SAVITRIBAI PHULE PUNE UNIVERSITY

A PROJECT REPORT ON

META-APP: A FRAMEWORK FOR INCOGNITO MODE OF SMARTPHONE APPLICATIONS

SUBMITTED TOWARDS THE
PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING (Computer Engineering)

BY

Tushar Badgu	Exam No: B120054232
Omkar Patil	Exam No: B120054416
Abhidnya Patil	Exam No: B120054413
Snehal Rasakar	Exam No: B120054433

Under The Guidance of

Prof. K.C. Waghmare

DEPARTMENT OF COMPUTER ENGINEERING
PUNE INSTITUTE OF COMPUTER TECHNOLOGY

Sr. No. 27,Pune Satara Road, Dhanakawadi PUNE 411043

PUNE INSTITUTE OF COMPUTER TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the Project Entitled

META-APP: A FRAMEWORK FOR INCOGNITO MODE OF SMARTPHONE APPLICATIONS

Submitted by

Tushar Badgu	Exam No: B120054232
Omkar Patil	Exam No: B120054416
Abhidnya Patil	Exam No: B120054413
Snehal Rasakar	Exam No: B120054433

is a bonafide work carried out by Students under the supervision of Prof. K.C. Waghmare and it is submitted towards the partial fulfillment of the requirement of Bachelor of Engineering (Computer Engineering).

Prof. K.C. Waghmare Internal Guide Dept. of Computer Engg.	Prof. G.P. Potdar H.O.D Dept. of Computer Engg.

Dr.P.T. Kulkarni

Principal

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

Signature of Internal Examiner

Signature of External Examiner

PROJECT APPROVAL SHEET

A Project Title

(MetaApp:A framework for incognito mode of smartphone applications)

Is successfully completed by

Tushar Badgu	Exam No: B120054232
Omkar Patil	Exam No: B120054416
Abhidnya Patil	Exam No: B120054413
Snehal Rasakar	Exam No: B120054433

at

DEPARTMENT OF COMPUTER ENGINEERING

(PUNE INSTITUTE OF COMPUTER TECHNOLOGY)

SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE

ACADEMIC YEAR 2015-2016

Prof. K.C. Waghmare	Prof.Dr.G.P. Potdar
Internal Guide	H.O.D
Dept. of Computer Engg.	Dept. of Computer Engg.

Abstract

Nowadays smartphones applications have become common and are used by millions of people. However poorly written apps pose the rest of exposing user data and breaching privacy. Although user is fine with sharing some of his data for the benefit of the service. In current app scenario, there is no functionality for customized data sharing. Apart from this, the continuous notifications from these apps, can be annoying to the users. A more convenient approach will be to provide notifications on the current user needs. Incognito aims at creating framework which will help publish these intents and regulate the responses of the dealers anonymously. For providing the users with the best offers, the dealers will be ranked based on user reviews. This framework also allows the user to give selective access to certain files which he is comfortable with sharing. Thus, this project aims at developing a symbiotic relationship between the users and dealers by giving user a control over his data sharing on one hand and providing dealers with potential buyers on the other.

Acknowledgments

It gives us great pleasure in presenting the preliminary project report on ***'MetaApp:A framework for incognito mode of smartphone applications'****.*

I would like to take this opportunity to thank my internal guide ***Prof.K.C. Waghmare*** *for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.*

I am also grateful to ***Prof.Dr.G.P. Potdar****, Head of Computer Engineering Department, Pune Institute of Computer Technology for his indispensable support, suggestions.*

In the end our special thanks to ***Computer Department*** *for providing various resources such as laboratory with all needed software platforms, continuous Internet connection, for Our Project.*

Tushar Badgu
Omkar Patil
Abhidnya Patil
Snehal Rasakar
(B.E. Computer Engg.)

INDEX

List of Figures

List of Tables

CHAPTER 1

SYNOPSIS

1.1 PROJECT TITLE

MetaApp:A framework for incognito mode of smartphone applications

1.2 PROJECT OPTION

Internal project

1.3 INTERNAL GUIDE

Prof. K.C. Waghmare

1.4 SPONSORSHIP AND EXTERNAL GUIDE

L3Cube
External Guide: Mr. Abhijit Gadgil

1.5 TECHNICAL KEYWORDS (AS PER ACM KEYWORDS)

1. E.3 DATA ENCRYPTION Public key cryptosystems, standards
2. H.2.4 SYSTEMS Query Processing
3. H.3.1 CONTENT ANALYSIS AND INDEXING Abstracting methods, Linguistic processing
4. H.3.3 INFORMATION SEARCH AND RETRIEVAL Clustering, Query formulation, Retrieval Models
5. H.3.5 ONLINE INFORMATION SERVICES Commercial services, Data sharing
6. I.2.1 APPLICATION AND EXPERT SYSTEMS Natural language interfaces
7. I.2.7 NATURAL LANGUAGE PROCESSING Language parsing and understanding, text analysis

1.6 PROBLEM STATEMENT

MetaApp which acts as a supplant to all the redundant applications and is based upon intent publishing to the providers of the services using a cloud server. A solution to provide fine control to the user over the data that is shared with the service providers and implement a Pull-based approach.

1.7 ABSTRACT

Nowadays smartphones applications have become common and are used by millions of people. However poorly written apps pose the rest of exposing user data and breaching privacy. Although user is fine with sharing some of his data for the benefit of the service. In current app scenario, there is no functionality for customized data sharing. Apart from this, the continuous notifications from these apps, can be annoying to the users. A more convenient approach will be to provide notifications on the current user needs. Incognito aims at creating framework which will help publish these intents and regulate the responses of the dealers anonymously. For providing the users with the best offers, the dealers will be ranked based on user reviews. This framework also allows the user to give selective access to certain files which he is comfortable with sharing. Thus, this project aims at developing a symbiotic relationship between the users and dealers by giving user a control over his data sharing on one hand and providing dealers with potential buyers on the other.

1.8 GOALS AND OBJECTIVES

- To try to move away from App only Approach.
- Expose User Authorized APIs to publishers.
- Increase User privacy by creating a user-defined namespace without storing the data anywhere apart from the users own storage.
- Creating a secure tunnel between the Producer and Consumer with multiple proxies.

1.9 RELEVANT MATHEMATICS ASSOCIATED WITH THE PROJECT

System Description:

- Let ' S ' be the solution set of the given problem statement such that, S = s, e, X, Y, f, DD, NDD, Sc, Fc —
  such that :

- s = Start State
  = User's Request in the form of a Natural Language Sentence

- e = End State
  = Successful response for user's request in the form of desired content

- X = Set of inputs
  = X1, X2
  X1 = Set of users intent.
  = x11,x12 ....., x1n
  X2 = Events triggered by users.
  = x21, x22....., x2n

- Y = Set of outputs
  = Y1, Y2
  Y1 = set of notifications
  = Y11,Y12.....,Y1n
  Y2 = actions in response to an event
  = Y21,Y22.....Y2n

- f = Set of functions
  = f1, f2, f3, f4, f5,f6
  where,

f1 = Function to develop android interfaces to interact with user.

f1(X2)= Y2

I/p : X2

O/p : Y2

f2 = Function to develop a cloud server to process the request

f2(X1)=Y1

I/p : X1

O/p : Y1

f3 = Function to generate tokens

f3(X1i) = Ti.

I/p : X1i

O/p : Ti

Where Ti is Token generated for the particular intent.

f4 = Function to rank the notification.

f4(Y1i)= Ri

I/p : Y1i

O/p : Ri

Where Ri is the rank for the notification

f5 = function to use natural language

processing to extract the keywords from the user request

f5(X1) = Ki

I/p : X1

O/p : Ki

Where Ki is the set of keywords

f6 = function to use android security to enhance the privacy to suit the cus-

tomer needs

I/p : Requests of the service provider for some user private data in order to benefit its own business.

O/p : Fetching the data from the provider namespace authorized by the user

and sending it to service provider as if it is all that is available at the user end.

- DD = Deterministic Data
  = Set of API's exposed on the cloud server

- NDD = Non-Deterministic Data
  = Path created by tunnel , Token Management

- Sc = Success Case
  = Intents received for the requests made by user

- Fc = Failure Case
  = Anything apart from the user approved data is sent to the service providers gets stored on the server

1.10 NAMES OF CONFERENCES / JOURNALS WHERE PAPERS CAN BE PUBLISHED

1. 2016 Second International Conference on Mobile and Secure Services (MobiSecServ)
   Dates:- 26 Feb - 27 Feb 2016

2. 2015 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL)
   Dates:- 19 Nov - 20 Nov 2015

3. 2016 Second National conference on Advancements in Computer and Information Technology(NCACIT)
   Dates:- 5 Feb - 6 Feb 2016

1.11 REVIEW OF CONFERENCE/JOURNAL PAPERS SUPPORTING PROJECT IDEA

1. Enhancing Personal Information Security on Android with a New Synchronization Scheme

   Authors:- Changhao Ai ,Jie Liu

   Year:- 2011

   Reference No:- 978-1-4244-6252-0

   Mobile phones which use Android or other smartphone operating systems, equipped with richer applications than traditional ones, enable users to leave more things to phones to deal with. As a result, personal information stored on mobile phones becomes more and more. It is well known that backup is an important means to keep the information safe.

   Issues:- Cannot keep backup of personal information on the smartphone other than the google provided services such as contacts and sms.

   Solution:- Creating a cloud server for storing all the personal information of the user such as contacts, sms, emails, game scores, application settings in a application predefined format. Whenever a user tries to sync the data, the user will be authenticated based on the google account and then the data is stored on the cloud server.

   Conclusion:- Driven by an analysis of Android security mechanisms and content provider,this paper presents a new Android-based application settings integration and management scheme. It enhances the security of personal information on Android by syncing a user's personal information to the Internet and syncing back if necessary to prevent the lost of personal information.

2. Enhancing Security of Android and IOS by Implementing Need-Based Security (NBS)

Authors:- Muneer Ahmad Dar, Javed Parvez

Year:- 2014

Reference No:- 978-1-4799-4190-2

The most popular Smartphone platforms i.e. Android and iOS are equipped with the built-in security features to safeguard their end users. Android, being an Open Source Mobile Operating System, has some security vulnerabilities. Such limitations are also present in iOS which is a proprietary platform with some open source components. This paper proposes the implementation of a reverse engineering process which restricts an apps permissions and provides a need based mechanism to access resources. The repackaged app with need based security will run on all devices that were supported by the original application.

Issues:- A naive user does not understand the permission management of the various apps while installation. Also android operating system does not provide the user with a facility to grant partial permissions while revoking certain ambiguous permissions Such apps may take advantage of the personal information stored on the phone to its own benefits.

Solution:- The apk file of the app to be installed is decompiled to generate the source package and files. Then the android manifest file is examined to detect some unwanted permissions been granted. All such permissions are removed and if the app requires to access a particular resource, then the permission is obtained at runtime. The packages are recompiled to generate the safe version apk file of that app.

Conclusion:- Our research came to the conclusion that the novice users are not aware of the permissions they have granted at the time of installation particularly of the Android Users. We removed the dangerous permissions from the app based on their functionality and then our research provided the second level of security to the users, without modifying the underlying struc-

ture of Android operating system. All the research undertaken in this paper on permissions removal and run time permission approval was implemented manually. Therefore future research recommendations include automating the process.

3. Vulnerabilities of Android Data Sharing and Malicious Application to Leaking Private Information

   Author:- Taenam Cho, Seung-Hyun Seo

   Year:- 2013

   Reference No:- 978-1-4673-5990-0

   Smartphones are not just phones but also portable computers, providing diverse services needed in life including calls, texts, emails, GPS, camera, Wi-Fi and Bluetooth apps. These apps keep and manage diverse intrinsic data as well as sensitive private information such as address books. Smartphones enable swift and easy data exchange via 3G, 4G and Wi-Fi. Thus, personal information stored on Smartphones is prone to leakage. Particularly, Content Providers provided by Android to share data between apps are susceptible to illegitimate leakage of data.

   Issues:- Android manages each app and its data separately to protect data. For apps requiring communication between them such as update apps, data need be exchanged. For the purpose of data sharing among different apps, Android furnishes Content Providers. The Android security architecture is based on the Sandbox, which isolates an apps codes and data from those of other apps, and the permission, which controls apps accessibility to system functions. So a attacker app can use content resolvers to query the database of server apps running on the phone and obtain the results using content providers. No solution has been provided by the OS to overcome such security loopholes.

   Conclusion:- On Android Smartphones, lots of private information of users is stored in databases. Android relies on the Sandbox to protect codes and data of an app from other apps, whilst it offers the Content Providers to share

databases as necessary. Client apps intending to share databases of other apps need to present the same URIs as the database identifiers and database information as well as permissions required by server apps. Attacker trying to attempt any unauthorized and illegitimate access can use reverse engineering to extract all of the necessary information. Proguard does not guarantee prevention of information extraction. In addition, assigning permissions to control accesses is not mandatory, which leads to security loopholes. In other words, attackers can easily access private information on Android Smartphones with no particularly complicated analysis or alteration or forgery of server apps. The present study analyzes such vulnerabilities and implements a sample malicious app to demonstrate that unauthorized and illegitimate database access/-modification/deletion/leakage can occur.

4. LeakMiner: Detect information leakage on Android with static taint analysis
   Author:- ZheMin Yang, Min Yang
   Year:- 2012
   Reference No:- 978-0-7695-4863-0
   State-of-the-art approaches for detecting Android information leakage apply dynamic analysis on user site, thus they introduce large runtime overhead to the Android apps. The paper proposes a new approach called LeakMiner, which detects leakage of sensitive information on Android with static taint analysis. Unlike dynamic approaches, LeakMiner analyzes Android apps on market site. Thus, it does not introduce runtime overhead to normal execution of target apps. Besides, LeakMiner can detect information leakage before apps are distributed to users, so malicious apps can be removed from market before users download them.
   Issues:- Some of the apps available at the market store might be malicious in the sense that they may be sending user personal information to some unauthorized end-points in the background without the knowledge of the smartphone user. Such information include unique device id, location, phone number, contact book, sms messages, calender.

Solution:- An android application does not have a single entry or exit point as compared to the main function of a C or C++ program. Thus in such case we create a Call graph between the root nodes and each of the entry nodes. Pointer information is also attached to each method denoting the possibility of the method accessing data viable to be leaked. LeakMiner figures out data dependencies between source and sink points in a similar way as Thin Slicing and then checks to see if the personal data is traveling from so source to sink points.

5. MetaExtract: An NLP System to Automatically Assign Metadata
   Author:- Ozgur Yilmazel, Christina M. Finneran,
   Elizabeth D. Liddy
   Year:- 2004
   Reference No:- 1-58113-832-6
   We have developed MetaExtract, a system to automatically assign Dublin Core + GEM Meta data using extraction techniques from our natural language processing research. MetaExtract is comprised of three distinct processes: eQuery and HTML-based Extraction modules and a Keyword Generator module. We conducted a Web-based survey to have users evaluate each metadata elements quality. Only two of the elements, Title and Keyword, were shown to be significantly different, with the manual quality slightly higher. The remaining elements for which we had enough data to test were shown not to be significantly different; they are: Description, Grade, Duration, Essential Resources, Pedagogy-Teaching Method, and Pedagogy-Group.
   Issues:- Extraction of a set of keywords representing the same semantic meaning as of the entire document was a manual process with a large amount of time wastage. Thus the need was to construct a automatic keyword extraction system.
   Conclusion:- Given the amount of effort manual metadata assignment takes and the fact that most automatically assigned metadata elements are not sta-

tistically different in quality,we believe that MetaExtract offers tremendous promise in solving the metadata generation bottleneck.

6. Automatic Keyword Extraction for the Meeting Corpus using Supervised approach
   and Bigram Expansion
   Author:- Fei Liu, Feifan Liu, Yang Liu
   Year:- 2008
   Reference No:- 978-1-4244-3472-5
   In this paper, we tackle the problem of automatic keyword extraction in the meeting domain, a genre significantly different from written text. For the supervised framework, we proposed a rich set of features beyond the typical TFIDF measures, such as sentence salience weight, lexical features, summary sentences, and speaker information. We also evaluate different candidate sampling approaches for better model training and testing. In addition, we introduced a bigram expansion module which aims at extracting entity bigrams using Web resources.
   Issues:- Keywords can provide important information about the content of the document.Web text or normal text have a particular structure as in paragraphs and other grammatical stuctures within the sentence. But meeting corpus is different in the ways that it has more than one person speaking, the discussion is not well organized, the speech is spontaneous and contains disfluencies and ill-formed sentences.
   Solution:- The solution proposed in this paper uses supervised approach for unigram keyword extraction from the corpus along with Bigram expansion. In the supervised approach, a maximum entropy (MaxEnt) classifier is used to determine whether a unigram word is a keyword (binary classification). Each candidate word is represented by a variety of features such as TFIDF, Position features, Stopword features, Sentence features, Lexical features, Summary features and speaker features. One way for model training and testing

in this supervised approach is to simply use all the words as instances. In this paper, we propose to use a resampling technique to select a subset of the candidate words. As the number of bigram keywords were more, thus we proposed a method to extract bigram keywords. First we compute the TFIDF scores for all the bigrams in a document. Then similar to the POS constrain used for word resampling, we predefine six POS patterns to filter bigrams, i.e., nn+nn, nn+nns, jj+nn, jj+nns, nnp+nn, and nnp+nnp (tags used in Penn Treebank tagset). The top N bigrams with the highest TFIDF scores as well as satisfying the POS requirement are selected.

Conclusion:- In this paper, we investigated the problem of automatic keyword extraction in the meeting domain. We adopt a supervised framework and leverage features extracted from meeting specific characteristic such as decision making sentences and system generated summaries. Feature selection and different candidate word resampling techniques prove to be helpful in the supervised method. In addition, we introduced a bigram expansion module which leverages both Web resources and confidence scores from the classifier. Our experimental results on both the human transcripts and ASR output demonstrate that the supervised approach and the bigram module improve keyword extraction performance.

7. Semantic Keyword Selection for Automatic Video Annotation

   Author:- Ali Shariq Imran, Laksmita Rahadianti

   Year:- 2013

   Reference No:- 978-1-4799-3211-5

   Choosing descriptive keywords to best describe digital media content is crucial for many applications, especially those involving content-based indexing or retrieval. Traditionally such keywords are selected manually, which is labor intensive, restrictive to a limited set of words and inherently subjective to the annotator. Therefore, in this paper, we propose an automatic and objective keyword selection method for annotating video. We specifically used lecture videos and surrogate documents, e.g. transcripts, to extract potential candidate

keywords.

Issues:- Content-based media retrieval is one of the most important problem to be solved. For video content there is no way to extract keywords that represent the video.

Solution:- We used cross document annotation principle to semantically select top ranked keywords for video annotation. These keywords are selected automatically using visual similarity and word sense disambiguation. Figure 1 shows a framework. Input to this framework is a lecture video with accompanying text material for cross document annotation. The additional text sources may include audio transcript, title of the video and video categories. Since we used lecture video database, the video categories are the different educational subjects under which a particular video is listed. This is feasible due to the fact that in todays e-learning.

Conclusion:- We develop a keyword selection method to automatically annotate lecture videos. The objective method is designed for digital media with surrogate documents. We specifically used lecture videos with accompanying transcripts to test the validity of proposed method. We used the title of video and the categories as ground truth for selecting potential keywords. The potential keywords were extracted from lecture transcript. We then score each potential keyword using a similarity measure and disambiguation criteria between different senses of words. We proposed LVD-F method to do this objectively. LVD-F employ visualness to describe the semantic value of the words and uses Lesk algorithm disambiguation for WSD. This method is a hybrid combination of visualness with TF-IDF.

