid: varchar(500)
username: varchar(100)

Figure 4.15: Atom_feed_reading Table

After this process, send_multievent_publisher.php is called to deliver the event to the subscribers via Google Cloud Service. It queries the “atom_feed_reading” table to get the event content, as well as “publish”, “publishsubscribe”, and “subscriber” tables to get the associated subscriber in order to deliver the correct event to the correct subscribers. Association between “subscriber”, “publish”, and “publishsubscribe” tables is illustrated in Figure 4.16.



Figure 4.16: Association Between subscriber, publish, and publishsubscribe Tables

Publisher information is stored in the *publish* table that consists of three fields. These fields are *publisher_id*, *publisher_topic* and *username*. Here *publisher_id* is auto incrementing integer, *publisher_topic* is unique text to identify the topic to be subscribed, and *username* is the publisher's account name.

The topics to be subscribed are queried from the *publish* table and they are presented to the user's choice by the mobile device application. When a user is subscribed to a topic, this info will be stored in "publishsubscribe" table and "subscriber" table.

The information of the registered users is stored in the "gcm_user" table. The "gcm_user" table consists of six fields: "id", "gcm_regid", "name", "email", "created_at" and "user_type". A registration ID is assigned to mobile device as a parameter during the registration process and it is stored in the "gcm_regid" field. The registration ID is used while sending reminder messages to the mobile devices via the application server. The gcm_users table is shown in Figure 4.17.



gcm_users	
id:	int
gcm_regid:	text
name:	varchar(50)
email:	varchar(255)
created_at:	timestamp
user_type:	set

Figure 4.17: gcm_users Table

4.1.5 Subscribers

In the prototype implementation, we used Java for the client side programming on Android powered device. The application receives subscribed events from publishers automatically and notifies users when the predefined conditions are satisfied. The received events are stored in the built-in database. SQLite database ("*SQLiteDatabase*", n.d.) is used for saving data.

Home Page of Application and Event List Page is illustrated in Figure 4.18.

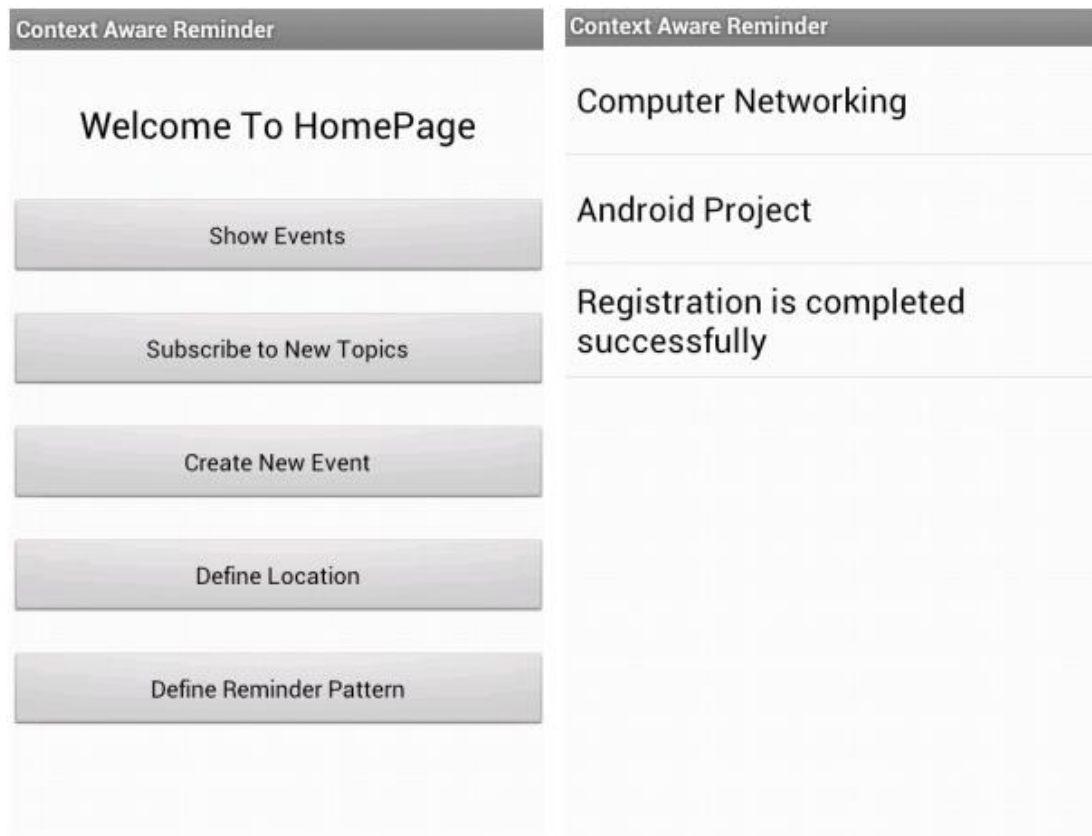


Figure 4.18: Home Page of Application and Event List Page

Android powered application provides five main features: Show events, Subscribe to New Topics, Create New Event, Define Location and Define Reminder Pattern.

Show Events feature shows the saved events as a list of event headers. When the event header is selected, the details of event are displayed in a new page. In order to make changes to the stored event, the users should select “Edit Message” option. After making changes, the user saves the event by clicking “Save Message” button. If the users want to delete the unwanted event, they should select “Delete Message” option. When a user makes changes, the predefined reminders are cancelled and new reminders are set with respect to event properties. Event Details Page is illustrated in Figure 4.19.

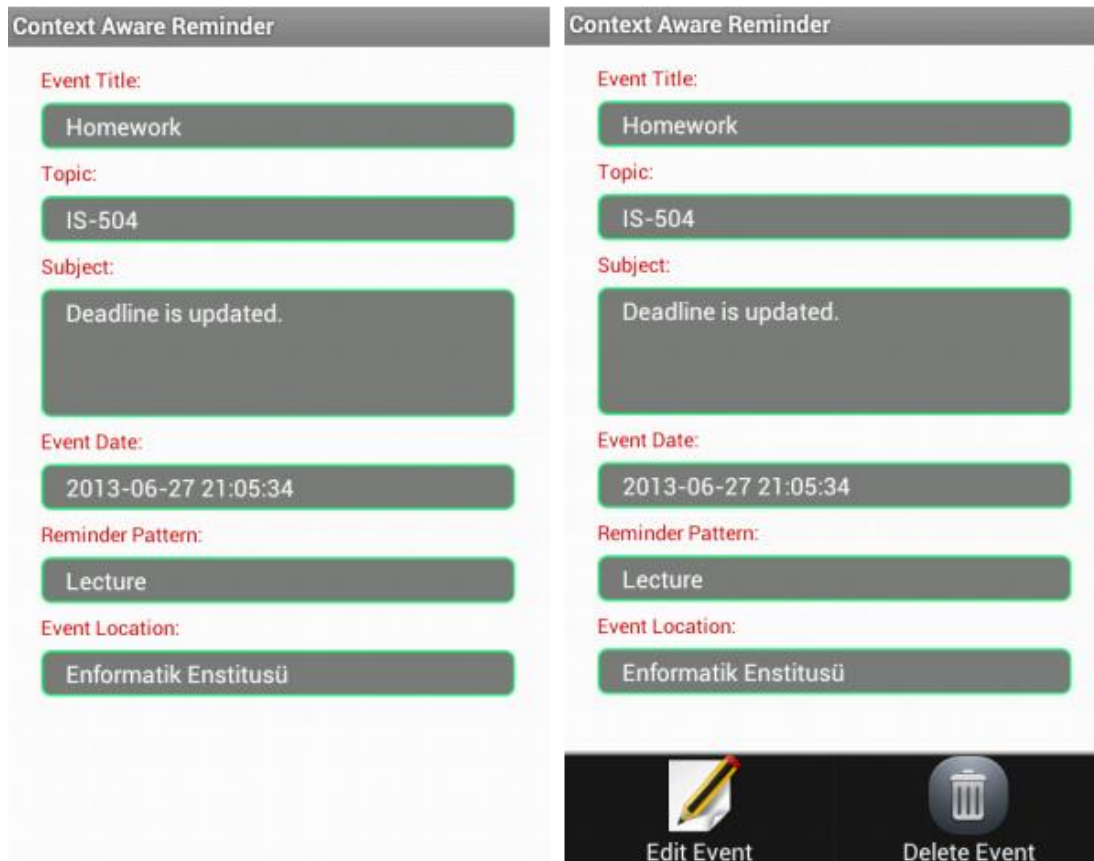


Figure 4.19: Event Details Page

The `GCMIntentService` class is used to retrieve the event from Google Cloud Messaging Service. The `DatabaseConnector` and the `DatabaseOpenHelper` classes are used to do database operations such as insertion, update and deletion of events. The `Message_List` class allows application to display events in the form of list view. The `View_Message` class steps in when a user wants to see the event details.

Subscribe to New Topics feature allows users to subscribe to a topic in order to receive events related with that topic. Users can subscribe to more than one topic of interest. After the subscribing process, users get topic related events automatically via the mobile application. Subscriber Page is illustrated in Figure 4.20.

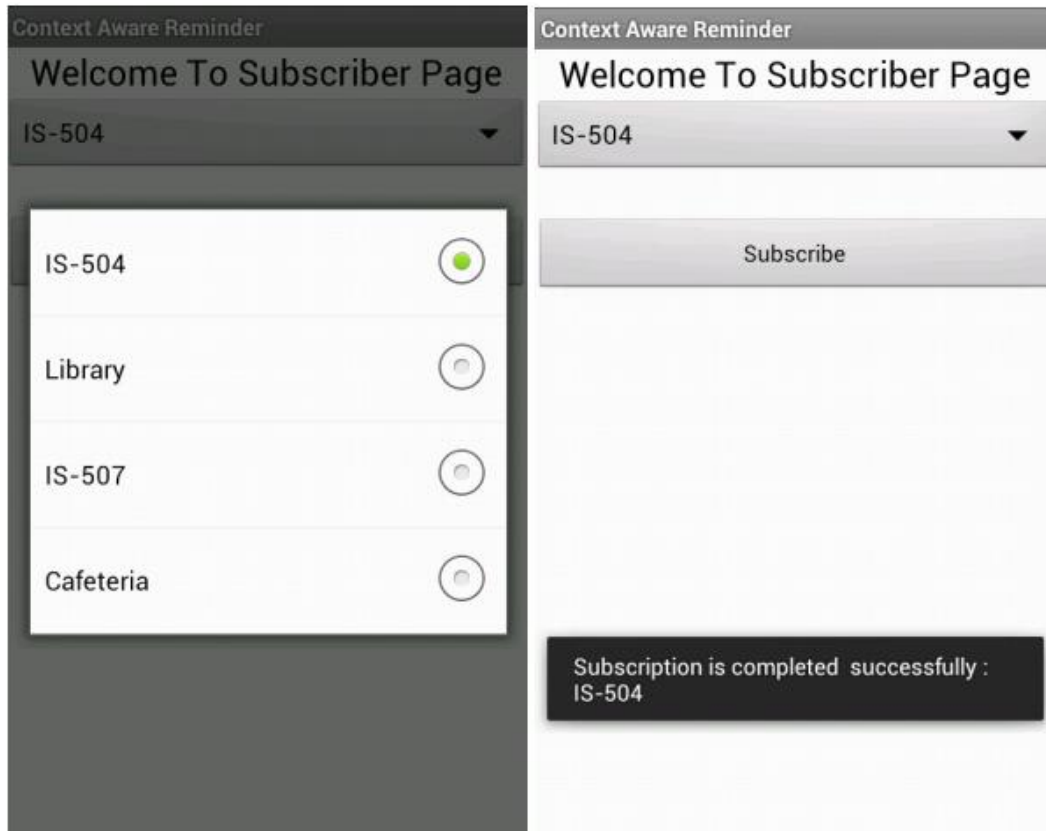


Figure 4.20: Subscriber Page

Topic list, which is used in subscription process, is stored at Application Servers' database. Therefore, when a user clicks to "Subscribe to New Topics" button, a request is sent to the application server from the mobile application. This request runs "atom_topic_list.php" code. PHP code retrieves the topic list from database and creates an XML file. After creating the XML file, application server returns back to the mobile application. Mobile application gets XML file and use XMLHandler.java, Subscriber.java and XMLGettersSetters.java classes to parse the XML file. The classes used for the Subscription Process is illustrated in Figure 4.21. After XML parsing, the topic is loaded to the dropdown menu and allows users to select the topic. Users choose one of the topics on the drop down list. Then the subscription process is completed by clicking the subscribe button.

Meanwhile, subscription request and subscriber information are sent to the application server and it handles requests and stores subscription information. After that, subscribed users receive the reminder messages, automatically.