8. A large scale publish-subscribe platform for information delivery to mobile phones

   Author:- Thejovardhana S. Kote , S. R. Jeyashankher

   Year:- 2006

   Reference No:- 0-7695-2616-0

The MyToday platform is built around the publish-subscribe paradigm where publishers publish or blog to channels and subscribers subscribe or opt-in to channels of their choice. This is very similar to RSS feeds where a publisher creates an RSS feed which contains the messages published and subscribers subscribe to a feed of information. We generalize this concept of feeds to include various modes of information delivery besides RSS. Our platform has several salient features. A schematic of the internals of the platform. In keeping with the publish-subscribe paradigm it comprises mainly of two pieces, the publishing and the subscriptions modules. The former provides the myriad interfaces (web, e-mail, SMS etc.) to publishers for information dissemination while the latter keeps track of user preferences and delivers alerts to subscribers on the medium of their choice. The two modules are tied together by a powerful feed management system.

9. Learning Query Intent from Regularized Click Graphs Author:- Xiao Li, Ye-Yi Wang

   Year:- 2008

   Reference No:- 978-1-60558-164-4

   This work presents the use of click graphs in improving query intent classifiers, which are critical if vertical search and general-purpose search services are to be offered in a unified user interface. Previous works on query classification have primarily focused on improving feature representation of queries, e.g., by augmenting queries with search engine results. In this work, we investigate a completely orthogonal approach instead of enriching feature representation, we aim at drastically increasing the amounts of training data by semi- supervised learning with click graphs. Specifically, we infer class memberships of unlabeled queries from those of labeled ones according to their proximities in a click graph. Moreover, we regularize the learning with click graphs by content-based classification to avoid propagating erroneous labels. We demonstrate the effectiveness of our algorithms in two different applications, product intent and job intent classification. In both cases, we expand the training data

with automatically labeled queries by over two orders of magnitude, leading to significant improvements in classification performance. An additional finding is that with a large amount of training data obtained in this fashion, classifiers using only query words/phrases as features can work remarkably well.

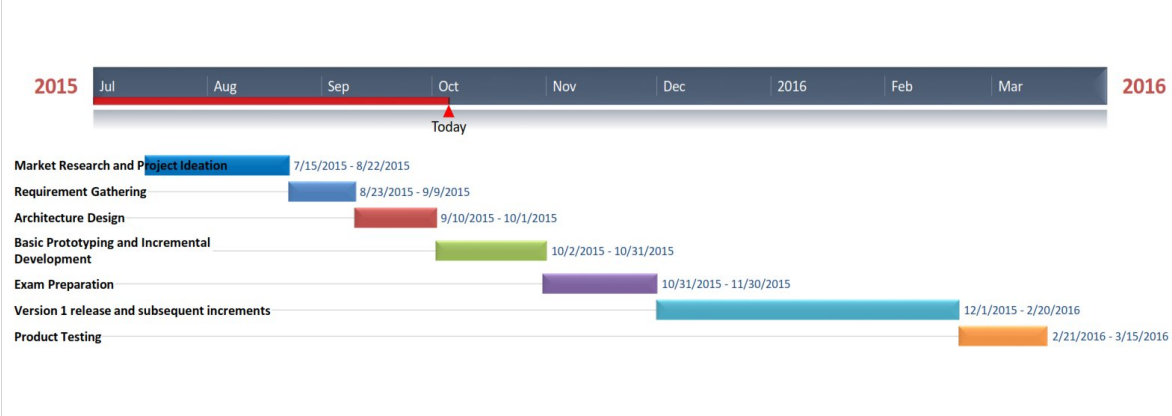
10. Time-based OTP Authentication via Secure Tunnel (TOAST): A Mobile TOTP Scheme Using TLS Seed Exchange and Encrypted Offline Keystore
    Author : Mariano Luis T. Uymatiao, William Emmanuel S. Yu
    Year : 2014
    Reference No. : 978-1-4799-4808-6 /14
    The main objective of this research is to build upon existing cryptographic standards and web protocols to design an alternative multi-factor authentication cryptosystem for the web. It involves seed exchange to a software-based token through a login-protected Transport Layer Security (TLS/SSL) tunnel, encrypted local storage through a password- protected keystore.
    Issues:- In order to keep data on the web as safe as possible, many clients and

    servers implement cryptographic techniques to encrypt sensitive data, as well as verify entities at the other end of the connection. Online users continue to use weak, guessable passwords, and password-cracking hardware continues to improve at a blistering pace. It is becoming increasingly clear that passwords are insufficient as a means for protecting online accounts. Solution:- Numerous aspects of online security today were considered throughout this research.
    A. Usernames and Passwords
    B. Multi-factor Authentication
    C. One-time Password (OTP)
    D. HOTP and TOTP
    a. SMS OTP
    b. Google authenticator Conclusion:- TOAST was built on the need to further

    harden existing two-factor authentication schemes by making sure that secret values remain in an encrypted form whenever possible. Approach to token-

based authentication will help further the discourse on the important topic of authentication, and introduce new ideas for removing any remaining points of vulnerability.

1.12 PLAN OF PROJECT EXECUTION



CHAPTER 2

TECHNICAL KEYWORDS

2.1 AREA OF PROJECT

The major area of the project will be developing a framework for e-commerce which will be focussing on preserving the privacy of users data as well as providing the sellers a platform to market their products and services.It will be primarily deployed on an Android App to replace all such e-commerce mobile applications.

2.2 TECHNICAL KEYWORDS

1. E.3 DATA ENCRYPTION Public key cryptosystems, standards
2. H.2.4 SYSTEMS Query Processing
3. H.3.1 CONTENT ANALYSIS AND INDEXING Abstracting methods, Linguistic processing
4. H.3.3 INFORMATION SEARCH AND RETRIEVAL Clustering, Query formulation, Retrieval Models
5. H.3.5 ONLINE INFORMATION SERVICES Commercial services, Data sharing
6. I.2.1 APPLICATION AND EXPERT SYSTEMS Natural language interfaces
7. I.2.7 NATURAL LANGUAGE PROCESSING Language parsing and understanding, text analysis

CHAPTER 3

INTRODUCTION

3.1 PROJECT IDEA

Nowadays, we can find an android application developed by every other service provider to enable the users easy access to their products simply by a click of a button on their phones. However, this necessitates the users to download the applications provided by each of these providers. Thus the user spends a lot of time compare the prices and offers by looking into each of these applications and find the best available deal. Also many of these applications are poorly written and thus exploit unnecessary personal information of the user.[1]

For example, Amazon , Flipkart , Myntra are all leading providers in providing products ranging from home appliances to electronic items , from clothing to footwear and even super market products . Uber, Ola, Meru are all the providers of taxi services to the user . Similarly, make my trip, yatra.com, travel advisor are useful when planning a trip and includes everything from transport bookings and hotel reservations. If a user thinks of buying shoes , he tends to check the prices offered by each of the providers to get the most cheapest deal for himself. Keeping this in mind, we design a tool which just asks the user to specify his search .The service providers will be provided with API's which can be used to publish their services to the users. We will be developing an efficient ranking strategy to order the search. And the best possible deal will be provided to the user without the need of installing so many apps and without even compromising the security of his personal information. A security layer will be developed over normal android security to give the user a complete control of sharing of his personal data.

3.2 MOTIVATION OF THE PROJECT

Currently most people in the world are using various kinds of apps in their day to day life for each and every task like for booking a cab, online shopping apps like Flipkart, Amazon, etc. However, slowly as the number of these apps has increased it is becoming more and more time consuming as well as data consuming to actually use this various apps at a time on various operating system.

Also everyone is moving app only so there is trend in todays market for having app for their product. Development and maintenance of this apps is also a major problem only few people are concern about it. Some apps required basic information of users such as name, email, phone number, etc. which is not required actually. Because of this app going trend everyone is developing app but no one is concern about fine UI, security and privacy so this poorly written apps harms the operating system. That is where comes our product. It is basically considers the need of security and privacy of user while using app. If we are going to ask for some product and server will give us information about various products on various apps with the help of APIs so it is kind of pull based approach. This is a totally new concept. A person in the future may only just check our product to check which provider giving him fast and best services. Some features might achieve this implicitly and some might be explicitly built for the same.

3.3 LITERATURE SURVEY

We found that the average Android app sends potentially sensitive data to third-party domains, and the average iOS app connects to 2.6 third-party domains.Android apps are more likely than iOS apps to share with a third party personally identifying information such as name (73% of Android apps vs. 16% of iOS apps) and email address (73% vs. 16%). For location data, including geo-coordinates, more iOS apps (47%) than Android apps (33%) share that data with a third party. In terms of potentially sensitive behavioral data, we found that 3 out of the 30 Medical and Health & Fitness category apps in the sample share medically-related search terms and user inputs with a third party. Finally, the third-party domains that receive sensitive data from the most apps are Google.com (36% of apps), Googleapis.com (18%), Apple.com (17%), and Facebook.com (14%). 93% of Android apps tested connected to a mysterious domain, safemovedm.com, likely due to a background process of the Android phone. Our results show that many mobile apps share potentially sensitive user data with third parties, and that they do not need visible permission requests to access the data. Future mobile operating systems and app stores should consider designs that more prominently describe to users potentially sensitive user data sharing

by apps.

For implementation of algorithm we searched for several IEEE papers on token generation algorithm, Numerous aspects of online security today were considered throughout this research.

A.Usernames and Passwords

B. Multi-factor authentication

C. One-time Password (OTP)

The main objective of this research is to build upon existing cryptographic standards and web protocols to design an alternative multi-factor authentication cryptosystem for the web. It involves seed exchange to a software-based token through a login-protected Transport Layer Security (TLS/SSL) tunnel, encrypted local storage through a password-protected keystore .

CHAPTER 4

PROBLEM DEFINITION AND SCOPE

4.1 PROBLEM STATEMENT

Meta-App which acts as a supplant to all the redundant applications and is based upon intent publishing to the providers of the services using a cloud server. A solution to provide fine control to the user over the data that is shared with the service providers and implement a Pull-based approach.

4.1.1 Goals and objectives

Goal and Objectives:

- Meta-App is a Pull-based approach to receive notifications in contrary to the current Push-based approach. We propose a generalized end-point like an app which would be used by the user to publish his intent. This intent would be then processed and forwarded to providers who provide that service.

- Instead of sending user info a special token would be generated by the meta-app server. This token would act as a key to establish a secure tunnel between the two entities. All the communication will take place using this tunnel without the interference of the MetaApp server.

  Main objectives are as follows :

  1. To try to move away from App only Approach.
  2. To expose User Authorized APIs to publishers.
  3. To increase User privacy.
  4. To create a secure tunnel.

4.1.2 Statement of scope

The main scope of our project is to design an android app that will expose few APIs(e.g. location, Interest, Notification). After designing the app, it will be deployed on the cloud server. An android app is acting as a end point communication in this process. Due to use of server provider will not be able to see users information. Users also can see log of data which to be sent to provider.

4.2 MAJOR CONSTRAINTS

- As our tool will be handling a lot of data of a number of users the Database at the backend must be efficient in retrieving the data. The fetch time should be less so that there are not any delays.
- The database server should not crash in handling huge amounts of data and when a lot of users access it at the same time.
- As we promised to provide complete privacy of users of our system,there should be efficient security mechanism which cannot be hacked.[2]
- The proxy server should be smart enough to close the session and stop the publishing of intents after the user request is serviced.

4.3 METHODOLOGIES OF PROBLEM SOLVING AND EFFICIENCY ISSUES

- Automatic Keyword extraction algorithm:-
  The algorithm can be divided into three main steps:

  1. Text pre processing :

     During the text pre processing phase, the article's content is divided into tokens an non-significant characters are removed. Named entities are recognized by the elementary method: the first upper-case letter with corpus-based statistic is good enough for this recognition. The tokens are divided by their POS tag and generally only nouns and adjectives can be declared as potential keywords. The next idea is to choose only common words instead of unique ones. Keywords are often used for clustering and a keyword is useless when it is assigned to only a few articles. This is the reason why we remove tokens with low frequency and set up rules for general nouns. There is no such rule for named entities. The remaining tokens and named entities are declared as keywords candidates. We calculate the TF*IDF score only for these keyword candidates.

2. Keyword extraction :

   TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

   TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

   IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such a ”is”, ”of”, and ”that”, may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing following:

   IDF(t) = log(Total number of documents / Number of documents with term t in it).

   The keyword extraction phase contains only the method for removing useless candidates whose TF*IDF score is lower than 1/5 of a maximal value. This boundary can be changed by the number of requested automatic keywords.

3. Keyphrase extraction :

   The keyphrase extraction part can be described by these steps:

   - NLP method interesting n-grams are chosen. The choice is based on their POS tag patterns and the corpus frequency is counted only for these n-grams. These n-grams can be marked as keyphrase candidates.
   - A score of importance is counted for each keyphrase candidate. This score contains the n-gram corpus frequency and TF*IDF score for each word. The score is used for document keyphrase selection.
   - Derivation keyphrase candidates are merged with named entities or

individual keywords if their co-occurrence is significant for this document.

Used POS patterns:
A) POS patterns for 3-grams: (N or named entity), (V or A or stop word), (N or named entity), 3x (named entity)
example: Tim Berners Lee
B) POS patterns for 2-grams
A, (N or named entity), 2x (named entity)
example: Bill Gates

4.4 OUTCOME

A new framework for Online market application with following functionalities:
1) User Privacy
2) Intent Publishing
3) Pull-based approach implementation
4) Single platform for all e-commerce applications

4.5 APPLICATIONS

Amazon , Flipkart , Myntra are all leading providers in providing products ranging from home appliances to electronic items , from clothing to footwear and even super market products . Uber, Ola, Meru are all the providers of taxi services to the user.
Similarly, make my trip, yatra.com, travel advisor are useful when planning a trip and includes everything from transport bookings and hotel reservations. If a user thinks of buying shoes , he tends to check the prices offered by each of the providers

to get the most cheapest deal for himself. The application of the project includes providing services to all these sellers

4.6 HARDWARE RESOURCES REQUIRED

The Hardware resources are specified in table below:

Sr. No.	Parameter	Minimum Requirement	Justification
1	CPU Speed	2 GHz	Computation and Storage
2	RAM	3 GB	Faster Computation

Table 4.1: Hardware Requirements

4.7 SOFTWARE RESOURCES REQUIRED

Platform :

1. Operating System: Ubuntu 14.10
2. IDE: Android Studio, PyCharm
3. Programming Language : Python, Java

CHAPTER 5

PROJECT PLAN

5.1 PROJECT ESTIMATES

5.1.1 Reconciled Estimates

- Statement of Scope

  i) Defination of atleast 3-4 apis to be exposed(e.g. location,Interest,Notification)

  ii) A token-based User Data access mechanism

  iii) Implementation of a Cloud and proxy Server

  iv) An Android App acting as an end-point

  v) Provider based namespace

  vi) Log of what is sent to the provider viewable by the user only

- Problem Functions

  i) Develop the basic communication model

  ii) Publish intents and provide notifications for responses

  iii) Api Definition

  iv) Implement provider namespace

  v) Ranking mechanism

- LOC or FP value for each function

  Function Point values for each above descriped function:

  i) b+c = 11

  ii) a+b+c+e = 17

  iii) d+e = 8

  iv) e = 3

  v) b+c+e = 14

  where,

  a=external inputs (e.g. file names)

  b=external outputs (e.g. reports, messages)

  c=queries (interactive inputs needing a response)

  d=external files or interfaces (files shared with other software system)

e=internal files (invisible outside the system)

5.1.1.1 Cost Estimate

The monetary cost of the project is negligible as all the softwares used are open-source.

5.1.1.2 Time Estimates

The project is supposed to be completed till February. So an estimated time of 5-6 months is required for getting all the modules working.

5.1.2 Project Resources

- Hardware resources required :
  1. Server to host main applications
  2. Server to host proxy
- Software resources required
  1. Python 2.7.9
  2. Flask 0.10.1
  3. Sqlite
  4. Android studio 1.4.0
  5. Google cloud messaging
  6. Sqlalchemy
  7. Ubuntu 14.04
  8. PyCharm IDE

5.2 RISK MANAGEMENT W.R.T. NP HARD ANALYSIS

This section discusses Project risks and the approach to managing them.

5.2.1 Risk Identification

Risk identification is a systematic attempt to specify threats to the project plan.

1) Generic Risks

1. Misunderstanding of requirement
2. Lack of top management commitment and support
3. Failure to manage end user expectation
4. Changes to requirements

2) Product-specific risks

1. Scope of the project
2. Acceptance of product by the huge e-commerce world
3. Appealing to the users of the products
4. Availability of hardware and software resources to the developer
5. Adoption to the destructive methodology to solve the problem as of the current trend
6. All the developers meeting the FP requirements to satisfy all the fuctional requirements within deadline

5.2.2 Risk Analysis

The risks for the Project can be analyzed within the constraints of time and quality

Risk Identified	Risk Category	Probability	Impact(1-10)
1.1	Customer Characteristics	20 %	6
1.2	Business Impact	10 %	3
1.3	Customer Characteristics	10 %	8
1.4	Customer Characteristics	20 %	8
2.1	Product Size	30 %	9
2.2	Business Impact	30 %	8
2.3	Customer Characteristics	10 %	6
2.4	Development Environment	10 %	8
2.5	Technology to be built	20 %	7
2.6	Staff size and experience	20 %	5

Table 5.1: Risk Analysis

5.2.3 Overview of Risk Mitigation, Monitoring, Management

1. Maintain a global perspective
2. View software risks within the context of a system
3. Take a forward-looking view
4. Think about risks that may arise in the future establish contingency plans
5. Encourage open communication
6. Encourage all stakeholders and users to point out risks at any time
7. Integrate risk management
8. Integrate the consideration of risk into the software process
9. Emphasize a continuous process of risk management

10. Modify identified risks as more becomes known and add new risks as better insight is achieved

11. Develop a shared product vision

12. A shared vision by all stakeholders facilitates better risk identification and assessment

13. Encourage teamwork when managing risk

14. Pool the skills and experience of all stakeholders when conducting risk management analysis

5.3 PROJECT SCHEDULE

5.3.1 Project task set

Major Tasks in the Project stages are:

- Task 1: Project Topic Selection

- Task 2: Literature Survey and Study of Papers
  2.1 : Study of ACM Papers
  2.2 : Study of IEEE Papers
  2.3 : Study of Python-Flask and Android Basics

- Task 3: Gathering Requirements
  3.1 : Installing all the necessary S/W like Eclipse ADT bundle, Flask Python Framework
  3.2 : Gathering the H/W required

- Task 4: Designing of the API
  4.1 : Studying an appropriate registration mechanism
  4.2 : Applying an algorithm to appropriately rank the requests from users

- Task 5: Application Development
  5.1 : Designing of the GUI
  5.2 : Development of the Application using Android SDK and the API

- Task 6: Testing and Debugging
- Task 7: Final Documentation

5.3.2 Task network

Project tasks and their dependencies are noted in this diagrammatic form.