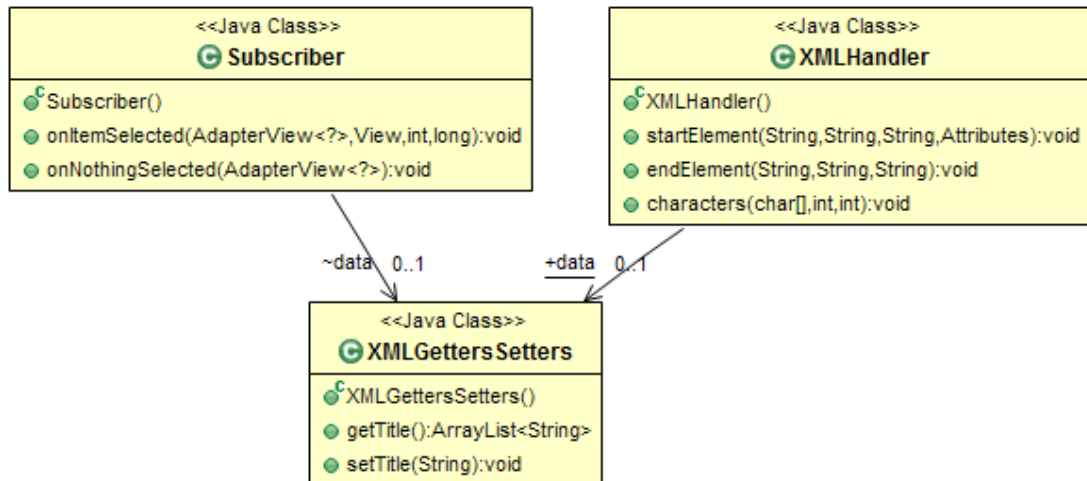


Figure 4.21: Subscription Process Classes

Create New Event features allows users to set reminders without receiving an event from a publisher or information providers. To create a new event “Create New Event” on the home screen of the application should be chosen. The required fields must be filled on the new event page. After selecting “Save Message” the event is created. New Event Page is illustrated in Figure 4.22.

Create_Event class is used to create a new event. ManagerTimeReminder and MainPage classes are used to set time and proximity alert. AlarmLocationReceiver and AlarmTimeReceiver classes step in when new event reminder requirements are satisfied and they work in notification of users. AlertDialogManager warns users if some of the required fields are not set or internet connection is not available.

The screenshot shows a mobile application interface titled "Context Aware Reminder". The form contains the following elements:

- Event Title:** A text input field with an orange border.
- Topic:** A text input field.
- Subject:** A text input field.
- Event Date:** A text input field next to a "Pick Date" button.
- Event Time:** A text input field next to a "Pick Time" button.
- Reminder Pattern:** A dropdown menu currently showing "Lecture".
- Event Location:** A dropdown menu currently showing "Enformatik Enstitüsü".
- Save Event:** A large button at the bottom of the form.

Figure 4.22: New Event Page

“Define Location” feature allows users to save current location’s latitude and longitude coordinates with a fictitious name. For example, users can save his/her homes’ coordinates as “Home” or some other name. In order to define a new location, “Define Location” button is clicked by user which is placed on the home screen of the application. The location name box is filled on the new page and by clicking Add Location button, new location is added. After that new location can be used while creating an event. Define Location Page is illustrated in Figure 4.23.

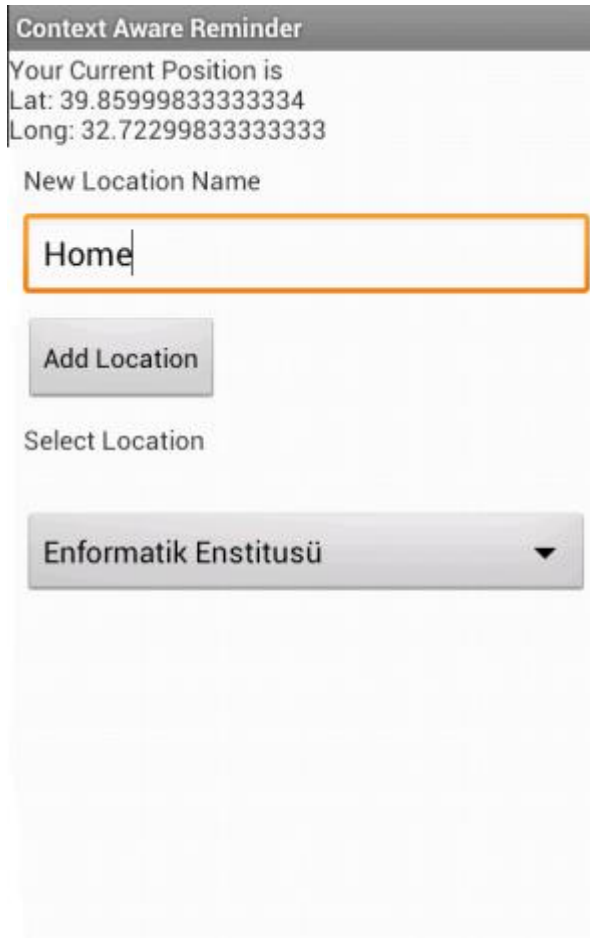


Figure 4.23: Define Location Page

Location_Add and Location_GPS_Tracking classes are used to define a new location. Location_GPS_Tracking class gets the user's current location by using the best available location provider (Network or GPS provider). Location_Add class displays current position and allows users to enter new location name and define that location.

Define Reminder Pattern feature allows users to create reminder patterns and reminder templates for their own interests. After creation of reminder patterns, users can create a new event or change an event with respect to the new reminder patterns. After that mobile application notifies user according to Reminder Pattern. Reminder Pattern Page and Reminder Template Page is illustrated in Figure 4.24.

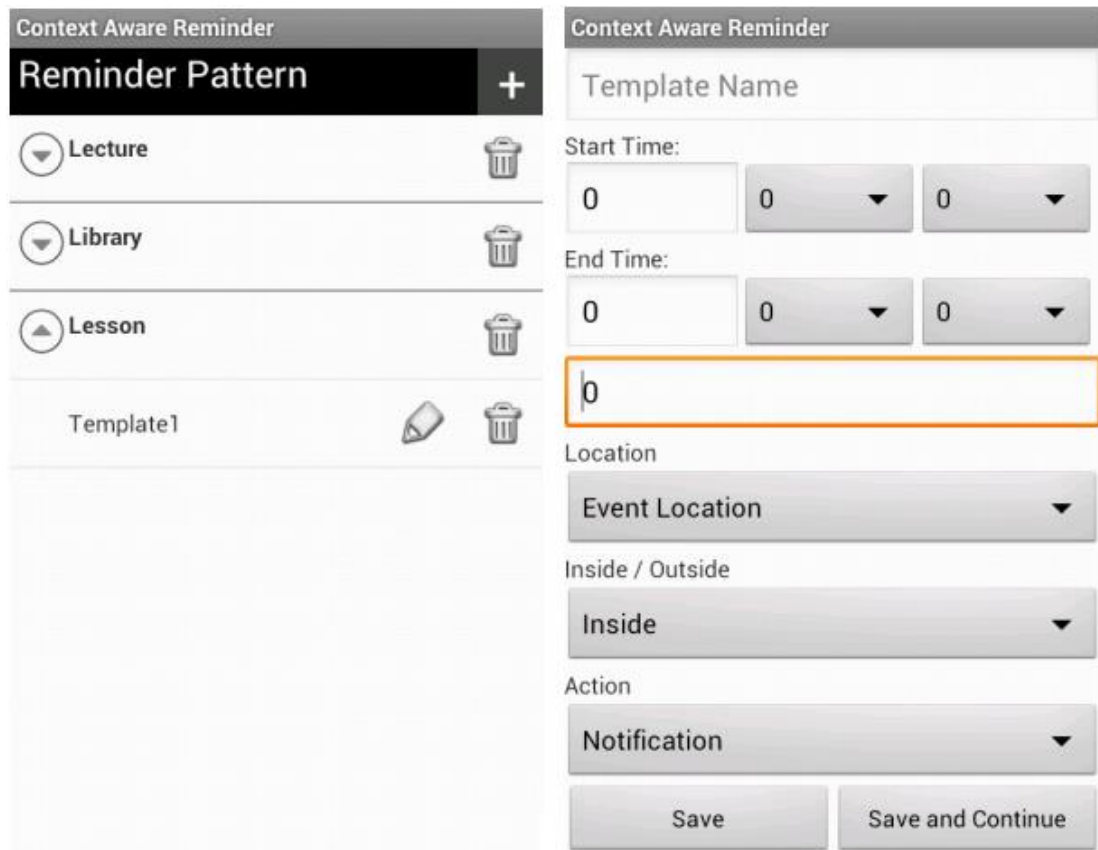


Figure 4.24: Reminder Pattern Page and Reminder Template Page

Define Reminder Pattern feature is implemented by Reminder_Pattern, Reminder_ExpandableListAdapter and Reminders classes. The Reminder_ExpandableListAdapter class allows the application to display reminder pattern and reminder templates in the same page. Reminder_Pattern and Reminders classes are used while defining reminder pattern and reminder templates.

Classes used in the Subscriber component is illustrated in Figure 4.25. Note that only significant classes are given in the class diagram.

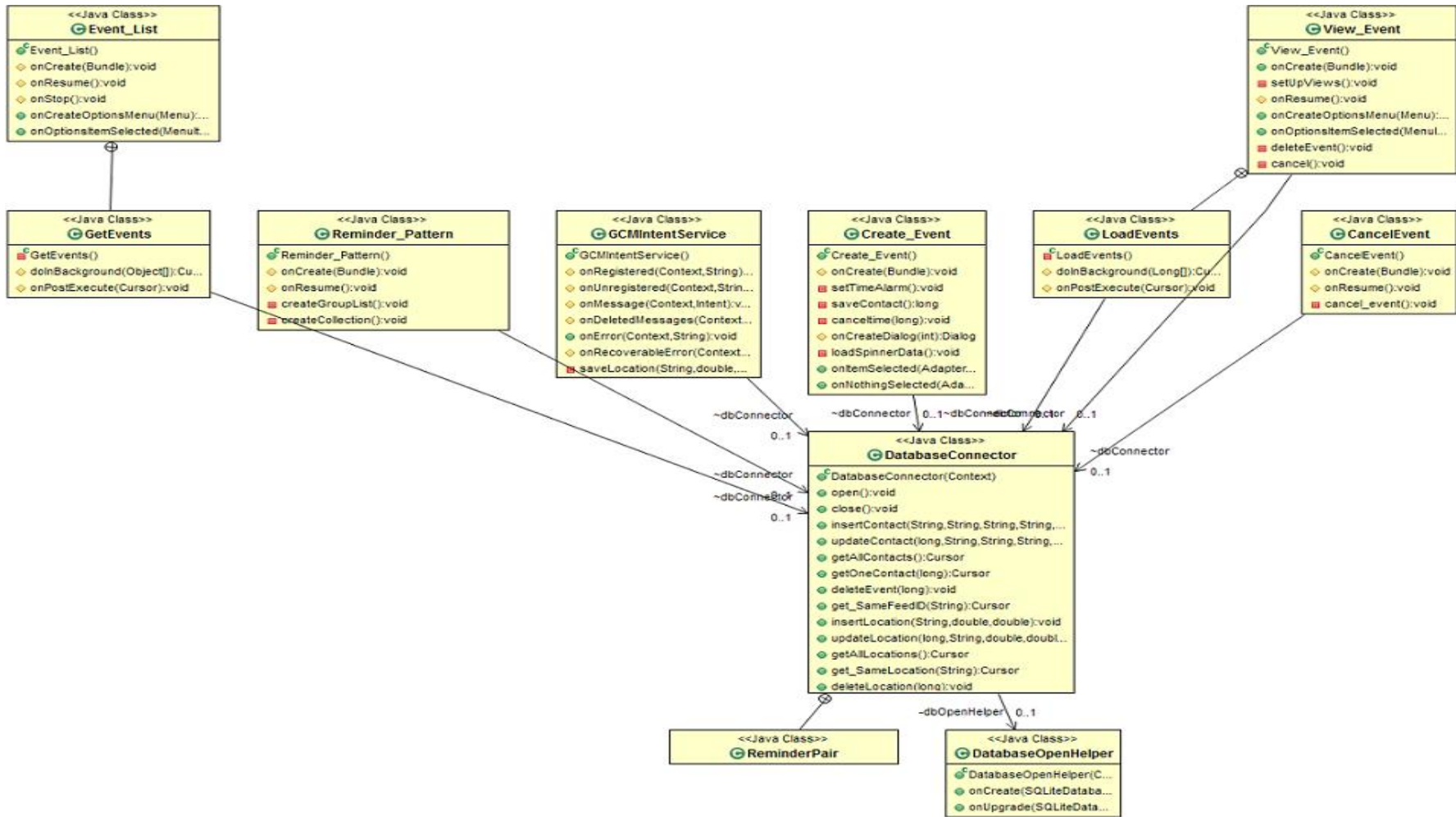


Figure 4.25: Class Diagram for the Subscriber Component

The reminder alarms which are set when the event is received are triggered when the event related conditions are satisfied. The basic control flows of event trigger mechanism is illustrated in 4.26.