5.3.3 Timeline Chart

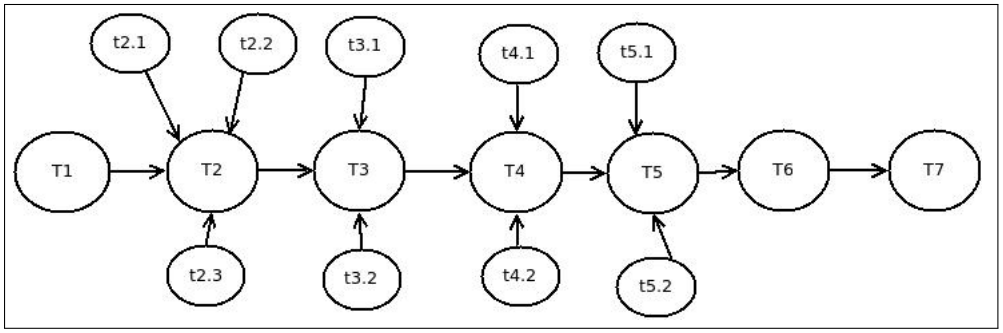


Figure 5.1: Timeline

5.4 TEAM ORGANIZATION

5.4.1 Team structure

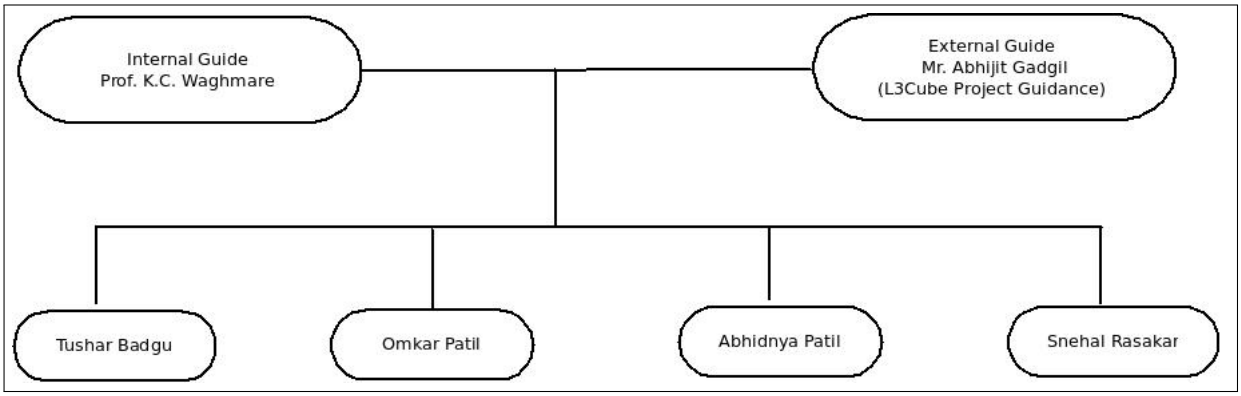


Figure 5.2: Team Structure Diagram

5.4.2 Management reporting and communication

It involves following steps like:

Role	Responsibilities	Participants
Head of Department	1. Approval of project. 2. Providing necessary facilities.	Prof. G.P. Potdar
Project Advisor	1. Provide project oversight and guidance. 2. Review/approve some project elements.	Mr. Abhijit Gadgil (L3Cube)
Project Guide	1. Manages project in accordance to the project plan. 2. Direct/lead team members toward project objectives. 3. Oversee discussions related to design,performance evaluation.	Prof. K.C.Waghmare (PICT-Computer Engineering)
Project Developers	1. Realizing the goals of the project. 2. Implementing the code of the project. 3. Enhancing the scope and functionality of the project.	Tushar Badgu, Omkar Patil, Abhidnya Patil, Snehal Rasakar

Table 5.2: Team Structure

- To maintain a log book and keep detail of all the information about the meeting with the Internal Guide
- To complete the synopsis and give review within the stipulated time
- Weekly update of the assignments given by the external guide
- Team meeting every day or as frequently as possible to discuss progress of

project

- Meeting with internal guide and external guides as frequently as possible
- Logging of work by every team member

CHAPTER 6

SOFTWARE REQUIREMENT SPECIFICATION

6.1 INTRODUCTION

The meta-app will be used by users to get the best results of a service or a product he is searching for online. This will be done without compromising the security of the user by creating a layer of security above android security. An intent will be published for every request from a user to all the sellers subscribed and the responses from them will be ranked appropriately as the final output of the users request.

6.1.1 Purpose and Scope of Document

The purpose of this document is to enlist the various software requirements of the system. It contains the data models that are to be used, the functionality of the system and the relationships that various parts of the system share with each other. This document is intended at the developer of Meta-App.

6.1.2 Overview of responsibilities of Developer

Developer is responsible in deciding the scope of the project. He confirms the goals and allied objectives of the project. He is responsible for gathering the requirements of customer,constraints and further extensions to the project. He is responsible for overall design of the project.
Other responsibilities are listed below:

- To have a comprehensive knowledge about APIs, python flask
- To carefully understand all requirements of the project
- To improve performance of applications

- The developer himself plays the roles of a coder, tester, etc

- To complete the project successfully and scale it on time

6.2 USAGE SCENARIO

There will be a multi user environment.

6.2.1 User profiles

Description of all main actors using use case template is provided.
Actors: Users, Service Providers, System
Main flow:

1. Installation of application on the mobile phone of user
2. Registration of new users to the system
3. Now there are two workflows for the system
4. Intent Publishing for a particular service
   a) The user will provide a decription of the service required along with the deadline
   b) The intents will be broadcasted to all the service providers registered with the system
   c) The responses from service providers are forwarded to the user through a proxy server
5. Service provider requesting user data
   a) Service provider makes a request to obtain some user data
   b) The system will forward the request to the user in need
   c) The application will fetch the data from customized namespace and provide it to the service provider

6. Ranking the responses from various service providers

6.2.2 Use-cases

- Use case name: Installation of application
  Actor: User
  Precondition: User must have an account on Google playstore
  Main flow: The application will be downloaded and installed.

- Use case name: Registration of user
  Actor: User
  Precondition: User should have a working phone number
  Main flow: Registration UI will show up. User inputs the credentials and if they are valid the user is registered to the system

- Use case name: Intent Publishing to the system
  Actor: User
  Precondition: User should provide a meaningful description of the service
  Main flow: The intent publishing role can be fulfilled in various ways such as:
  i) Providing a text description of the service
  ii) Use the flow of application UI to generate an intent

- Use case name: Broadcast the intent via RESTful apis
  Actor: System
  Precondition: The service providers who want to use the system must implement the provided REST apis in their own system
  Main flow: The intents of user will be broadcasted through REST apis. Any responses are forwarded to provider.

- Use case name: Provider Namespace

  Actor: User

  Precondition: User must choose a set of user data which is subset of entire user data which can be afforded to be provided to service providers

  Main flow: The user will be choosing a set of data(e.g contacts,photos,sms) which will be considered a data from this namespace is provided

- Use case name: Ranking the responses

  Actor: System

  Precondition: User must make a intent request and various service providers must respond to the intent request.

  Main flow: Whenever we have more than one response to the intent request then the system must provide a ranked output. The rankings are real-time and user-provided and hence reliable.

6.2.3 Use Case View

Use Case Diagram.

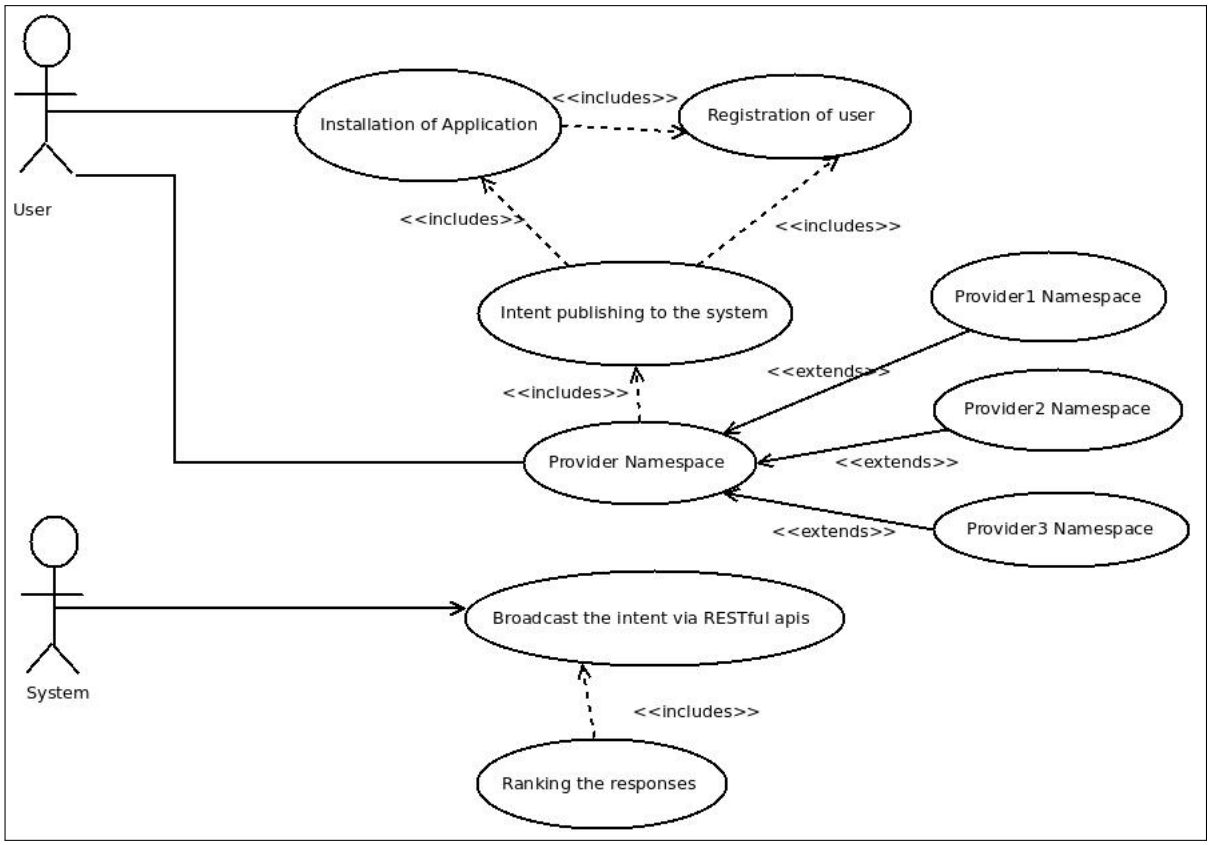


Figure 6.1: Timeline

6.3 DATA MODEL AND DESCRIPTION

6.3.1 Data Description

6.3.2 Data objects and Relationships

Data objects and their major attributes and relationships among data objects are described using an ERD- like form.

6.4 FUNCTIONAL MODEL AND DESCRIPTION

6.4.1 Data Flow Diagram

6.4.1.1 Level 0 Data Flow Diagram

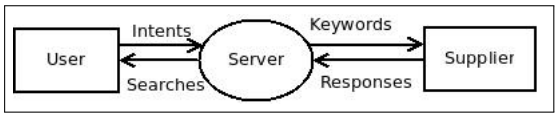


Figure 6.2: Level 0 diagram

6.4.1.2 Level 1 Data Flow Diagram

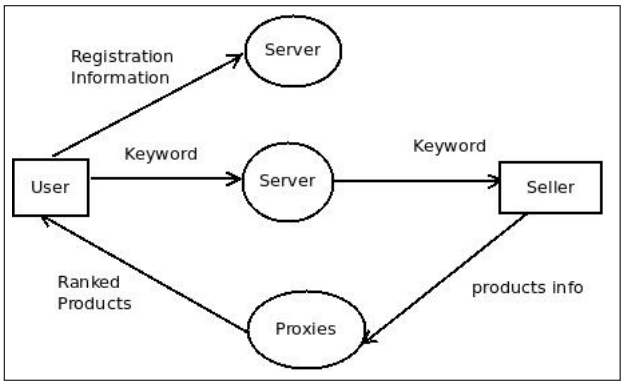


Figure 6.3: Level 1 diagram

6.4.2 Activity Diagram:

- The Activity diagram represents the steps taken.

6.4.3 Non Functional Requirements:

- Interface Requirements
- Performance Requirements

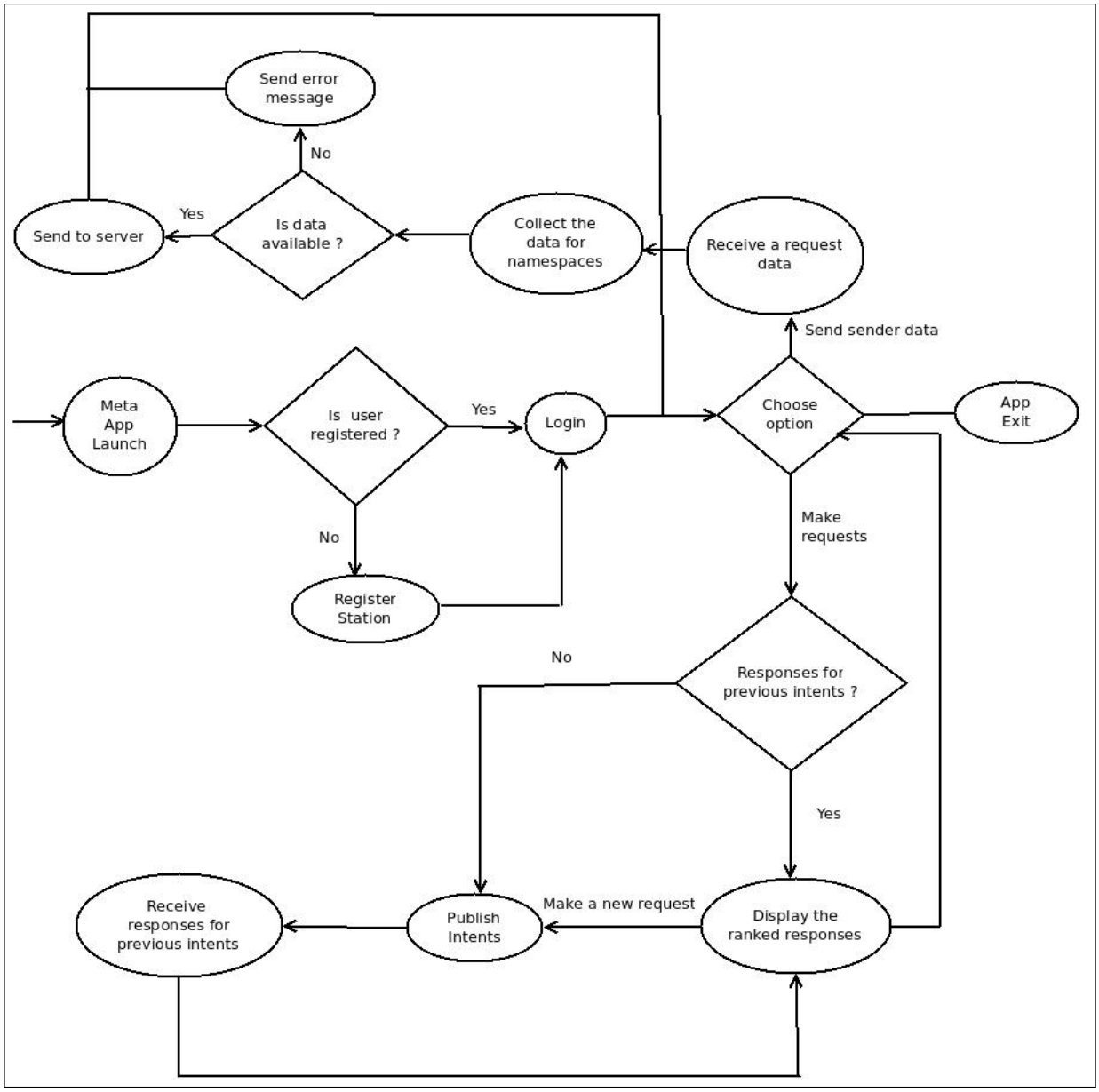


Figure 6.4: Activity diagram

- Software quality attributes such as availability [related to Reliability], modifiability [includes portability, reusability, scalability] , performance, security, testability and usability[includes self adaptability and user adaptability]

6.4.4 State Diagram:

State Transition Diagram Fig.6.5 example shows the state transition diagram. The states are represented in ovals and state of system gets changed when certain events occur. The transitions from one state to the other are represented by arrows. The Figure shows important states and events that occur.

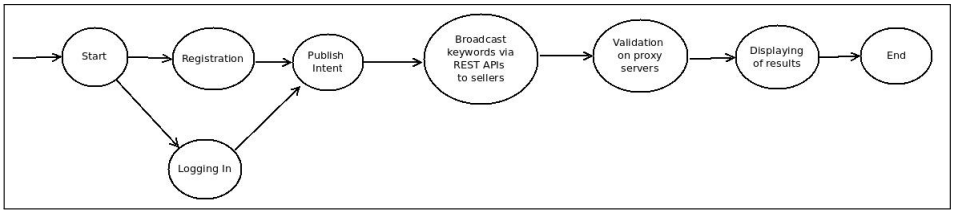


Figure 6.5: State transition diagram

6.4.5 Design Constraints

The Design of the system has following constraints:

- Android App
    1. A few clicks should be able to publish a intent.
    2. The Meta-App requires the android system to provide all the permissions.
    3. Single phone user has single account on the system.No user information stored on the server.
- Server
    1. Number of requests/sec served must be high.
    2. The security mechanism shoukd be strong enough.
    3. No user data stored on the server thus ensuring the user complete anonymity.

6.4.6 Software Interface Description

The software consists of data inputs such as:

- Http Post Requests: The android app does post requests to server with data in particular form.
- Textual Intents

CHAPTER 7

DETAILED DESIGN DOCUMENT USING APPENDIX A AND B

7.1 INTRODUCTION

The purpose of the Software Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to built. The Software Design Document provides information necessary to provide description of the details for the software and system to be built.

7.2 ARCHITECTURAL DESIGN

The application is divided into three modules

- Client-side android app (End-User)

  This app will act as the end point of the system and will perform two tasks first being taking user intents and displaying the offers. second will be generation of a unique token for each request made by the user. this token will be encrypted and sent to the cloud server using a public-private key mechanism.

- A Cloud Service

  This component will be working from the cloud.it will contain two types of servers. One main cloud server which will be performing all the computational tasks and manage the entire system. It will be connected to a database containing all the information about sellers and their services. this database will also include the various valid token currently available.

  It will also contain system to extract keywords from the user intent using natural language processing. These keywords will be used to find the appropriate seller from the database and the intent will be forwarded to all those sellers which offer that service. The token-id will also be sent along with the intent.the token is will act as an open port on the client-app.anyone having this token can publish the offers to the client app.So the cloud server will also contain a module for encryption and decryption.

The second component will multiple proxy servers. These will act as middleware between the sellers and the users.their task will be to check the validity of the token, authenticity of the seller and the weather the intent has been fulfilled.

- Seller-side app

  We will be providing a seller side app to provide information of the intents to the seller.the seller can then send his offer to the user using the token-id manually or he can write an automated system using our APIs.

- Proxy

  The proxy is responsible for invalidation of user requests with the offers received. It will also discard the offers received after deadline and prevent the user to be spammed by unnecessary offers. The proxy is responsible for ranking of the offers. This is done by a ranking algorithm which decides the rank of a seller based on two parameters : 1) An initial score 2) Sentiment score obtained from sentiment analysis engine by working on the received user reviews.