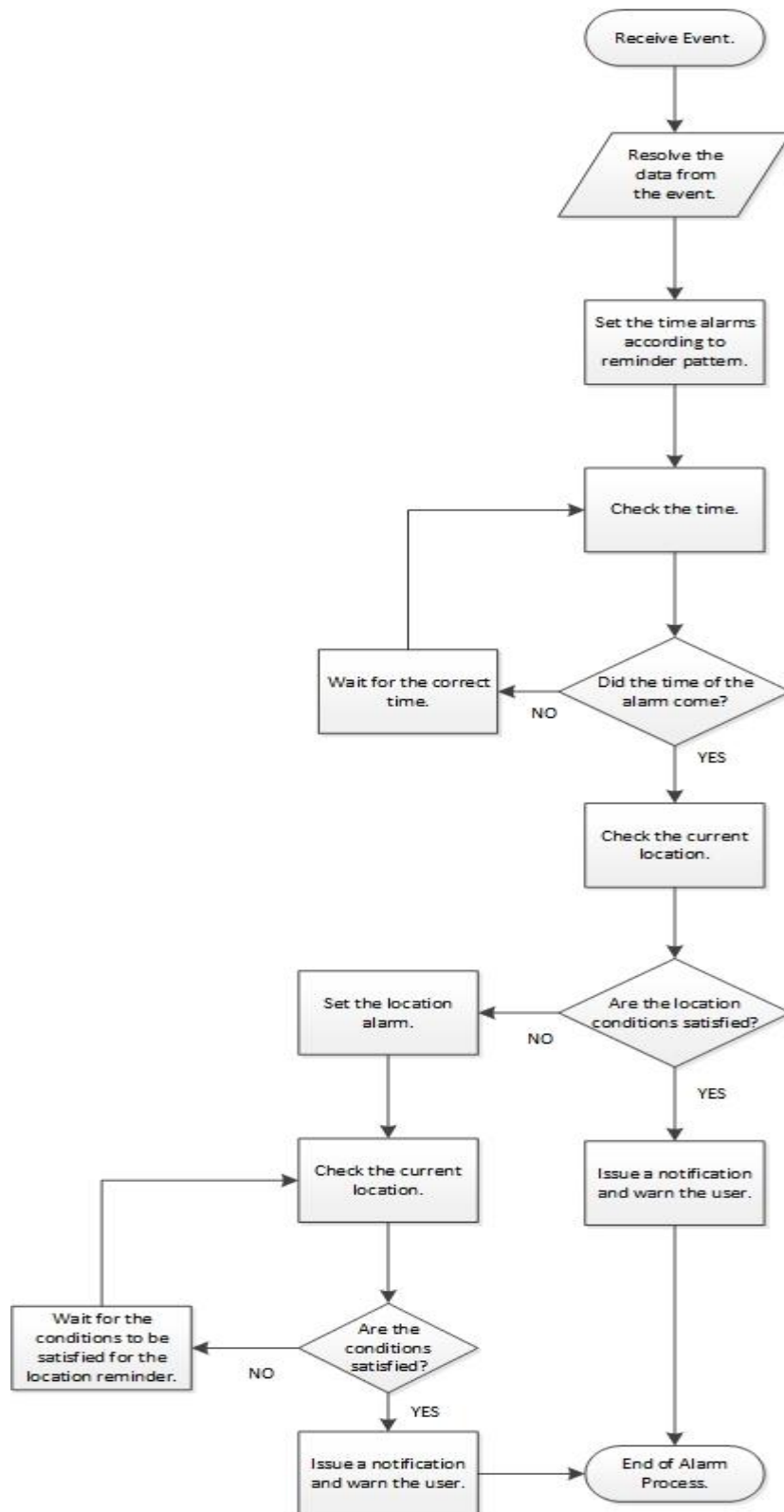


Figure 4.26: The flowchart for event triggering

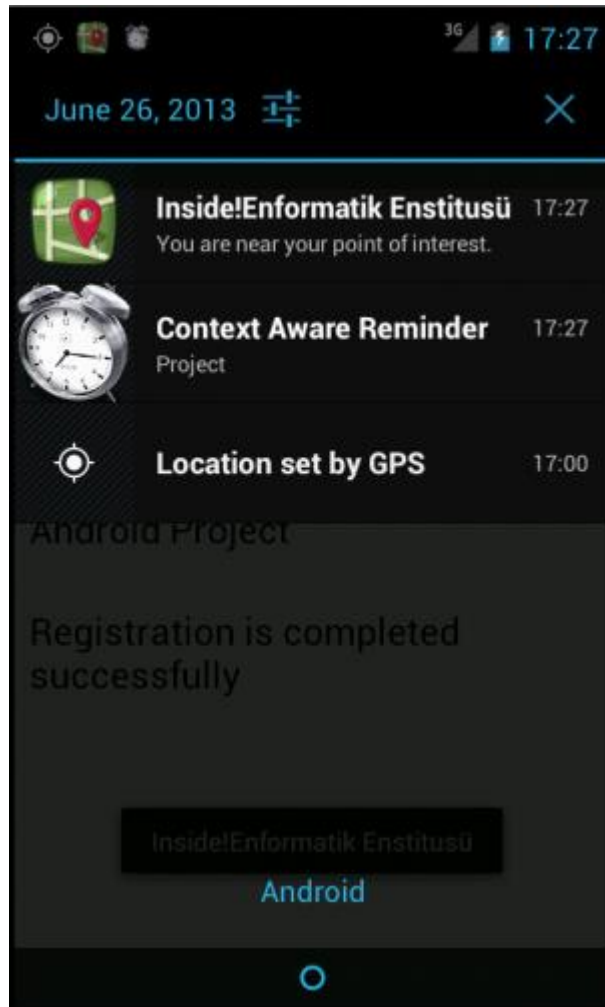


Figure 4.27: Notification Example

The most significant feature of the mobile application is to notify user according to "Reminder Pattern" which is sent by the publisher. Users can also change the reminder pattern for their own interests. The notice of reminders can be a sound, a vibration or a visual according to the request of the user. A notification example can be seen in figure 4.27 above. The visual notification of time alarm and location alert are chosen differently to improve perception.

4.1.5.1 Permissions

The following permission must be included in the AndroidManifest.xml file for the proper functioning of the application.

- `<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>`

Application can register and receive messages from GCM.

- `<uses-permission android:name="com.cara.onder.permission.C2D_MESSAGE"/>`

It builds a custom permission a result only this application can receive its messages.

- `<uses-permission android:name="android.permission.INTERNET"/>`

Allows application to use Internet services

- `<uses-permission android:name="android.permission.GET_ACCOUNTS" />`

GCM needs a Google account information. It is not necessary for the devices which are running a version greater than Android 4.0.3

- `<uses-permission android:name="android.permission.WAKE_LOCK" />`

Allows application to wake up the processor when a message is received from GCM.

- `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>`

Allows applications to get information about networks.

- `<uses-permission android:name="android.permission.READ_PHONE_STATE" />`

Allows application to read the phone state.

- `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />`

Allows application to get the precise location information.

- `<uses-permission android:name="android.permission.VIBRATE" />`

Allows application to use the vibration.

- `<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="8" />`

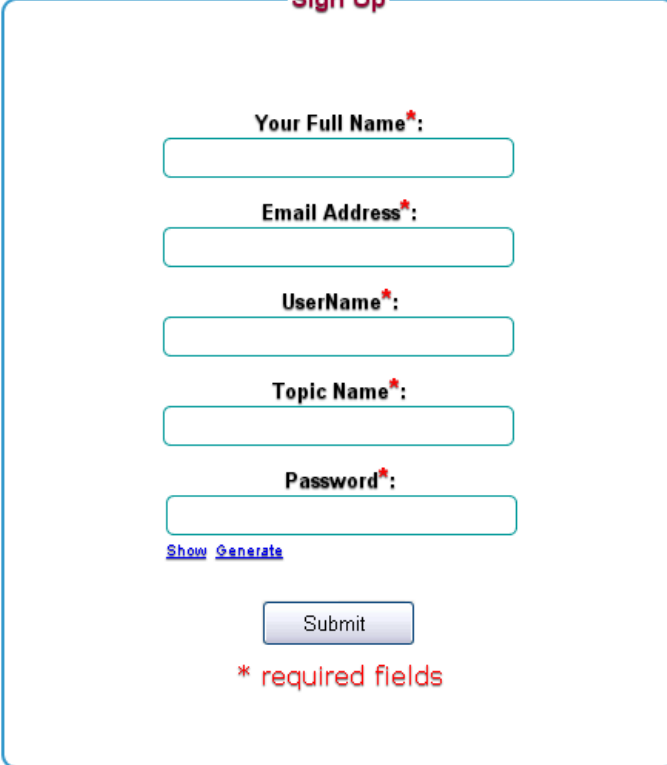
Allows application to use minimum 8 for API level or above. It is required for the proper functioning of GCM features.

4.2 System Security

Proposed model provides an authentication service for Publisher component. The publishers need to be authenticated for accessing message service. For registration and login operations RegistrationForm with GNU Lesser General Public License has been used. It provides operations such as registration, authentication, form validation, and sending emails. Moreover, the passwords are encrypted with MD5 algorithm.

4.2.1.1 Registration

If a user accesses the system for the first time, he/she has to register (sign up) for context aware reminder system. Registration starts by filling the registration form which has the following fields: full name, email address, username, topic name, and password. Then the server sends a verification code to the user's email address. Registration process ends when the user verifies the code by visiting the link with the code. Registration Form is illustrated in Figure 4.28.



The registration form is titled "Sign Up" in red text at the top center. It contains five input fields, each with a label and a red asterisk indicating it is required: "Your Full Name*", "Email Address*", "UserName*", "Topic Name*", and "Password*". Below the password field, there are two small blue links: "Show" and "Generate". At the bottom of the form is a "Submit" button. Below the button, the text "* required fields" is displayed in red.

Figure 4.28: Registration Form

4.2.2 Login

Proposed model implements an authentication service for verifying the identity of the user. The users need to be authenticated with username and password in order to access the message service. The system compares the MD5 hashed passwords stored in the MySQL database with the visitor's one to authenticate the user. If both passwords are the same the visitor is assigned an authorized session to use the system.

Context Aware Reminder System

UserName*:

Password*:

* required fields

[Sign Up](#) [Forgot Password?](#) [Get Application](#)

Figure 4.29: Login Page

In addition, subscribers also need to register first in order to use the mobile application. During the registration process the mobile application sends its registration intent to the GCM. After successful registration GCM sends the registration ID to the android device and the android device sends the registration ID to our Application server while subscribing to a publisher topic. The Application server stores the registration ID in the database and uses it while sending events to the right subscriber. Subscription is an optional procedure and users can subscribe to the desired topic.

4.3 Evaluation and Test Case (User Experiment)

In this section, the test of different components of the context aware reminder system by two types of users, subscriber and publisher, and the evaluation results of these tests are presented. There are predefined and different tasks for subscribers and publishers. These tasks are listed and explained below for each group.

4.3.1 Tasks

4.3.1.1 Subscriber's Tasks (Android Application Side)

1. **Register for the first usage:** The subscribers need to register in order to use the context aware reminder application when they run it for the first time after downloading the application. They need to specify a valid username and e-mail address for the completion of this task.

2. **Subscribe to the topic:** One of the features of the system is to get notifications for a chosen topic. Users should subscribe to a topic so as to receive these notifications with the help of the “Subscribe to New Topics”.
3. **Receive event and edit event:** Another functionality of the application is to receive events created by the publishers. Subscribers could change the settings of these events. These settings are event title, topic, subject, event date, event time, reminder pattern and event location.
4. **Create new event (time and location reminder):** Subscribers have the option of creating new events. They need to enter the “Create New Event Menu” to do this. While creating a new event, they specify the event title, topic, subject, event date, event time, reminder pattern and event location.
5. **Define location:** Subscribers should define location in order to use these locations for the event location while creating new event. They can achieve this task via “Define Location” menu.
6. **Define reminder pattern:** Subscribers should use “Define Reminder Pattern” menu so as to create reminder pattern. They need to specify the name of the pattern and save it. They can use the created reminder patterns while creating a new event.