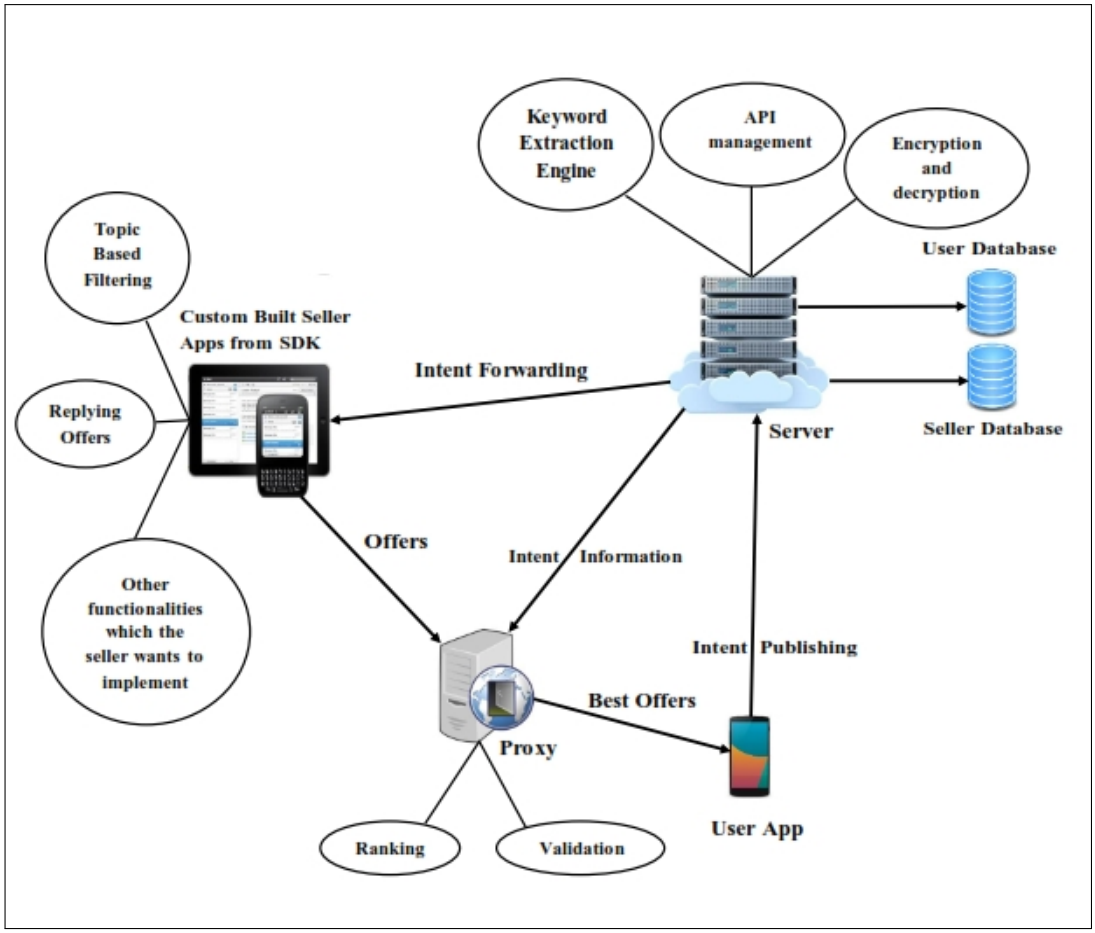


Figure 7.1: Architecture diagram

7.3 DATA DESIGN (USING APPENDICES A AND B)

A description of all data structures including internal, global, and temporary data structures, database design (tables), file formats.

7.3.1 Internal software data structure

- Each User ID and Request ID will be stored on the server for tracking of orders from a particular customer. Along with this the proxy will keep a log of deadlines for each request for invalidations. Also the server public key is stored in the Android Phone.

7.3.2 Global data structure

- The ranking of requests received from the sellers will be done on a training set of 25000 labelled documents in the form of IMDB Movie Reviews. After these offers are retrieved by the app , they will be stored in a local SQLlite database on the Android phones. ALso the log of what all data is being shared with which sellers is stored on the local memory.

7.3.3 Temporary data structure

- If a user is comfortable with sharing some contacts , these will be taken from the user and stored on the server temporarily. After the sellers retrieve this information from the APIs provided to them,this information will be permanently deleted from our servers.

7.3.4 Database description

Databases Used are:
SQLlite : Storing data on Android Application
SQL Alchemy and MongoDB : Storing Data on Server
Labelled Documents : 25000 IMDB Movie Reviews for Sentiment Analysis Engine.

7.4 COMPONENT DESIGN

The program will regularly fetch data from various APIs using the user's access token.
In our system we will be having following components.

- Android smartphones above android 2.2
- Laptops and desktops with configuration atleast i3 processor, 4GB RAM, 250 GB HDD.

7.4.1 Class Diagram

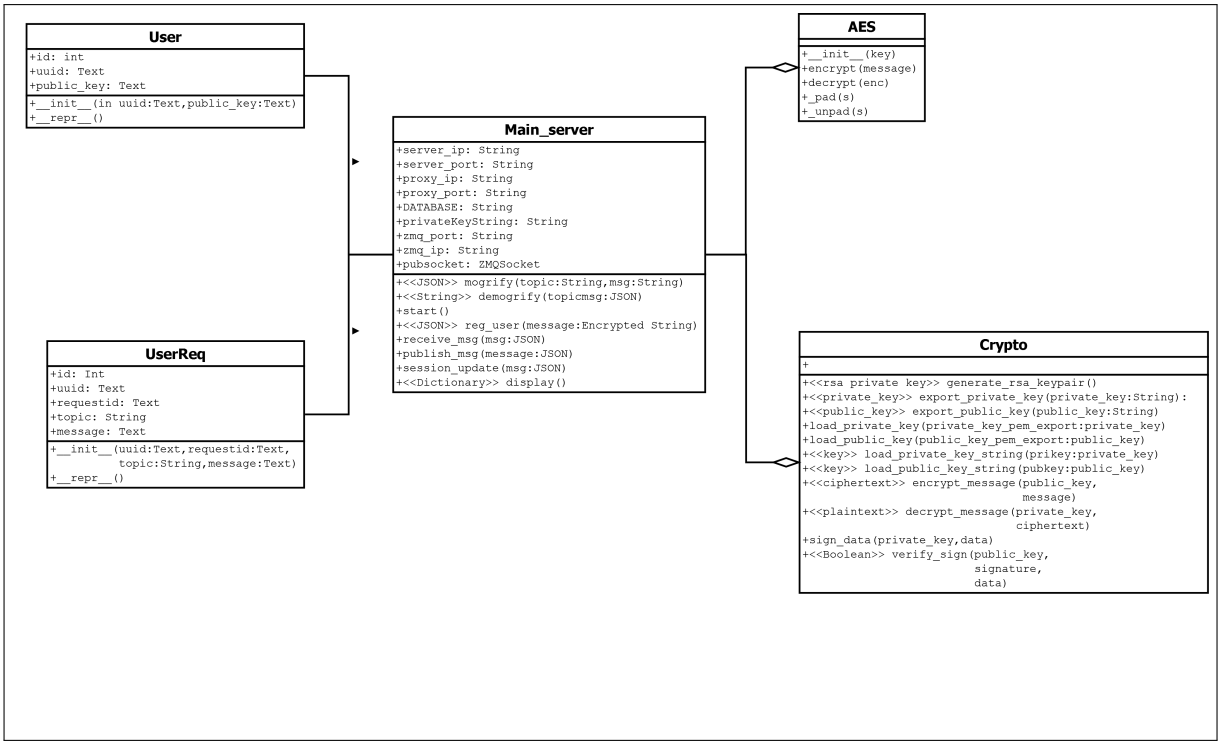


Figure 7.2: Class Diagram1

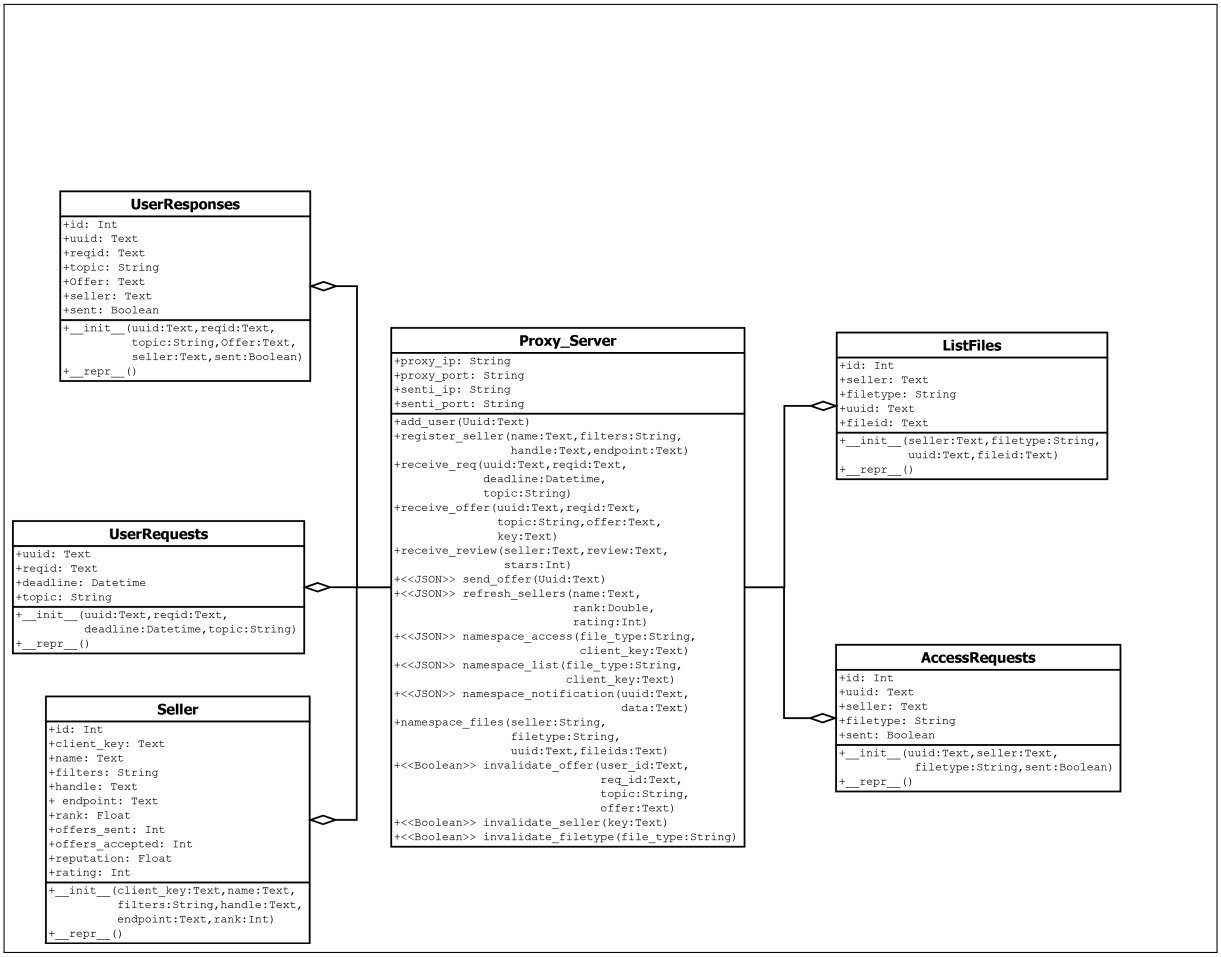


Figure 7.3: Class Diagram2

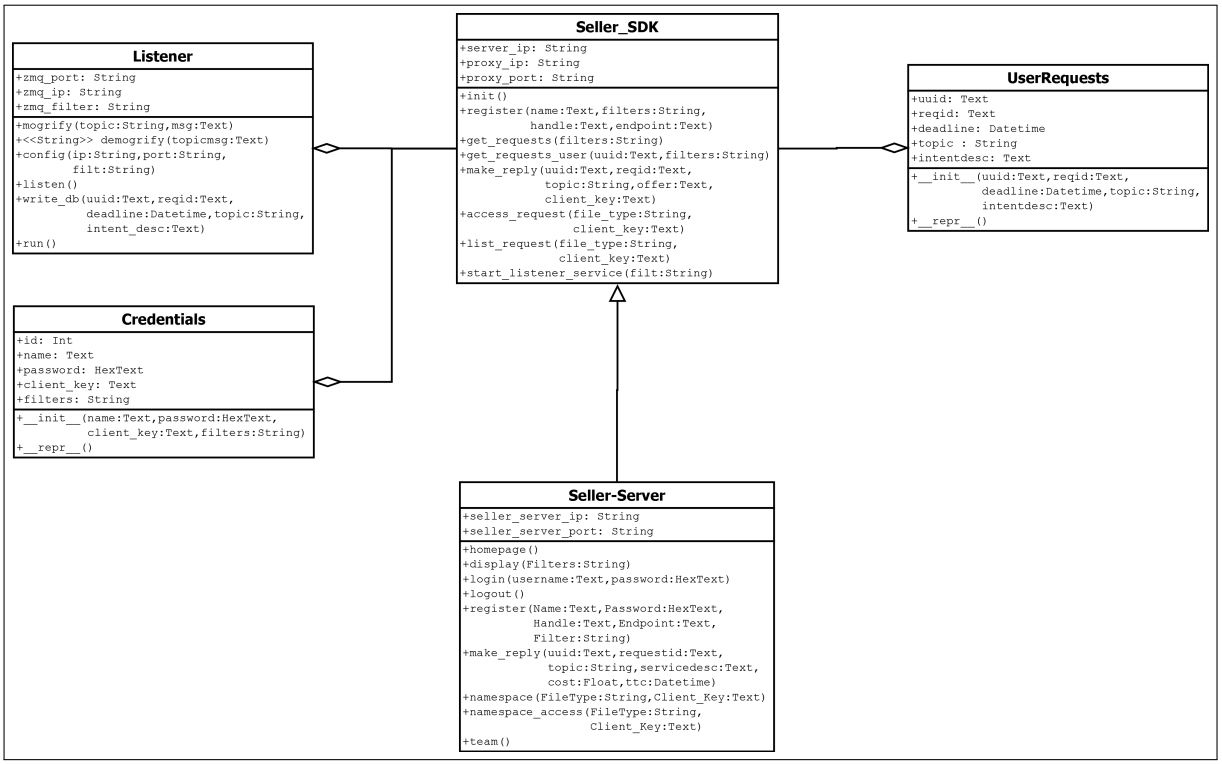


Figure 7.4: Class Diagram3

CHAPTER 8

PROJECT IMPLEMENTATION

8.1 INTRODUCTION

- The entire framework is designed keeping two things in mind i.e. the user identity should not be revealed thus preventing any future spamming by the dealers and giving user a more fine grain control over what is to be shared .The project takes care of users data by encrypting all data going to and fro from the app using a TLS based two level encryption algorithm. No user data is taken during registration of this service by the users thus eliminating any possibility of contacting the user outside this app eco-system.

- A PUBLISHER/SUBSCRIBER model using ZMQ is used to broadcast the user intents to all the registered dealers. Dealer registration is done to prevent spamming of system by unauthorized dealers. A python-SDK is provided to the dealers to implement the APIs released by the service to integrate it with their own servers and websites. These APIs include listen, read, display, reply, etc. using these APIs the dealer sends a reply to the user. however before reaching user it is passed through a proxy which performs invalidation of the offers and then ranks them according to dealers reputation. The best offers are sent to use of which user can select one and give a review when it is completed. this user review will be used to update the ranking of the dealers.

- The Namespaces functionality provides the dealers with an option to get access to certain user data. The user can actually specify what files he is comfortable with sharing and no other data will be visible to the dealer. A Follow utility is also provided where dealers can create handles to run their promotional campaigns and users can follow them to get updates on a news feed.

8.2 TOOLS AND TECHNOLOGIES USED

1. Android Studio is the official integrated development environment (IDE) for Android platform development.

2. Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine.

3. ZeroMQ (also spelled MQ, 0MQ or ZMQ) is a high-performance asynchronous messaging library, aimed at use in distributed or concurrent applications.

4. Python Cryptography Toolkit (pycrypto) is a collection of both secure hash functions (such as SHA256 and RIPEMD160), and various encryption algorithms (AES, DES, RSA, ElGamal, etc.).

5. Javax.crypto package provides the classes and interfaces for cryptographic applications implementing algorithms for encryption, decryption, or key agreement.

6. SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

7. PyMongo is a Python distribution containing tools for working with MongoDB, and is the recommended way to work with MongoDB from Python.

8. context

8.3 METHODOLOGIES/ALGORITHM DETAILS

8.3.1 Algorithm

1. User ID generation and encryption based on TLS Whenever the user registers on the application it will perform following :

   (a) The server will have its own pair of RSA public-private key. The server public key will be saved in the app to encrypt all outgoing data from the app to server.

   (b) Whenever a user registers he will get his unique UUID from the server. Along with it an RSA algorithm will run on the app to generate a user specific Public-Private key.

(c) The UUID will be generated using python uid generation library which uses timestamps to generate a unique id which can be used as UUID.

(d) The User-Public key will be stored in the server to create a downlink encryption channel.

(e) However time required for data encryption and decryption is large for RSA. So we propose a session channel.

(f) We propose creating session for data exchange of fixed time-frame which can be decided by the user. RSA will be used to manage establishment of these session.

(g) The client public key and UUID will be given to an AES algorithm to generate a cipher text.

(h) The seed value of AES which will be used to generate the cipher text will then be encrypted using server-public key to generate a cipher-seed and sent to server.

(i) The server can then decrypt the seed value from the cipher-seed using server-private key and use it to decrypt the data sent by the client.

(j) This seed-value will vary from session to session and would thus reduce overheads.

(k) The UUID generated will be used for identifying user requests and categorization of the same.

2. Ranking Algorithm It is used to rank the dealers based on two parameters

(a) Frequency of occurrence

Fi = Oi / $\sum O$

$Where,$

$Oi = Numberofusersacceptingtheoffer$

$\sum O = Totalnumberofofferssentbysellers.$

(b) Dealer reputation

Ri = Reputation

Vi = Sentiment value of Review

Initially Ri is 0 for all the dealers Di

After a review Vi arrives for a dealer Di

Ri = Ri-1 + Vi (New reputation of Di)

Rj = Rj-1 - [m(Rj-1 - Rmax)* Vi/n] (New reputation of Dj where ji)

Where m = n/ 2(Rmin - Rmax)

n = total number of dealers

Final Rank :

Rank = Fi Ri

CHAPTER 9

SOFTWARE TESTING

The Testing phase forms an important part of the software development life cycle. Any software product has to be tested thoroughly before it is delivered to the end customer. This document provides a general overview of the testing strategy adopted for testing our product.

9.1 TYPE OF TESTING USED

The project can be considered as a combination of several modules. In case of our project, it consists of a User Application, Server, Seller Sdk and a Demonstration website. So, each such module can be considered as composed of different units which are responsible for the working of the entire module. Thus, we need to test each and every module independently as well as combine together to satisfy the user requirements.

9.1.1 Unit Testing

Unit testing is testing the smallest testable part of a application. This is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. Unit can be entire module or a function or procedure. The tools used for testing different modules are:

(a) User Application (Android)
Junit Runner and Monkey

(b) Server and Seller SDK (Python)
Unit Tests

(c) Seller Website (HTML5, CSS3)
Selenium Tool

9.1.2 Integration Testing

Integration testing corresponds to a phase in software testing in which individual software modules (which are unit tested) are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing can expose

problems with the interfaces among program components before trouble occurs in real-world program execution. Integration testing is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision. There are two major ways of carrying out an integration test, called the bottom-up method and the top-down method. We have used bottom up test methodology.

Integration Testing in case of our project aims to formulate a test plan and a suite of test cases to test the functionalities of the modules combined together from end to end. The Android Application has a range of functionalities represented by a GUI in the App but requires a bug-free server responding to such requests. All such cases are considered in Integration Testing.