4.3.1.2 Publisher’s Tasks

1. **Register for the first usage by defining a Topic name:** Publishers need to register in order to use the context aware reminder system. They fulfill this task via entering the website, and using the sign up option. They need to fill in the full name, e-mail address, username, topic name and password fields.
2. **Create Event and Send to Subscribers:** Publishers achieve this task with the help of “Create Event” menu. They need to specify the event title, topic, subject, event date and time, reminder pattern and location. When they fill in these fields and click the submit button, notification is sent to the subscribers based on the specified features generated by the publishers.
3. **Define reminder pattern:** Publishers should use “Define Pattern” menu so as to create a reminder pattern. They need to specify the name of the pattern and click the submit button. They can use the created reminder patterns while creating a new event.
4. **Cancel Event:** Publishers have the option to cancel the created events. They need to use “Sent Events” menu for this purpose. When they enter this menu, they should click on the event name. After that, they will see the chosen event’s details. Moreover, they will have a choice to cancel the event that they have created.
5. **Define Location:** Publishers should define location in order to use these locations for the event location while creating a new event. They can achieve this task via “Define Location” menu. They need to fill in the location name, latitude and longitude fields and click the submit button in order to define the location.

4.3.1.3 Participants

There were 3 participants in the evaluation. They were all M.Sc. students at METU. The first participant is a M.Sc. student at Graduate School of Applied Mathematics. The second participant is a M.Sc. student at Graduate School of Informatics and lastly the third participant is a M.Sc. student at Department of Computer Education and Instructional Technology.

The participants were asked to experience both the publisher and subscriber components of the system. The general opinions of participants about the system are outlined in the next section.

4.3.2 General Opinions of Participants about the System

4.3.2.1 Subscriber's Tasks (User Evaluation)

1. **Register for the first usage:** Generally, registration for the first usage was easy and participants did not have any problems during the registration process.
2. **Subscribe to the topic:** None of the participants encountered any problem during subscribing to a topic. They stated that they chose easily a topic from the dropdown menu and subscribed to the topic easily.
3. **Receive event and edit event:** All of participants received and edited events easily. Moreover, they emphasized that they got the message automatically and modified the event easily.
4. **Create new event (time and location reminder):** In creating new event process, none of participants encountered a problem. They said that after filling in the necessary fields, they created new events easily. Furthermore, one of the participants stated that the design of the menus and application was simple.
5. **Define location:** All participants were pleased with the simplicity of new location definition. Defining location was not a big deal for the participants.
6. **Create reminder pattern:** All of participants understood what a “reminder pattern” is. However, one of the participants did not understand the working principle of the reminder pattern. Furthermore, one of the participants emphasized that “reminder pattern” can cause misunderstanding for the people that do not use mobile applications regularly.

4.3.2.2 Publisher's Tasks (User Evaluation)

1. **Register for the first usage by defining a Topic name:** All participants were pleased with the registration process for the first usage by defining a topic name. They did not experience any problem with the registration process. However, one participant suggested the following three main points about this process:
 - Same topic name can be chosen by different users

- Confirmation email is dropped to the junk folder.
2. **Create an Event and Send It to Subscribers:** Creating an event and sending it to subscribers was not difficult for participants. Furthermore, they pointed out that setting the event date was a helpful function for subscribers since they may forget it after they receive the first message. Although, this process is simple in general, students experienced some problems such as choosing the month and year and “&” symbol by mistake.
 3. **Create reminder pattern:** According to all participants, creating a reminder pattern process was trivial. Participants did not have any problems with this process. However, one of the participants mentioned some programming errors such as
 - Other users’ reminder patterns can be deleted.
 - The same reminder pattern name can be added.
 - Reminder template can be added only after creation of a pattern.
 - After deleting, it redirects to an inconsistent page.
 4. **Cancelling Event:** All of participants are pleased to see that it is possible to cancel an event in this application since generally cancelling an event is not used in such kinds of applications. Furthermore, participants had the following problems during cancelling the events:
 - Finding the “cancelling event” function is not easy. Participants could not find place of it easily.
 - Cumbersome in headings of Archived Events and Cancelled Events pages.
 - After deleting it redirects to an inconsistent page.
 5. **Define Location:** All of participants encountered some problems while defining a new location. Two of the participants said that finding latitude and longitude values is difficult. The other participant said that he did not find delete or update button in the application and adding same location name and coordinates is possible.

4.3.2.3 Participants’ Ideas about the System

Student 1

“Considering the general use and functionality of the application and the website, it could be said that it was easy to use and navigate between menus. Moreover, most of the tasks could be achieved easily in two or three steps. Having a reminder which is based on the location is a useful feature. However, there are some problematic things. One of them is the design of the application side. It seems simple and it needs to be developed. Another thing is the concepts used such as latitude and longitude. It seems difficult to find the coordinate values for these concepts.”

Student 2

“Especially user-friendly interface and functionality of the application is good for the people that are not familiar to use mobile applications. Subscribing an event via “subscribe button” and creating events are appropriate for the functionality of the program. Furthermore,

“define location” function of the application distinguishes this application from other applications. Application is working smoothly and there is no technical problem. Just, I had a problem in the “pattern” name. Researcher should find the more meaningful name instead of pattern. Web interface of the application is also user-friendly and functional for familiar users. However, users that are unfamiliar can encounter some problems. I easily found and used “create event” and “cancel event” buttons. Lastly, I believe that this application should be developed and improved more in the second version.”

Student 3

“I lived some problems during the use of application. I did not understand the notion of pattern. I could not create a pattern and I was bored of understanding its functionality. Generally application is too complex to use it. I could not find what I looked for in the menu. I am not familiar with the using of mobile application so I could not use this application easily. Lastly, I won’t use this application and these types of applications in the future.”

CHAPTER 5

5 CONCLUSION AND FUTURE WORK

The use of smart devices becomes greatly widespread in people's lives with the advances in the Information and Communication Technology. One of the expected features of smart devices is to remind event and activities properly by using context information of user's current environment. However, the development in context aware reminder system area could not adequately follow the improvements in smart devices. Therefore, current reminder systems are not sufficient and they do not meet the user needs. On the other hand, the demand on context aware reminder systems are growing dramatically. In the near future, context aware systems would take more places in our daily lives.

In this thesis, we proposed a model to remind important event and activities at the right time and at the right place in order to improve the quality of life for students in any university campus. In this thesis, the main contribution of the proposed model to the context aware reminder system is that it uses the Publish Subscribe model and it utilizes Reminder Patterns. In this model, the receivers of events are called as subscribers and the senders of events are called as publishers. Users subscribe in one or more topics that attract their attention and receive events about these topics. There are two significant advantages of this model and these advantages are the scalability and loose coupling. In this way, publishers do not have to know all of their subscribers and multiple publishers send events to multiple subscribers.

In addition, our proposed model utilizes Reminder Patterns. In a nutshell, Reminder Patterns can be defined as a collection of reminder templates. One reminder pattern can contain several reminder templates which adjust when and where to fire on the reminder alarms. The users are notified with a proper action when predefined conditions are satisfied. (Context info such as time and location).

In order to prove the applicability of our proposed model, a prototype context aware reminder system has been developed. The prototype system consists of two server side components, two supporting components and one mobile component. The Publisher and the Dispatcher are the server side components which are used to create and send events to the correct subscribers. As a subscriber, an android application is implemented to receive and display the reminder events. The PubSubHubBub service and Google Cloud Messaging service are used as supporting components. The PubSubHubBub protocol has a role in transmitting the events from publisher to dispatcher and Google Cloud Messaging service has a role in distribution of events to subscribers' smart devices.

In order to evaluate the functionality of the reminder system proposed, the prototype system has been used by three users. In the tests, users were asked to fulfill the predefined tasks and

to use the system for two days. After the tests, users were asked to indicate the general opinion about the system.

According to the indicated users' opinions, two of users are pleased with the use of the system and one of participants is not happy with it and he indicated that the system was a little complex.

As a future work, some improvements can still be made on this model. Especially, visual design of the android application can be improved and user interfaces can be designed more intuitively and user-friendly. Furthermore, the complexity of the reminder pattern can be reduced and the use of the reminder pattern can be simplified. Finally, the system evaluation can be performed with more users. In this manner, the results about the system will be more efficient. Consequently, it is believed that the system would be more efficient and subscribers and publishers would use the system more frequently in their daily lives together with the aforementioned enhancements.

REFERENCES

- Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999, January). Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing* (pp. 304-307). Springer Berlin Heidelberg.
- Asai, D., Orszulak, J., Myrick, R., Lee, C., Coughlin, J. F., & de Weck, O. L. (2011, October). Context-aware reminder system to support medication compliance. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on* (pp. 3213-3218). IEEE.
- Chaminda, H. T., Klyuev, V., & Naruse, K. (2012, February). A smart reminder system for complex human activities. In *Advanced Communication Technology (ICACT), 2012 14th International Conference on* (pp. 235-240). IEEE.
- DeVaul, R. W., & Clarkson, B. (2000). The memory glasses: towards a wearable, context aware, situation-appropriate reminder system.
- Dey, A. K., & Abowd, G. D. (2000, January). CybreMinder: A context-aware system for supporting reminders. In *Handheld and Ubiquitous Computing* (pp. 172-186). Springer Berlin Heidelberg.
- Ho, J., & Intille, S. S. (2005, April). Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 909-918). ACM.
- Kim, S. W., Kim, M. C., Park, S. H., Jin, Y. K., & Choi, W. S. (2004, August). Gate reminder: a design case of a smart reminder. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques* (pp. 81-90). ACM.
- Kwon, O., & Choi, S. (2008). Applying associative theory to need awareness for personalized reminder system. *Expert Systems with Applications*, 34(3), 1642-1650.
- Kwon, O., Choi, S., & Park, G. (2005). NAMA: a context-aware multi-agent based web service approach to proactive need identification for personalized reminder systems. *Expert Systems with Applications*, 29(1), 17-32.
- Li, K. A., Sohn, T. Y., Huang, S., & Griswold, W. G. (2008, June). Peopletones: a system for the detection and notification of buddy proximity on mobile phones. In

Proceedings of the 6th international conference on Mobile systems, applications, and services (pp. 160-173). ACM.

Meier, R. (2012) *Professional Android 4 Application Development*. Wiley.

Weiser, M. (1991). The computer for the 21st century. *Scientific american*, 265(3), 94-104.

Creating a registration form using PHP (2012, January 9). Retrieved June 29, 2013, from <http://www.html-form-guide.com/php-form/php-registration-form.html>

SQLiteDatabase (n.d.). Retrieved June 29, 2013, from <http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html>

Google Cloud Messaging for Android (n.d.). Retrieved June 29, 2013, from <http://developer.android.com/google/gcm/index.html>

Pubsubhubbub. (n.d.). Retrieved June 29, 2013, from <https://code.google.com/p/pubsubhubbub/>

Android Tutorial For Beginners : Android SQLite Database Example Tutorial (n.d.). Retrieved June 29, 2013, from <http://vimaltuts.com/android-tutorial-for-beginners/android-sqlite-database-example>

Message /Icons (n.d.). Retrieved June 29, 2013, from <http://www.iconarchive.com/>

Create alarm set on a specified time, using AlarmManager and BroadcastReceiver. (2012, May 23). Retrieved June 29, 2013, from <http://android-er.blogspot.com/2012/05/create-alarm-set-on-specified-time.html>

Tsagklis, I. *Android Proximity Alerts Tutorial* (2011, January 10). Retrieved June 29, 2013, from <http://www.javacodegeeks.com/2011/01/android-proximity-alerts-tutorial.html>

Buikema, B. *Developing Proximity Alerts for Mobile Applications using the Android Platform* (2011, January 10). Retrieved June 29, 2013, from <http://blog.brianbuikema.com/2010/07/part-1-developing-proximity-alerts-for-mobile-applications-using-the-android-platform/>

TEZ FOTOKOPİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü

Sosyal Bilimler Enstitüsü

Uygulamalı Matematik Enstitüsü

Enformatik Enstitüsü

Deniz Bilimleri Enstitüsü

YAZARIN

Soyadı :

Adı :

Bölümü :

TEZİN ADI (İngilizce) :

.....

.....

.....

.....

TEZİN TÜRÜ : Yüksek Lisans Doktora

1. Tezimin tamamı dünya çapında erişime açılsın ve kaynak gösterilmek şartıyla tezimin bir kısmı veya tamamının fotokopisi alınsın.
2. Tezimin tamamı yalnızca Orta Doğu Teknik Üniversitesi kullanıcılarının erişimine açılsın. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.)
3. Tezim bir (1) yıl süreyle erişime kapalı olsun. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.)

Yazarın imzası

Tarih