9.1.3 Automated Testing

Automated testing of the web application hosted is done by using Selenium.Selenium is a portable software testing framework for web applications. Selenium provides a record/playback tool for authoring tests without learning a test scripting language (Selenium IDE). Various functionalities were tested using this tool.

9.1.4 Validation Testing

The intents published are stored in a database. Validations used while storing the data such as ensuring unique entry for each user, each request of a user is uniquely identified, and the registered sellers along with their reputation scores are validated in this Testing phase.

9.1.5 High order Testing

High order testing was done after completely in- tegrating all the modules of the system using various test cases.

9.2 TEST CASES AND TEST RESULTS

9.2.1 Test cases in Unit Testing

(a) User Application

Test Case 1:

Aim: Encryption and Decryption of User Data on Phone.

Test data: A text to be retrieved from test file.

Expected Result: Encrypted text must be obtained and orginal text must be obtained from the encrypted text by performing decryption.

Actual Result: Same as expected.

Test Case 2:

Aim: File manager utility for Namespace module of App.

Test data: Select a list of files using the module.

Expected Result: The files must be selected and viewable inside the namespaces tab.

Actual Result: Files can be opened using installed apps and also perform other activities.

Test Case 3:

Aim: Session Expiry of the User App

Test data: Change the user phone date to date after session expiry.

Expected Result: App should initiate initial handshake to exchange the encryption-decryption keys.

Actual Result: New keys generated.

(b) Server and Seller SDK

Test Case 1:

Aim: Encryption and Decryption of Data on Server.

Test data: A text to be retrieved from test file.

Expected Result: Encrypted text must be obtained and orginal text must be obtained from the encrypted text by performing decryption.
Actual Result: Same as expected.

Test Case 2:
Aim: ZMQ Publish and Subscribe module for publishing intents.
Test data: Test files contain a list of text to be published along with the topics.
Expected Result: Listener for a particular topic must receive all the text with that topic.
Actual Result: Same as expected.

(c) Seller Website

Test Case 1:
Aim: Registration and Login Components.
Test data: Test files contain a list of credentials to test registration and login.
Expected Result: Duplicate entries are discarded for registration.
Actual Result: Only the successful registrations are allowed to login.

9.2.2 Test cases in Integration Testing

Manual Testing methodology was used for performing Integration testing of the various units integrated together. The various tests performed are given below:

TestCase ID	Objective	Case	Procedure	Expected Result	Actual Result	Pass/ Fail
Publish Intents	To test publishing of intents to server	1.	a) Select food category b) Enter Intent Description and Expiry Date and Time c) Press Publish	Request should be published and should be recieved by each seller subscribed to food topic.	Same as expected	Pass
		2.	a) Select food category b) Leave Intent Description field empty. c) Press Publish	Request should be discarded as the required fields are not provided.	A toast message specifying empty field will be displayed.	Pass
		3.	a) Select food category b) Expiry Date is set to invalid date. c) Press Publish	User should be restricted to select only the upcoming dates.	The calender displayed in App does not allow user to select previous dates.	Pass
Namespace Utility	To test various functions included in Namespaces.	1.	a) A seller sends an acess request. b) Select Namespace tab in App drawer.	A Notification for each such access requets from each seller should be displayed to User.	The Notifications which are not yet delivered to user are fetched and displayed.	Pass
		2.	a) Select a File from Namespace section and provide access to a first seller. b) Again provide access of another file to second seller.	First seller can only view the files which are permitted to him and not the second seller.	List requests sent from each such seller will result in a different set of files accessible.	Pass
		3.	a) Select Logs tab from the App drawer.	A list of logs recording the access requests sent by the sellers as well as files provided access to a seller.	Logs are displayed and are persistent since app installation.	Pass

Fetch Responses	To test fetching of responses on android and sending of responsed using sdk/website.	1. 2.	a) Select Offers tab from the App drawer. a) Send a response to a particular user request using the seller sdk. b) Fetch the responses by selecting the offers tab.	All the unfetched responses for user requests must be displayed. The response must be fetched and dispalyed within the offers for that particular request.	Same as expected. Same as expected.	Pass Pass
Ranking Mechanism	To test the accuracy of sentiment scores generated and evaluate the ranking mechanism	1. 2. 3.	a) Select any offer from the offers tab. b) Provide a textual user review as well as ratings. c) Click on submit. a) Send a postivite text review to the sentiment engine. a) Generate a random text and send it for evaluation to the sentiment engine.	The user review is sent to sentiment analysis engine and eventually change the rankings of the sellers. Postive score must be greater than 0.5 Positive and negative scores must be balanced as the text doesnt have a meaning.	Ranks of the sellers are changed accordingly. Postive score is greater than negative score. Positive Score approximately equal to negative score.	Pass Pass Pass

Registration and Login	To test seller registration and login.	1.	a) Select registration tab on the seller website. b) Provide the information required. c) Click on Register Button.	Unique Seller Client Key should be generated.	Client key obtained.	Pass
		2.	a) Name and/or handle provided already is used by another seller. b) Register	Registration should be unsucessfull as the constraints are not met.	Error message displayed indicating the same Name or Handle or both.	Pass
		3.	a) Enter valid username and password on login page b) Press Login	User redirected to the Homepage.	Same as expected.	Pass
		4.	a) Enter invalid username and password. b) Press Login	Error message should be displayed and user stays on login page.	Same as expected.	Pass

9.2.3 Test cases in Automated Testing

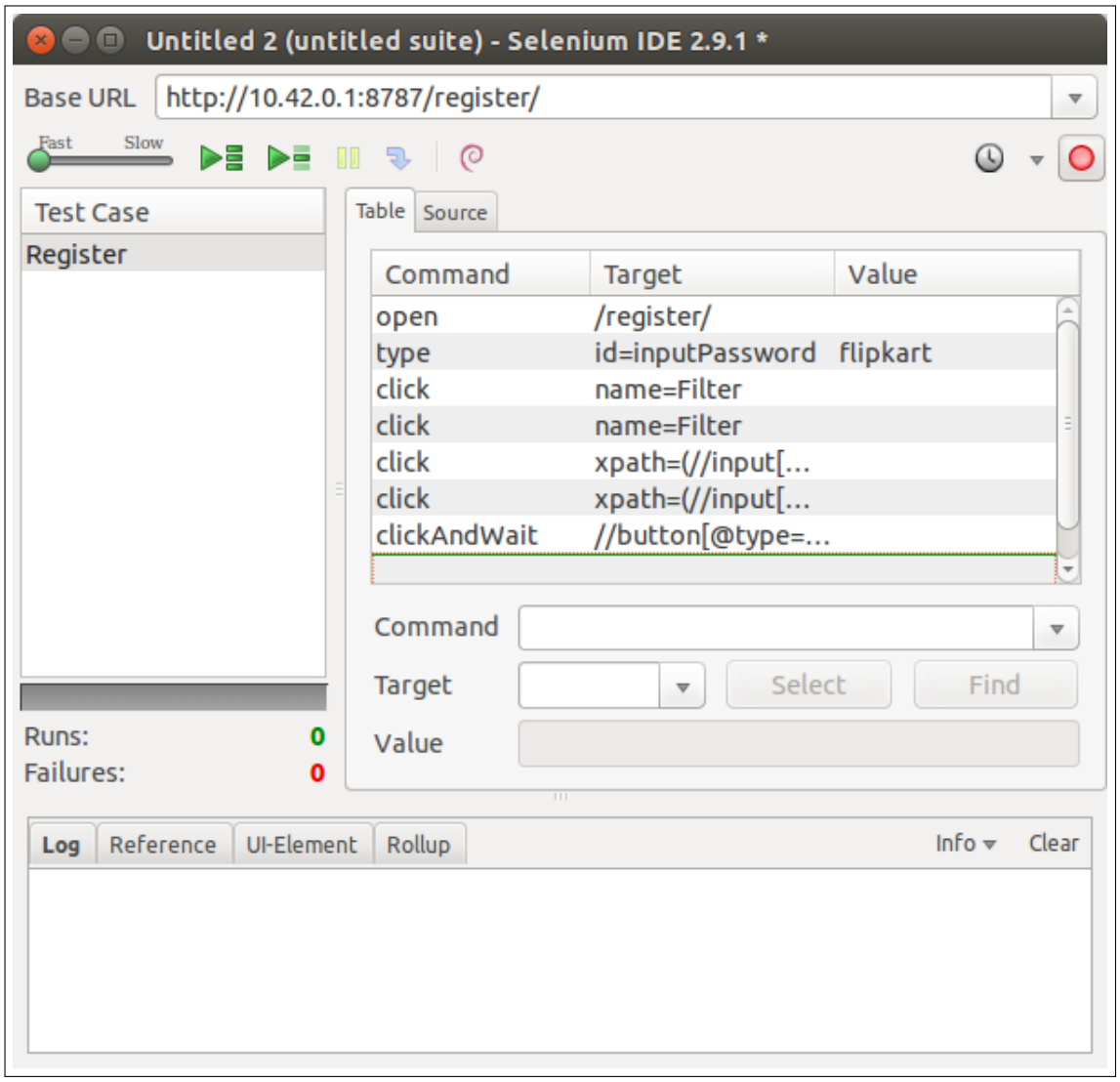


Figure 9.1: Selenium Testing Tool for Registration

9.2.4 Test cases in Validation Testing

Test Case 1:

Aim: Registration of Seller.

Test data: A list of values to be used for registration.

Expected Result: Unique Name and Handle should be provided. If not then Registration fails otherwise it is successfull and a seller client key is generated.

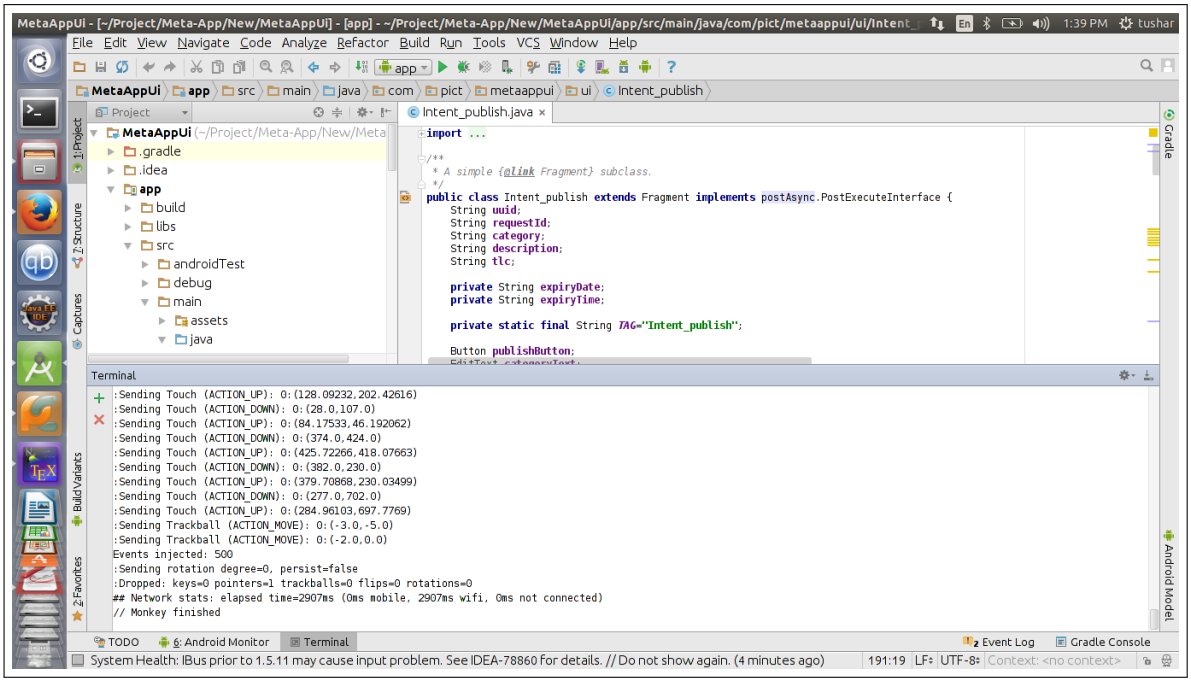


Figure 9.2: Monkey Testing Tool for Android App

Actual Result: Same as expected.

Test Case 2:

Aim: Validation of sellers.

Test data: Any function from the SDK is executed.

Expected Result: The client key provided by the seller should be validated and accordingly the response is sent.

Actual Result: Service is denied if the client key is incorrect.

9.2.5 Test cases in High order Testing

Test Case 1:

Aim: Complete end to end testing.

Test data: Any function from the App/Seller Website is executed.

Expected Result: The intents from a user are published to athe respective sellers and the sellers are able to respond to the intents with their offers. The users can provide a review to rank the sellers.

Actual Result: Same as expected.

CHAPTER 10

RESULTS

10.1 SCREEN SHOTS

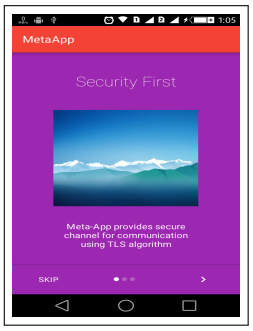


Figure 10.1: App Startup screen

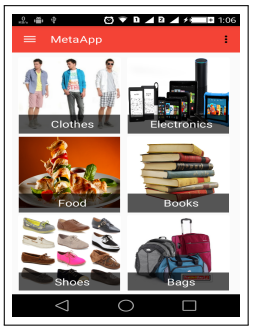


Figure 10.2: Categories tab

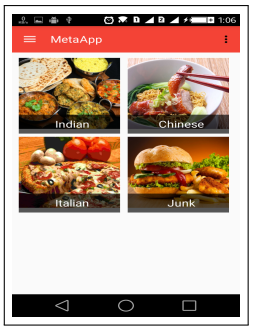


Figure 10.3: Detailed Food category

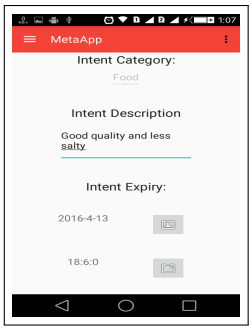


Figure 10.4: Intent publish screen

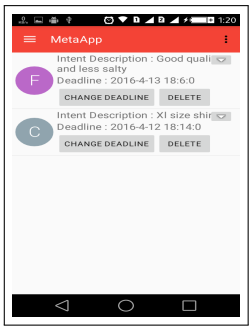


Figure 10.5: Offers tab

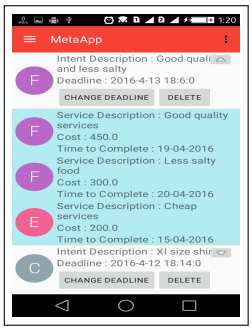


Figure 10.6: Expanded requests with all the offers

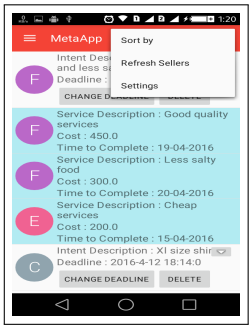


Figure 10.7: Options for Offers tab

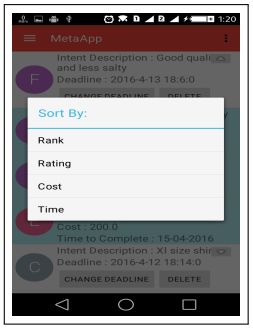


Figure 10.8: Sort the offers by selecting a criterion

Figure 10.9: Sort the offers by cost

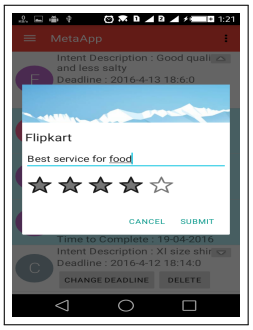


Figure 10.10: Send review for an offer of a particular seller (In this case flipkart)

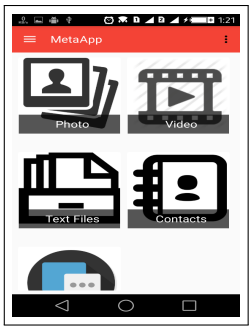


Figure 10.11: Namespaces tab

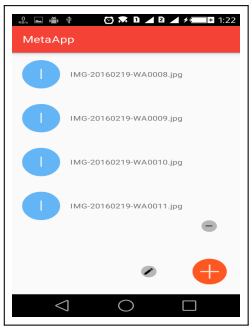


Figure 10.12: Select photos to be provided access to different sellers

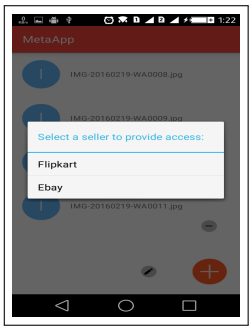


Figure 10.13: Access provided to a file

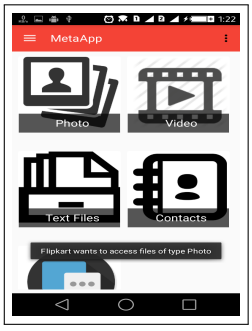


Figure 10.14: Notifications for access requests from different sellers

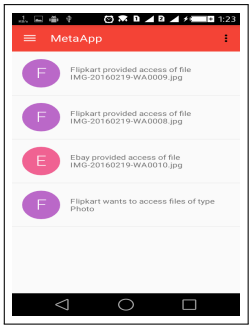


Figure 10.15: Logs of namespaces

10.2 OUTPUTS

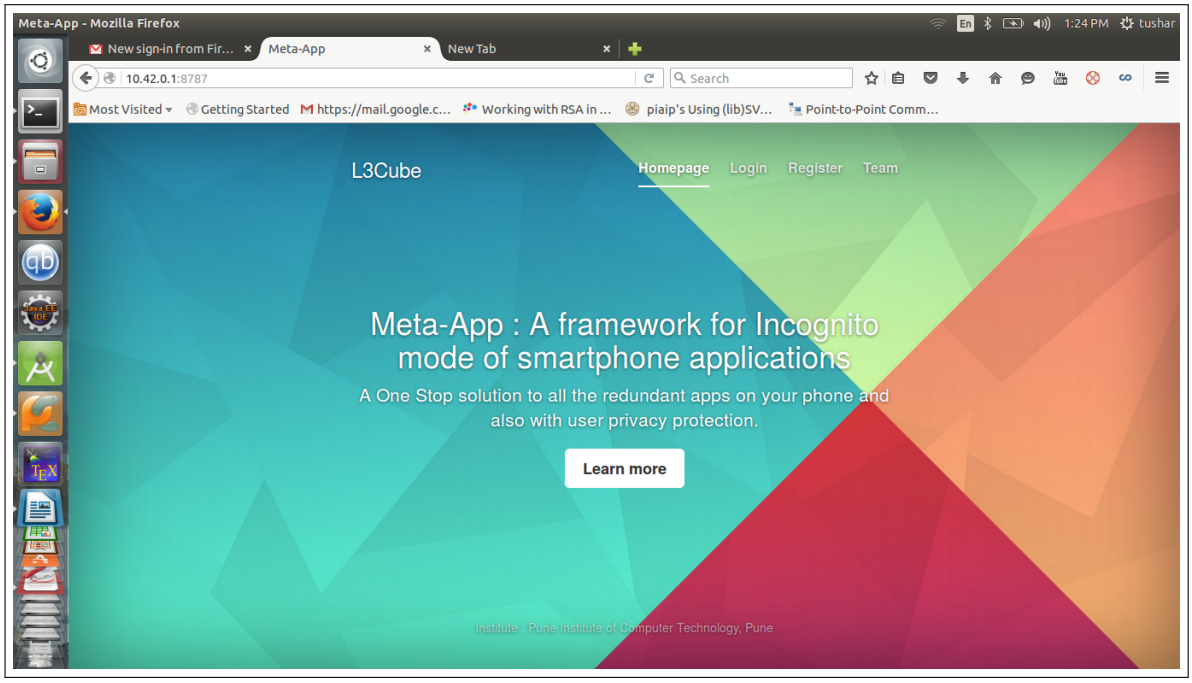


Figure 10.16: MetaApp seller website

Figure 10.17: Registration for a seller(Flipkart)

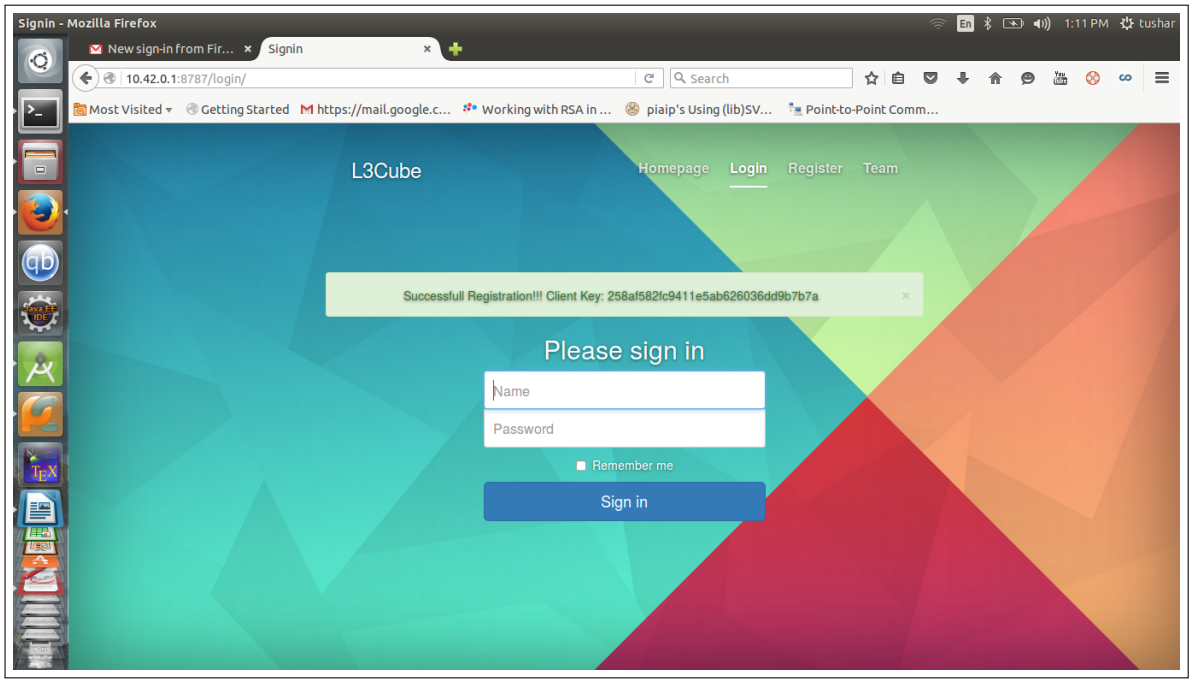


Figure 10.18: Successfull registration of seller

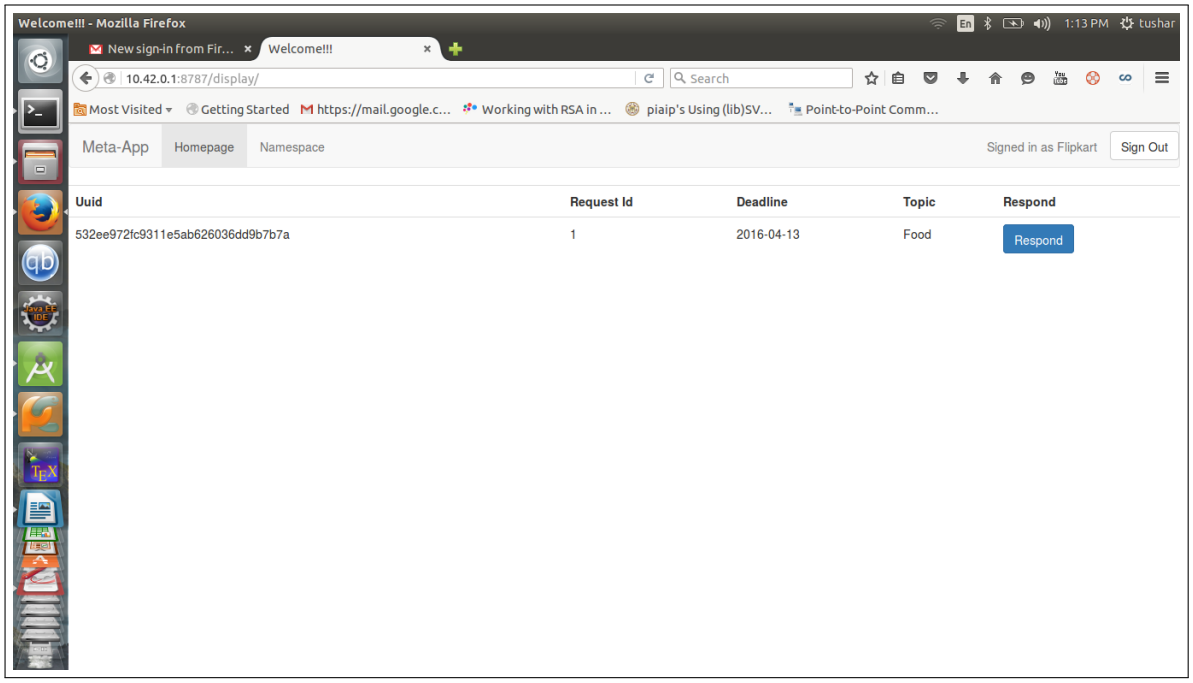


Figure 10.19: Homepage showing all the intents from users

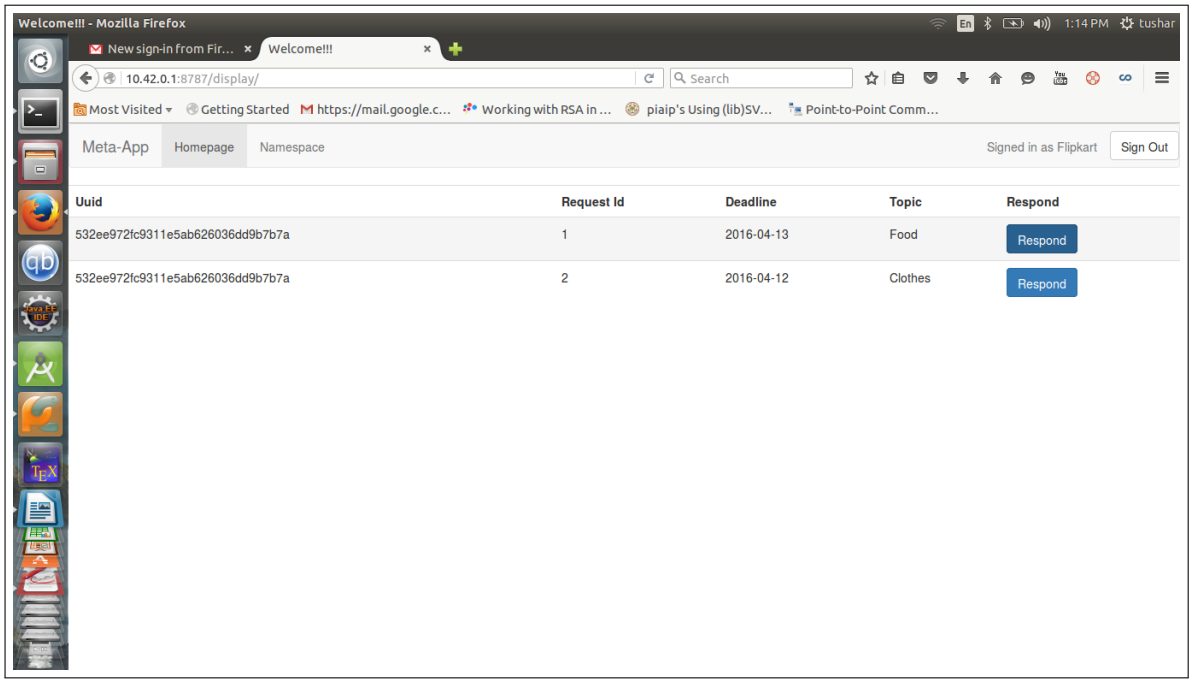


Figure 10.20: New intents

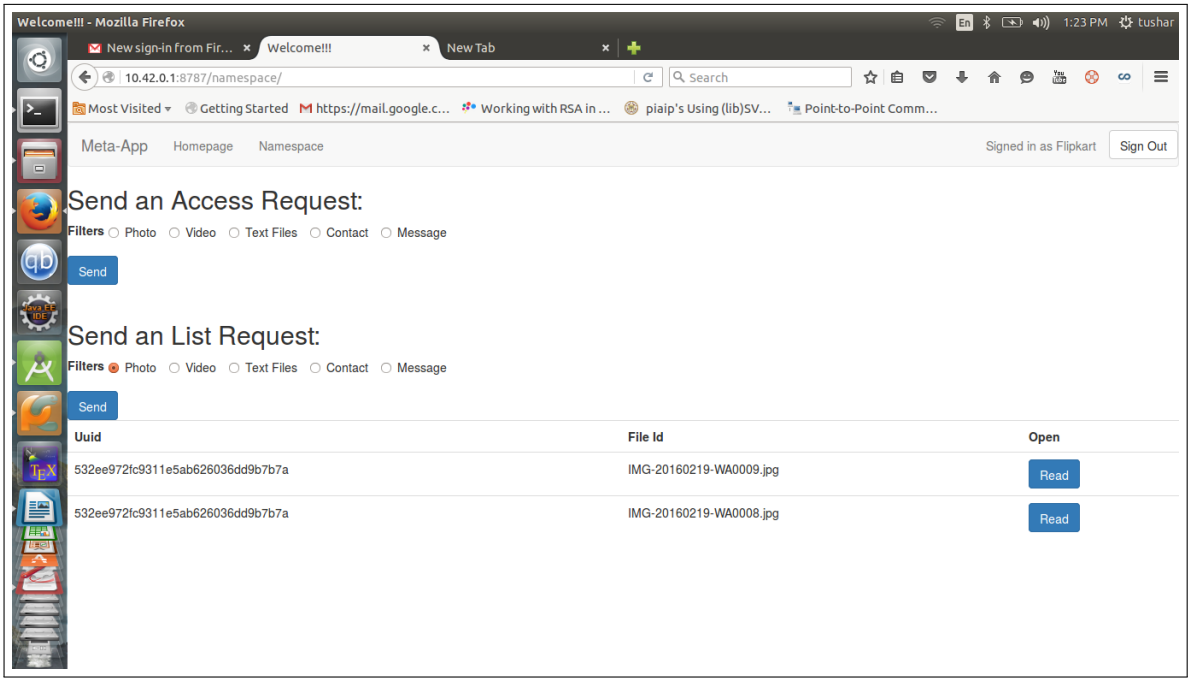


Figure 10.21: Namespace showing options to send access and list requests

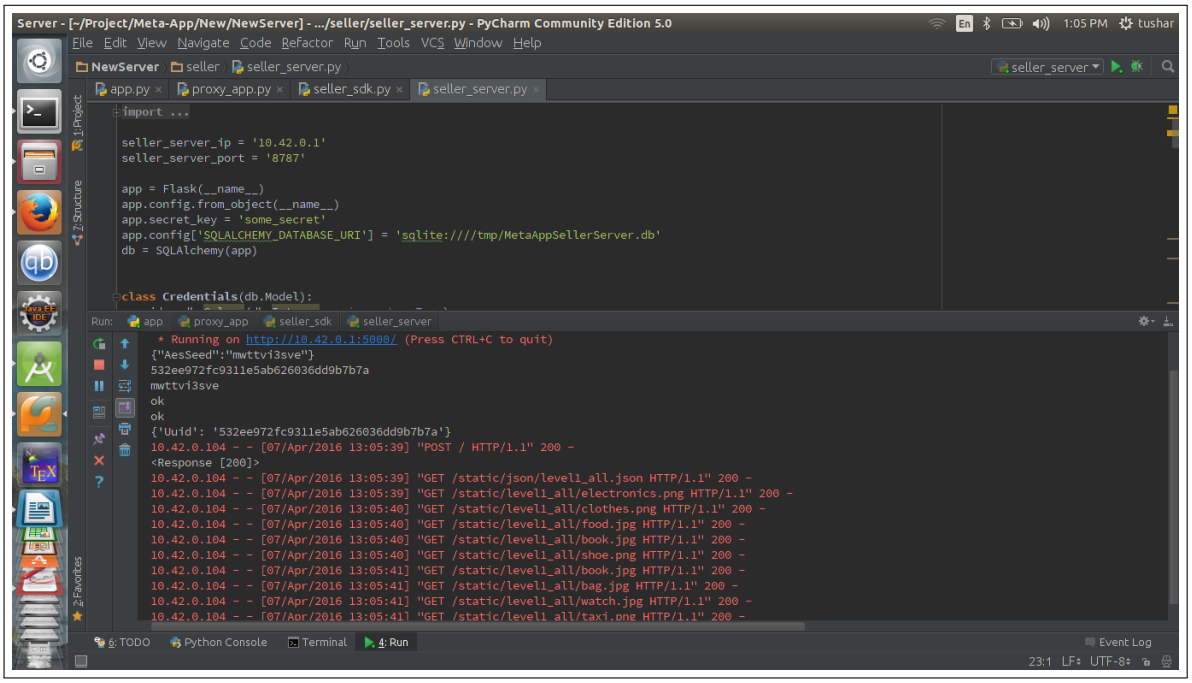


Figure 10.22: Server screenshot showing user registration and Uuid generation for user

CHAPTER 11

DEPLOYMENT AND MAINTENANCE

11.1 INSTALLATION AND UN-INSTALLATION

Installation Manual for Application Users :

1. Make sure that the user is connected to active internet connection.
2. The apk file of the android app will be provided through a platform like Google Play.
3. After installing this apk the app will be installed and the user will be registered without asking for registration details.
4. The installation is complete and the user can browse through various sections.

Installation Manual for Sellers using the SDK:

The sellers will be provided an SDK which can perform functions of register, listen and reply. For accesing the user information the functions provided are access request, listen and list request. This SDK can be integrated with their existing ecosystem , i.e, an already existing web site or however they want.

11.2 USER HELP

These are the guidelines for the Application User:

The sections user can navigate to are:

1. Categories : where all the different categories will be listed. After following the subcategories , you will reach to an intent publishing page where you can specify your order details and send this intent to the sellers.
2. Offers: The offers from the sellers of the requests you made will be displayed here. This page also allows you to sort the offers based on Rank, rating, Cost and Time.After clicking on the offer you went for , you can review the seller by giving him a rating and a user review.

3. User Namespace : Here you will be given the options to select what information are you comfortable sharing with the user. The categories of data are photos, videos, text files, contacts and messages. After further navigation you will be able to select an item by clicking on the + sign at the right bottom of the screen.By clicking on the appropriate choice your SD card contents will be displayed and you can select the respective items. After adding them to the sharable items , you just need to choose and send to the seller requesting it.

4. Logs: The logs of the shared data will be listed on this page.

5. Follow Utility: This will show an instagram like timeline of the newest posts by all the different sellers the user is following.

Index

Methods

1. register

Register the seller with the system to obtain a seller client key to be used for each further requests.

Request

Function Name	Method	URL
register	**POST**	register/seller/

Params	Values
Name	string(unique)
Filter	string
Handle	string(unique)
Endpoint	string

Name
`Name` must be sent with each registration request. It specifies the name of the seller/service provider. Also this field must be unique i.e no other seller must have taken this name.

Filter
`Filter` specifies the type of requests to be forwarded to the seller. It must be a space seprated list of filters specified as a string.
List of filters are : Food, Clothes, Taxi, Electronics, Books, Shoes

Handle
`Handle` specifies the handle of the seller for namespace follow utility. It must also be unique.

Endpoint
`Endpoint` specifies the point of contact for that particular seller which can be a website, mobile app or as simple as a telephone number.

Response

Status	Response

200	{
	"Client_Key": <client_key>
	}
	
	client_key (**string**) - all further API calls must have this key for authorization.

404	{"error":"Name or Handle already exists."}

2. get requests

Retreives a list of all the requests matching the filters specified by the seller during registration process.

Request

Function Name	Method	URL
get_requests	**DB FETCH**	---

Params	Values
Filter	string

Filter
Filter specifies the type of requests to be forwarded to the seller. It must be a space seprated list of filters specified as a string.
List of filters are : Food, Clothes, Taxi, Electronics, Books, Shoes

Response

Status	Response
200	[{ "Uuid": <uuid>, "RequestId": <reqid>, "Topic": <topic>, "Intent": <intent>, "Deadline": <deadline> }, ...] uuid (**string**) - Id representing a user. reqid (**string**) - Id of a request from a particular user. So, uuid and reqid together will uniquely identify a particular request. topic (**string**) - category of the request matching any one of the filters. intent (**string**) - intent description of the request as specified by user. deadline (**string**) - deadline of the request specified in YYYY-MM-DD format.
404	{"error":"No requests to be fetched"}

3. get requests by user

Retreives a list of all the requests of a particular user indicated by uuid matching the filters specified by the seller during registration process

Request

Function Name	Method	URL
get_requests_user	**DB FETCH**	---

Params	Values
Uuid	string
Filter	string

Uuid
Uuid specifies the user whose requests are to be fetched from the database.

Filter
Filter specifies the type of requests to be forwarded to the seller. It must be a space seprated list of filters specified as a string.
List of filters are : Food, Clothes, Taxi, Electronics, Books, Shoes

Response

Status	Response
200	[{ "Uuid": <uuid>, "RequestId": <reqid>, "Topic": <topic>, "Intent": <intent>, "Deadline": <deadline> }, ...] uuid (**string**) - Id representing a user. reqid (**string**) - Id of a request from a particular user. So, uuid and reqid together will uniquely identify a particular request. topic (**string**) - category of the request matching any one of the filters. intent (**string**) - intent description of the request as specified by user. deadline (**string**) - deadline of the request specified in YYYY-MM-DD format.
404	{"error":"Incorrect Uuid"}
404	{"error":"No requests to be fetched"}

4. make reply

Enables the seller to respond to a particular request by specifying the description of the service provided by the seller.

Request

Function Name	Method	URL
make_reply	**POST**	receive/offer/

Params	Values
Uuid	string
RequestId	string
Topic	string
Offer	string
Key	string

Uuid
Uuid must be sent with each reply request to specify the user for whom the reply is generated.

RequestId
RequestId is used to identify a particular request made by a user among several requests. Uuid and RequestId form a unique tuple for each request.

Topic
Topic specifies the category of the request for which a response is generated.

Offer
Offer specifies the services provided by the seller for that kind of request. Offer param is a json string containing fields 'Service Description', 'Cost' and 'Time to complete'.

Key
Key provides the seller client key obtained by that seller during registration. It is used for authorization of the request made by the seller.

Response

Status	Response
200	{"status":"Offer received."}
404	{"error":"Invalid Client Key."}
404	{"error":"Offer discarded."}

5. send access request

Enables the seller to send a access request to all the users to access a particular user data such as photos, contacts, messages, files.

Request

Function Name	Method	URL
access_request	**POST**	namespace/access/

Params	Values
FileType Key	string string

FileType
FileType must be sent with each request to specify the type of user data to be accessed by the seller. It can be anyone of : Photo, Video, Text Files, Contacts, Message.

Key
Key provides the seller client key obtained by that seller during registration. It is used for authorization of the request made by the seller.

Response

Status	Response
200	{"status":"Request Sent."}
404	{"error":"Invalid Client Key."}
404	{"error":"Invalid File Type."}

6. send list request

Enables the seller to send a list request which fetches the files permitted to be accessed by the users for that seller.

Request

Function Name	Method	URL
list_request	**POST**	namespace/list/

Params	Values
FileType Key	string string

FileType
FileType must be sent with each request to specify the type of user data to be accessed by the seller. It can be anyone of : Photo, Video, Text Files, Contacts, Message.

Key
Key provides the seller client key obtained by that seller during registration. It is used for authorization of the request made by the seller.

Response

Status	Response
200	[{ "Uuid": <uuid>, "FileId": <fileid> }, ...] uuid (**string**) - Id representing a user. fileid (**string**) - Name of the file permitted by the user to be accessed by that seller.
404	{"error":"Invalid Client Key."}
404	{"error":"Invalid File Type."}

Glossary

Conventions

- **Client** - Client application.
- **Status** - HTTP status code of response.
- **Function Name** - Name of the function to be invoked from the sdk with the required parameters.
- All the possible responses are listed under ‘Responses’ for each method. Only one of them is issued per request server.
- All response are in JSON format.
- All request parameters are mandatory unless explicitly marked as [optional]
- The type of values accepted for a *request* parameter are shown the the values column like this [**10**|<any number>] .The | symbol means *OR*. If the parameter is [optional], the default value is shown in blue bold text, as **10** is written in [**10**|<any number>].

Status Codes

All status codes are standard HTTP status codes. The below ones are used in this API.

2XX - Success of some kind
4XX - Error occurred in client’s part
5XX - Error occurred in server’s part

Status Code	Description
200	OK
201	Created
202	Accepted (Request accepted, and queued for execution)
400	Bad request
401	Authentication failure
403	Forbidden
404	Resource not found

405	Method Not Allowed
409	Conflict
412	Precondition Failed
413	Request Entity Too Large
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

CHAPTER 12

CONCLUSION AND FUTURE SCOPE

Conclusion:

The framework put forward is successfully implemented and provides various functionalities that where thought of like ranking, user namespace and user privacy. Developing android applications, making servers using python flask, sentiment analysis, designing efficient security mechanisms and handling a distributed environment are some of the things we learned in the process of making this project.

The future scope of the project involves:

1. Increasing the training corpus for sentiment analysis after actual user reviews are appended.
2. Deploying an anonymous chat server for user and the seller for communication after the offer has been chosen and delivery and payment needs to be made.
3. Adding more parameters to the ranking algorithm
4. Provide more APIs to the sellers.

ANNEXURE A

REFERENCES

1 Taenam Cho, Seung-Hyun Seo, Vulnerabilities of Android Data Sharing and Malicious Application to Leaking Private Information, 2013.

2 Nai-Wei Lo, Kuo-Hui Yeh, Leakage Detection and Risk Assessment on Privacy for Android Applications: LRPdroid, 2014.

3 Hongliang Liang, Dongyang Wu, Jiuyun Xu, Hengtai Ma, Suvery on Privacy Protection of Android Devices, 2015.

4 Kuo-Hui Yeh ,Nai-Wei Lo, Chuan-Yen Fan An Analysis Framework for Information Loss and Privacy Leakage on Android Applications,2014

5 Muneer Ahmad Dar, Javed Parvez, Enhancing Security of Android & IOS by Implementing Need-Based Security (NBS), 2014.

6 ZheMin Yang, Min Yang, LeakMiner: Detect information leakage on Android with static taint analysis, 2012.

7 Zang J, Dummit K, Graves J, Lisker P, Sweeney L. Who Knows What About Me? A Survey of Behind the Scenes Personal Data Sharing to Third Parties by Mobile Apps. Technology Science. 2015103001. October 30, 2015. http://techscience.org/a/2015103001

8 Thejovardhana S. Kote and S. R. Jeyashankher, A large scale publish subscribe platform for information delivery to mobile phones, 2006.

9 Keke Chen, Ya Zhang, Zhaohui Zheng, Adapting Ranking functions to User Preference, 2008.

10 Xumin Liu, Arpeet Kale, Javed Wasani, Extracting, Ranking, and Evaluating Quality features of web services through User Review Sentiment Analysis, 2015

ANNEXURE B

LABORATORY ASSIGNMENTS ON PROJECT ANALYSIS OF ALGORITHMIC DESIGN

I	D	E	A
Increase	Drive	Educate	Accelerate
Improve	Deliver	Evaluate	Associate
Ignore	Decrease	Eliminate	Avoid

Table B.1: IDEA Matrix

- Increase Improve Ignore Drive Deliver Decrease Educate Evaluate Eliminate Accelerate Associate Avoid The explanation of the above matrix in regards of problem under consideration is:

  1. Increase: Increase user privacy

  2. Improve: Improve the ease of online service procurement

  3. Ignore: Ignore the use of traditional push based systems

  4. Drive: Drive towards increase in trust of users over services

  5.Deliver: To deliver a new framework for user intent publishing and offer acquistion

  6. Decrease: Decrease the need of installing unnecessary apps

  7. Educate: Educate the users of the vulnerability of current applications

  8. Evaluate: Evaluate the products provided by the sellers

  9. Eliminate: Eliminate the need of redundant apps

  10. Accelerate: Accelerate the rate of communication mechanism between the seller and the user.

  11. Associate: Associate sellers and users via a secure tunnel

  12. Avoid: Avoid breaching of user privacy

- Project problem statement feasibility assessment using NP-Hard, NP-Complete or satisfiability issues using modern algebra and or relevant mathematics.

  Project problem statement: Meta-App which acts as a supplant to all the redundant applications and is based upon intent publishing to the providers of the services using a cloud server. A solution to provide ne control to the user over the data that is shared with the service providers and implement a Pull-based approach.

  Feasibility check: The Meta-app will try to reduce or remove the dependency

of the user on many of the services apps on the users phone along with providing him all the perks of using such apps. The APIs provided by the cloud service can be implemented by any seller giving him access to a network of users interested in his service and the access tokens are expired after regular intervals to prevent spamming the user with oers. This tool will provide a person with an option to manage all his information from one platform and help him improve his privacy.

- Mathematical model: System Description:

  Let S be the solution set of the given problem statement such that, S = s, e, X, Y, f, DD, NDD, Sc, Fc such that :

  s = Start State = Users Request in the form of a Natural Language Sentence

  e = End State = Successful response for users request in the form of desired content

  X = Set of inputs = X1, X2 X1 = Set of users intent. = x11,x12 ....., x1n X2 = Events triggered by users. = x21, x22....., x2n

  Y = Set of outputs = Y1, Y2 Y1 = set of notications = Y11,Y12.....,Y1n Y2 = actions in response to an event = Y21,Y22.....Y2n

  f = Set of functions = f1, f2, f3, f4, f5,f6 where, f1 = Function to develop android interfaces to interact with user. f1(X2)= Y2 I/p : X2 O/p : Y2 f2 = Function to develop a cloud server to process the request

  f2(X1)=Y1 I/p : X1 O/p : Y1 f3 = Function to generate tokens

  f3(X1i) = Ti. I/p : X1i O/p : Ti Where Ti is Token generated for the particular intent. f4 = Function to rank the notication.

  f4(Y1i)= Ri I/p : Y1i O/p : Ri Where Ri is the rank for the notication f5 = function to use natural language processing to extract the keywords from the user request f5(X1) = Ki I/p : X1 O/p : Ki Where Ki is the set of keywords f6 = function to use android security to enhance the privacy to suit the customer needs

  I/p : Requests of the service provider for some user private data in order to

benet its own business.

O/p : Fetching the data from the provider namespace authorized by the user and sending it to service provider as if it is all that is available at the user end.

DD = Deterministic Data = Set of APIs exposed on the cloud server.

NDD = Non-Deterministic Data = Path created by tunnel , Token Management

Sc = Success Case = Intents received for the requests made by user.

Fc = Failure Case = Anything apart from the user approved data is sent to the service providers gets stored on the server.

Thus we have presented mathematical modelling in this assignment and proved that the problem under consideration is P type problem.

ANNEXURE C

LABORATORY ASSIGNMENTS ON PROJECT QUALITY AND RELIABILITY TESTING OF PROJECT DESIGN

- Design : The Meta-app will try to reduce or remove the dependency of the user on many of the services apps on the users phone along with providing him all the perks of using such apps. The apis provided bby the cloud service can be implemented by any seller giving him access to a network of users interested in his service and the access tokens are expired after regular intervals to prevent spamming the user with offers. This tool will provide a person with an option to manage all his information from one platform and help him improve his privacy.

  Design testing strategies:

- Equivalence partitioning- It is the technique that divides the input data of a software unit into partitions of data from which test cases can be derived. The equivalence partitions are usually derived from the requirements specification for input attributes that influence the processing of the test object. An input has certain ranges which are valid and other ranges which are invalid. Invalid data here does not mean that the data is incorrect; it means that this data lies outside of specific partition.

- Boundary value analysis:- The boundary between two partitions is the place where the behavior of the application changes and is not a real number itself. The boundary value is the minimum (or maximum) value that is at the boundary. Boundary value analysis does not require invalid partitions.

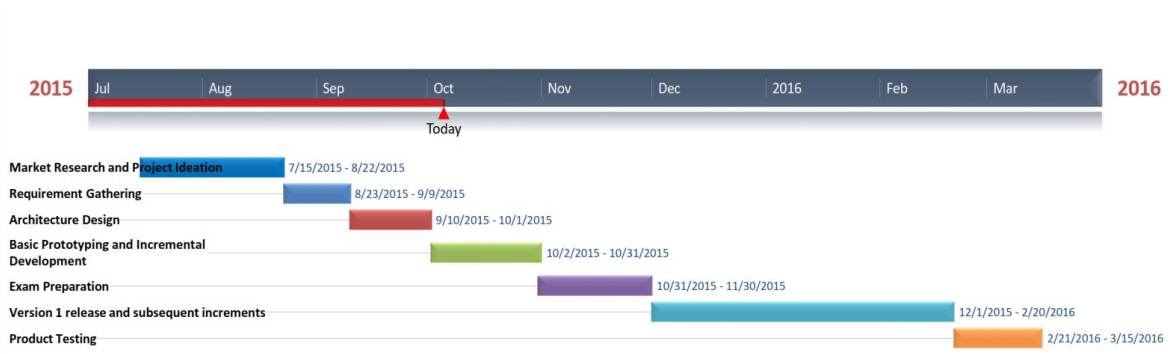
- Decision table:- Some decision tables use simple true/false values to represent the alternatives to a condition, other tables may use numbered alternatives, and some tables even use fuzzy logic or probabilistic representations for condition alternatives. In a similar way, action entries can simply represent whether an action is to be performed (check the actions to perform), or in more advanced decision tables, the sequencing of actions to perform.

- Design testing using use cases Approaches to documenting use cases that are more formal cover not only typical and exceptional work flows, but also explicit identification of the actor, the preconditions, the post conditions, the priority, the frequency of use, special requirements, assumptions, and potentially

more. The formal approach might also entail the creation of a use case diagram that shows all the actors, all the use cases, and the relationship between the actors and the use cases. Remember in decision tables we were looking for combinations of conditions that result in the wrong action occurring or the right action wasnt occurring. Here, we are looking for a situation where the system interacts improperly with the user or delivers an improper result.

ANNEXURE D

PROJECT PLANNER


2015
Jul
Aug
Sep
Oct
Nov
Dec
2016
Feb
Mar
2016
Today
Market Research and Project Ideation
7/15/2015 - 8/22/2015
Requirement Gathering
8/23/2015 - 9/9/2015
Architecture Design
9/10/2015 - 10/1/2015
Basic Prototyping and Incremental Development
10/2/2015 - 10/31/2015
Exam Preparation
10/31/2015 - 11/30/2015
Version 1 release and subsequent increments
12/1/2015 - 2/20/2016
Product Testing
2/21/2016 - 3/15/2016

ANNEXURE E

REVIEWERS COMMENTS OF PAPER SUBMITTED

1. Paper Title: Meta-App : A Pull-Based Approach

2. Name of the Conference/Journal where paper submitted : International Journal of Computer Applications (IJCA)

3. Paper accepted/rejected : Accepted

4. Review comments by reviewer :

   (a) Authors must develop original image/figures with high resolution for proper rendering of the final published paper and should not distort on zooming. The text labels in the images/figures must be clearly visible.

   (b) The research paper should be written in perspective of third person. Words such as I we our etc. needs to be avoided.

   (c) The presentation of the paper needs improvement.

5. Corrective actions if any :

   (a) The figures used in Paper were redeveloped using Dia-tool.

   (b) Formation of sentences in Paper was changed as specified in reviewer comments.

ANNEXURE F

PLAGIARISM REPORT

Plagiarism report

ANNEXURE G

TERM-II PROJECT LABORATORY ASSIGNMENTS

G.1 ASSIGNMENT 1

Review of design and necessary corrective actions taking into consideration the feedback report of Term I assessment, and other competitions/conferences participated like IIT, Central Universities, University Conferences or equivalent centers of excellence etc.

Based on the reviews received in Term I and feedback received from various judges by participating in different competitions, necessary corrective actions were taken as follows:

1. Development of a seller website for demonstartion purpose.
2. Enhancement of the Android App UX.
3. Provide a documentation of the Apis and functions in the SDK to be exposed.
4. Configuration of the server in Python-Flask
5. Stress testing of the android app and the server.

G.2 ASSIGNMENT 2

Project workstation selection, installations along with setup and installation report preparations.

- Project Workstation selection

  The Project Workstation selection was simplified due to fixed technology stack. All the project components were installed on Ubuntu 15.04 station. The different components installed were :

  1. Android Studio
  2. Flask
  3. PyZmq
  4. ConText Library

- Installation

G.3 ASSIGNMENT 3

Programming of the project functions, interfaces and GUI (if any) as per 1 st Term term-work submission using corrective actions recommended in Term-I assessment of Term-work.

Programming of the project functions was done in Python and Java. Java was used to develop End-User Android Application and Python to develop the server components and the SDK.

- Python-Flask code that runs the server:

```
from flask import Flask
from flask import render_template
from flask import request
import json
from crypto import *
from aes import AESCipher
import zmq
import time
import requests
from flask_sqlalchemy import SQLAlchemy
import uuid

server_ip = '10.42.0.1'
server_port = '5000'
proxy_ip = '10.42.0.1'
proxy_port = '5001'

# configuration
DATABASE = 'sqlite:////tmp/MetaAppServer.db'
DEBUG = True
privateKeyString = ''
zmq_port = '5556'
zmq_ip = server_ip

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = DATABASE
db = SQLAlchemy(app)
```

```
28 context = zmq.Context()
29 pubsocket = context.socket(zmq.PUB)
30
31
32 class User(db.Model):
33     id = db.Column(db.Integer, primary_key=True)
34     uuid = db.Column(db.Text, unique=True)
35     public_key = db.Column(db.Text)              #Actually the
       Aes seed
36
37     def __init__(self, uuid, public_key):
38         self.uuid = uuid
39         self.public_key = public_key
40
41     def __repr__(self):
42         return '<User %r>' % self.uuid
43
44
45 class UserReq(db.Model):
46     id = db.Column(db.Integer, primary_key=True)
47     uuid = db.Column(db.Text)
48     requestid = db.Column(db.String(80))
49     topic = db.Column(db.String(80))
50     message = db.Column(db.Text)
51
52     def __init__(self, uuid, requestid, topic, message):
53         self.uuid = uuid
54         self.requestid = requestid
55         self.topic = topic
56         self.message = message
57
58     def __repr__(self):
59         return '<User %r>' % self.uuid
60
61
62 def mogrify(topic, msg):
63     """ json encode the message and prepend the topic """
64     return topic + ' ' + json.dumps(msg)
```

```
65
66
67  def demogrify(topicmsg):
68      """ Inverse of mogrify() """
69      json0 = topicmsg.find('{')
70      topic = topicmsg[0:json0].strip()
71      msg = json.loads(topicmsg[json0:])
72      return topic, msg
73
74
75  @app.route('/', methods=['POST', 'GET'])
76  def start():
77      if request.method == 'POST':
78          return reg_user(request.form['Message'])
79      else:
80          curr = User.query.all()
81          entries = [dict(id=row.id, uuid=row.uuid, public_key=
        row.public_key) for row in curr]
82          return render_template('display.html', entries=entries
        )
83
84
85  def reg_user(message):
86      with open('private.pem', 'rb') as f:
87          privateKeyString = f.read()
88      privateKey = load_private_key_string(privateKeyString)
89      message = decrypt_message(privateKey, message)
90      message = message.replace('-', '+');
91      message = message.replace('_', '/');
92      print message
93      d = json.loads(message)
94      aes_seed = d['AesSeed']
95      Uuid = uuid.uuid1().hex
96      print Uuid
97      print aes_seed
98      user = User(Uuid, aes_seed)
99      print 'ok'
100     db.session.add(user)
```

```
101     db.session.commit()
102     print 'ok'
103     d1 = {}
104     d1['Uuid'] = Uuid
105     print d1
106     result = requests.post('http://'+proxy_ip+":"+proxy_port+'
        /adduser/', data=json.dumps(d1))
107     print result
108     return json.dumps(d1)
109
110
111 @app.route('/receive/', methods=['POST'])
112 def receive_msg():
113     with open('private.pem', 'rb') as f:
114         privateKeyString = f.read()
115     privateKey = load_private_key_string(privateKeyString)
116     emessage = request.form['Message']
117     seed = request.form['Seed']
118     seed = decrypt_message(privateKey, seed)
119     emessage = emessage.replace('-', '+')
120     emessage = emessage.replace('_', '/')
121     print seed
122     print emessage
123     obj = AESCipher(seed)
124     finalmessage = obj.decrypt(emessage)
125     print finalmessage
126
127     d = json.loads(finalmessage)
128     uuid = d['Uuid']
129     requestId = d['RequestId']
130     topic = d['Topic']
131     message = d['Message']
132
133     d1 = json.loads(message)
134     intent_description = d1['Intent_Description']
135     deadline = d1['Deadline']
136
137     print uuid
```

```
138     print requestId
139     print topic
140     print message
141     userreq = UserReq(uuid, requestId, topic, message)
142     db.session.add(userreq)
143     db.session.commit()
144     d2 = {}
145     d2['Uuid'] = uuid
146     d2['RequestId'] = requestId
147     d2['Topic'] = topic
148     d2['Deadline'] = deadline
149     result = requests.post('http://'+proxy_ip+":"+proxy_port+'
    /receive/request/', data=json.dumps(d2))
150     print result
151     publish_msg(mogrify(topic, d))
152     return "Intent Published"
153
154
155 def publish_msg(message):
156     url = 'tcp://'+zmq_ip+':'+zmq_port
157     print url
158     try:
159         pubsocket.bind(url)
160         time.sleep(1)
161         print "Sending message : "+message
162         pubsocket.send_string(message)
163     except Exception as e:
164         print "Error : "+str(e)
165     finally:
166         pubsocket.unbind(url)
167
168
169 @app.route('/session/update', methods=['POST'])
170 def session_update():
171     if request.method == 'POST':
172         message = request.form['Message']
173         with open('private.pem', 'rb') as f:
174             privateKeyString = f.read()
```

```
175         privateKey = load_private_key_string(privateKeyString)
176         message = decrypt_message(privateKey, message)
177         message = message.replace('-', '+')
178         message = message.replace('_', '/')
179         print message
180         d = json.loads(message)
181         Uuid = d['Uuid']
182         aes_seed = d['AesSeed']
183         user = User(Uuid, aes_seed)
184         db.session.add(user)
185         db.session.commit()
186         return "Session updated"
187
188
189 @app.route('/display/', methods=['GET'])
190 def display():
191     if request.method == 'GET':
192         curr = User.query.all()
193         entries = [dict(id=row.id, uuid=row.uuid, public_key=
    row.public_key) for row in curr]
194         return render_template('display.html', entries=entries
    )
195
196
197 if __name__ == '__main__':
198     app.run(host=server_ip, port=int(server_port))
```

- Python code of the SDK:

```
1 from datetime import datetime
2 import json
3 import requests
4 import zmq
5 import time
6 import threading
7 from sqlalchemy import Column, ForeignKey, Integer, String,
    DateTime
8 from sqlalchemy.ext.declarative import declarative_base
9 from sqlalchemy import create_engine
```

```
10 from sqlalchemy.orm import sessionmaker
11 from sqlalchemy.orm import scoped_session
12
13 server_ip = '10.42.0.1'
14 proxy_ip = '10.42.0.1'
15 proxy_port = '5001'
16
17 proxy_final = proxy_ip+":"+proxy_port
18 Base = declarative_base()
19
20 # Create an engine that stores data in the local directory's
21 engine = create_engine('sqlite:////tmp/MetaAppSeller.db')
22 Session_factory = sessionmaker(bind=engine)
23 Session = scoped_session(Session_factory)
24
25
26 class UserRequests(Base):
27     __tablename__ = 'user_requests'
28     uuid = Column(String(200), primary_key=True)
29     reqid = Column(String(200), primary_key=True)
30     deadline = Column(DateTime)
31     topic = Column(String(1000))
32     intentdesc = Column(String(1000))
33
34     def __init__(self, uuid, reqid, deadline, topic,
       intentdesc):
35         self.uuid = uuid
36         self.reqid = reqid
37         self.deadline = deadline
38         self.topic = topic
39         self.intentdesc = intentdesc
40
41     def __repr__(self):
42         return '<Requests %r>' % self.uuid
43
44
45 class Listener(threading.Thread):
46     zmq_port = "5556"
```

```
47      zmq_ip = "tcp://"+server_ip+":"
48      zmq_filter = []
49
50      def __init__(self, filt, ip="tcp://"+server_ip+":", port="
        5556"):
51          threading.Thread.__init__(self)
52          self.zmq_filter = filt
53          self.zmq_ip = ip
54          self.zmq_port = port
55
56      def mogrify(self, topic, msg):
57          """ json encode the message and prepend the topic """
58          return topic + ' ' + json.dumps(msg)
59
60      def demogrify(self, topicmsg):
61          """ Inverse of mogrify() """
62          json0 = topicmsg.find('{')
63          topic = topicmsg[0:json0].strip()
64          msg = json.loads(topicmsg[json0:])
65          return topic, msg
66
67      def listen(self):
68          # Socket to talk to server
69          context = zmq.Context()
70          socket = context.socket(zmq.SUB)
71          socket.connect(self.zmq_ip + self.zmq_port)
72          # Subscribe to filter Food
73          for filt in self.zmq_filter:
74              socket.setsockopt(zmq.SUBSCRIBE, filt)
75
76          while True:
77              time.sleep(1)
78              msg = socket.recv()
79              topic, messagedata = self.demogrify(msg)
80              dict_json = messagedata
81              print dict_json
82              uuid = dict_json['Uuid']
83              reqid = dict_json['RequestId']
```

```
84              message = dict_json['Message']
85              dict1_json = json.loads(message)
86              deadline_s = dict1_json['Deadline']
87              dict_deadline = json.loads(deadline_s)
88              deadline = dict_deadline['Date']
89              # Not considering time here. Add Code for it.
90              deadline = datetime.strptime(deadline, "%Y-%m-%d")
91              intent_desc = dict1_json['Intent_Description']
92              topic = dict_json['Topic']
93              session = Session()
94              new_entry = UserRequests(uuid, reqid, deadline,
    topic, intent_desc)
95              session.add(new_entry)
96              session.commit()
97              Session.remove()
98              print "Record added"
99
100     def run(self):
101         self.listen()
102
103
104 # Perform User Registration and return a seller client key to
    be used for each further requests
105 # Name and handle should be unique otherwise an error will
    occur
106 def register(name, filters, handle, endpoint):
107     cred = {}
108     cred['Name'] = name
109     cred['Filter'] = filters
110     cred['Handle'] = handle
111     cred['Endpoint'] = endpoint
112     result = requests.post('http://'+proxy_final+'/register/
    seller/', data=json.dumps(cred))
113     return result
114
115
116 # Returns an json array of user requests in json format
117 def get_requests(filters):
```

```
118     session = Session()
119     filters = filters.split(" ")
120     reqs = []
121     for topic in filters:
122         entries = session.query(UserRequests).filter(
    UserRequests.topic == topic).all()
123         if entries:
124             for i in entries:
125                 reqs.append(i)
126     Session.remove()
127     user_requests = []
128     for i in reqs:
129         req = {}
130         req['Uuid'] = i.uuid
131         req['RequestId'] = i.reqid
132         req['Topic'] = i.topic
133         req['Intent'] = i.intentdesc
134         req['Deadline'] = i.deadline.strftime("%Y-%m-%d")
135         user_requests.append(req)
136     return json.dumps(user_requests)
137
138
139 # Returns an json array of user requests in json format for a
    particular user
140 # Perform exception handling for incorrect uuid
141 def get_requests_user(uuid, filters):
142     session = Session()
143     filters = filters.split(" ")
144     reqs = []
145     for topic in filters:
146         reqs.append(session.query(UserRequests).filter(
    UserRequests.topic == topic).all())
147     Session.remove()
148     user_requests = []
149     for i in reqs:
150         req = {}
151         req['Uuid'] = i.uuid
152         req['RequestId'] = i.reqid
```

```
153         req['Topic'] = i.topic
154         req['Intent'] = i.intentdesc
155         req['Deadline'] = i.deadline.strftime("%Y-%m-%d")
156         user_requests.append(req)
157     return json.dumps(user_requests)
158
159
160 # Send a reply where the parameters are UserId, RequestId,
    Topic and Offer in a Json format
161 # The compulsory fields in offer are Service description, Cost
    and Time to complete
162 # Add response code in proxy_app for different responses
163 def make_reply(uuid, reqid, topic, offer, client_key):
164     reply = {}
165     reply['Uuid'] = uuid
166     reply['RequestId'] = reqid
167     reply['Topic'] = topic
168     reply['Offer'] = offer
169     reply['Key'] = client_key
170     result = requests.post('http://'+proxy_final+'/receive/
    offer/', data=json.dumps(reply))
171     return result
172
173
174 # Send an Access request where the parameters are File type
    and Client key
175 # The proxy will store the pending requests and then send them
    the app when fetched
176 # Json with field either Status or Error
177 def access_request(file_type, client_key):
178     req = {}
179     req['FileType'] = file_type
180     req['Key'] = client_key
181     result = requests.post('http://'+proxy_final+'/namespace/
    access/', data=json.dumps(req))
182     return result
183
184
```

```
# Send an List request where the parameters are File type and Client key
# The proxy will fetch the list of files and send back
# A json array containing json objects with fields uuid and fileid
def list_request(file_type, client_key):
    req = {}
    req['File'] = file_type
    req['Key'] = client_key
    result = requests.post('http://'+proxy_final+'/namespace/list/', data=json.dumps(req))
    return result


# Accepts a filter to configure listener and start listening
def start_listener_service(filt):
    l = Listener(filt)
    l.listen()


def init():
    Base.metadata.drop_all(engine)
    Base.metadata.create_all(engine)


if __name__ == '__main__':
    start_listener_service(["Food", "Taxi"])
```

- Java code of intent publishing:

```
package com.pict.metaappui.ui;


import android.app.DatePickerDialog;
import android.app.DatePickerDialog.OnDateSetListener;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
```

```
10 import android.support.v4.app.Fragment;
11 import android.view.View;
12 ...
13
14 /**
15  * A simple {@link Fragment} subclass.
16  */
17 public class Intent_publish extends Fragment implements
       postAsync.PostExecuteInterface {
18     String uuid;
19     String requestId;
20     String category;
21     String description;
22     String tlc;
23
24     private String expiryDate;
25     private String expiryTime;
26
27     private static final String TAG="Intent_publish";
28
29     Button publishButton;
30     EditText categoryText;
31     EditText descriptionText;
32     TextView expiryDateText;
33     TextView expiryTimeText;
34     ImageButton expiryDateButton;
35     ImageButton expiryTimeButton;
36     DatabaseHelper db;
37
38     public Intent_publish() {
39         // Required empty public constructor
40     }
41
42
43     @Override
44     public View onCreateView(LayoutInflater inflater,
       ViewGroup container,
45                              Bundle savedInstanceState) {
```

```
46          // Inflate the layout for this fragment
47          category=getArguments().getString("Category");
48          tlc=getArguments().getString("TLC");
49
50          View view=inflater.inflate(R.layout.
    fragment_intent_publish, container, false);
51
52          categoryText=(EditText)view.findViewById(R.id.
    categoryText);
53          categoryText.setText(tlc);
54          categoryText.setEnabled(false);
55
56          descriptionText=(EditText)view.findViewById(R.id.
    descriptionText);
57
58          expiryDateText = (TextView)view.findViewById(R.id.
    expiryDateText);
59          expiryTimeText = (TextView)view.findViewById(R.id.
    expiryTimeText);
60          expiryDateButton = (ImageButton)view.findViewById(R.id
    .expiryDateButton);
61          expiryTimeButton = (ImageButton)view.findViewById(R.id
    .expiryTimeButton);
62
63          expiryDateButton.setOnClickListener(new View.
    OnClickListener() {
64              @Override
65              public void onClick(View v) {
66                  pickDate();
67              }
68          });
69
70          expiryTimeButton.setOnClickListener(new View.
    OnClickListener() {
71              @Override
72              public void onClick(View v) {
73                  pickTime();
74              }
```

```
75          });
76
77          publishButton=(Button)view.findViewById(R.id.
    publishButton);
78          publishButton.setOnClickListener(new View.
    OnClickListener() {
79              @Override
80              public void onClick(View v) {
81                  try {
82                      onPublishButtonClicked();
83                  } catch (Exception e) {
84                      e.printStackTrace();
85                  }
86              }
87          });
88          return view;
89      }
90
91      public void pickDate(){
92          DatePickerFragment dateFragment=new DatePickerFragment
    ();
93          dateFragment.setOnDateSetListener(new DatePickerDialog
    .OnDateSetListener() {
94              @Override
95              public void onDateSet(DatePicker view, int year,
    int monthOfYear, int dayOfMonth) {
96                  expiryDate = new StringBuilder().append(year).
    append("-").append(monthOfYear + 1).append("-").append(
    dayOfMonth).toString();
97                  expiryDateText.setText(expiryDate);
98              }
99          });
100         dateFragment.show(getFragmentManager(),"Expiry Date");
101     }
102
103     public void pickTime(){
104         TimePickerFragment timeFragment=new TimePickerFragment
    ();
```

```
105         timeFragment.setOnTimeSetListener(new TimePickerDialog
    .OnTimeSetListener() {
106             @Override
107             public void onTimeSet(TimePicker view, int
    hourOfDay, int minute) {
108                 expiryTime = new StringBuilder().append(
    hourOfDay).append(":").append(minute).append(":").append
    (0).toString();
109                 expiryTimeText.setText(expiryTime);
110             }
111         });
112         timeFragment.show(getFragmentManager(),"Expiry Time");
113     }
114
115     public void onPublishButtonClicked() throws JSONException,
     NoSuchPaddingException,
    InvalidAlgorithmParameterException,
    NoSuchAlgorithmException, IllegalBlockSizeException,
    BadPaddingException, InvalidKeyException, IOException {
116         uuid = Preferences.getString(Preferences.PHONE_NUMBER)
    ;
117         requestId = Integer.toString(Preferences.getInteger(
    Preferences.REQUEST_ID));
118         description=descriptionText.getText().toString();
119
120         JSONObject deadline=new JSONObject();
121         deadline.accumulate("Date",expiryDate);
122         deadline.accumulate("Time",expiryTime);
123
124         JSONObject message=new JSONObject();
125         message.accumulate("Intent_Description",description);
126         message.accumulate("Deadline",deadline.toString());
127
128         JSONObject finalmessage=new JSONObject();
129         finalmessage.accumulate("Uuid",uuid);
130         finalmessage.accumulate("RequestId",requestId);
131         finalmessage.accumulate("Topic", tlc);
132         finalmessage.accumulate("Message", message.toString())
```

```
    ;
133
134         String plainText=finalmessage.toString();
135         String seed=Preferences.getString(Preferences.AES_SEED
    );
136         String aesEncrptJstr= AESnew.getInstance(seed).
    encrypt_string(plainText);
137         Log.i(TAG, "Encrpyted Message:" + aesEncrptJstr);
138         Log.i(TAG, "Length:" + aesEncrptJstr.length());
139         String rsaEncryptSeed= RSA.encryptWithKey(Preferences.
    SERVER_PUB_KEY, seed);
140         JSONObject sendJson=new JSONObject();
141         sendJson.accumulate("Message",aesEncrptJstr);
142         sendJson.accumulate("Seed", rsaEncryptSeed);
143         String sendStr=sendJson.toString();
144         Log.i(TAG, "Final Message:" + sendStr);
145         new postAsync("Publishing Intent...",getActivity(),
    this).execute("4", "Message", aesEncrptJstr, "Seed",
    rsaEncryptSeed, Preferences.url + "receive/");
146         Log.i(TAG, "Publish intent");
147
148
149     }
150
151     @Override
152     public void postExecute(int responseCode) {
153         if(responseCode==200){
154             //add to database
155             db=new DatabaseHelper(getActivity());
156             UserRequest obj=new UserRequest();
157             obj.setRequestId(Integer.parseInt(requestId));
158             obj.setTopic(tlc);
159             obj.setIntent_desc(description);
160             obj.setDeadline_date(expiryDate);
161             obj.setDeadline_time(expiryTime);
162             obj.setPending(true);
163             long id=db.createUserRequest(obj);
164             db.closeDB();
```

```
165                 Log.i(TAG,"Entry made with id "+id);
166                 Preferences.putInteger(Preferences.REQUEST_ID,
    Integer.parseInt(requestId)+1);
167             }
168         }
169 }
```

G.4 ASSIGNMENT 4

Test tool selection and testing of various test cases for the project performed and generate various testing result charts, graphs etc. including reliability testing.

1. Type of Testing Used

   The project can be considered as a combination of several modules. In case of our project, it consists of a User Application, Server, Seller Sdk and a Demonstration website. So, each such module can be considered as composed of different units which are responsible for the working of the entire module. Thus, we need to test each and every module independently as well as combine together to satisfy the user requirements.

   - Unit Testing

     Unit testing is testing the smallest testable part of a application. This is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. Unit can be entire module or a function or procedure. The tools used for testing different modules are:

     (a) User Application (Android)

         Junit Runner and Monkey

     (b) Server and Seller SDK (Python)

         Unit Tests

     (c) Seller Website (HTML5, CSS3)

         Selenium Tool

   - Integration Testing

     Integration testing corresponds to a phase in software testing in which individual software modules (which are unit tested) are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing can expose problems with the interfaces among

program components before trouble occurs in real-world program execution. Integration testing is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision. There are two major ways of carrying out an integration test, called the bottom-up method and the top-down method. We have used bottom up test methodology.

- Automated Testing

  Automated testing of the web application hosted is done by using Selenium.Selenium is a portable software testing framework for web applications. Selenium provides a record/playback tool for authoring tests without learning a test scripting language (Selenium IDE). Various functionalities were tested using this tool.

- Validation Testing

  The intents published are stored in a database. Validations used while storing the data such as ensuring unique entry for each user, each request of a user is uniquely identified, and the registered sellers along with their reputation scores are validated in this Testing phase.

- High order Testing

  High order testing was done after completely in- tegrating all the modules of the system using various test cases.

2. Test Cases and Test Results

   - Test cases in Unit Testing

     (a) User Application

     Test Case 1:

     Aim: Encryption and Decryption of User Data on Phone.

     Test data: A text to be retrieved from test file.

     Expected Result: Encrypted text must be obtained and orginal text must be obtained from the encrypted text by performing decryption.

     Actual Result: Same as expected.

Test Case 2:

Aim: File manager utility for Namespace module of App.

Test data: Select a list of files using the module.

Expected Result: The files must be selected and viewable inside the namespaces tab.

Actual Result: Files can be opened using installed apps and also perform other activities.

Test Case 3:

Aim: Session Expiry of the User App

Test data: Change the user phone date to date after session expiry.

Expected Result: App should initiate initial handshake to exchange the encryption-decryption keys.

Actual Result: New keys generated.

(b) Server and Seller SDK

Test Case 1:

Aim: Encryption and Decryption of Data on Server.

Test data: A text to be retrieved from test file.

Expected Result: Encrypted text must be obtained and orginal text must be obtained from the encrypted text by performing decryption.

Actual Result: Same as expected.

Test Case 2:

Aim: ZMQ Publish and Subscribe module for publishing intents.

Test data: Test files contain a list of text to be published along with the topics.

Expected Result: Listener for a particular topic must receive all the text with that topic.

Actual Result: Same as expected.

(c) Seller Website

Test Case 1:

Aim: Registration and Login Components.

Test data: Test files contain a list of credentials to test registration and login.

Expected Result: Duplicate entries are discarded for registration.

Actual Result: Only the successful registrations are allowed to login.

- Test cases in Integration Testing

  Manual Testing methodology was used for performing Integration testing of the various units integrated together. The various tests performed are given below:

TestCase ID	Objective	Case	Procedure	Expected Result	Actual Result	Pass/ Fail
Publish Intents	To test publishing of intents to server	1.	a) Select food category b) Enter Intent Description and Expiry Date and Time c) Press Publish	Request should be published and should be recieved by each seller subscribed to food topic.	Same as expected	Pass
		2.	a) Select food category b) Leave Intent Description field empty. c) Press Publish	Request should be discarded as the required fields are not provided.	A toast message specifying empty field will be displayed.	Pass
		3.	a) Select food category b) Expiry Date is set to invalid date. c) Press Publish	User should be restricted to select only the upcoming dates.	The calender displayed in App does not allow user to select previous dates.	Pass
Namespace Utility	To test various functions included in Namespaces.	1.	a) A seller sends an acess request. b) Select Namespace tab in App drawer.	A Notification for each such access requets from each seller should be displayed to User.	The Notifications which are not yet delivered to user are fetched and displayed.	Pass
		2.	a) Select a File from Namespace section and provide access to a first seller. b) Again provide access of another file to second seller.	First seller can only view the files which are permitted to him and not the second seller.	List requests sent from each such seller will result in a different set of files accessible.	Pass
		3.	a) Select Logs tab from the App drawer.	A list of logs recording the access requests sent by the sellers as well as files provided access to a seller.	Logs are displayed and are persistent since app installation.	Pass

Fetch Responses	To test fetching of responses on android and sending of responsed using sdk/website.	1. 2.	a) Select Offers tab from the App drawer. a) Send a response to a particular user request using the seller sdk. b) Fetch the responses by selecting the offers tab.	All the unfetched responses for user requests must be displayed. The response must be fetched and dispalyed within the offers for that particular request.	Same as expected. Same as expected.	Pass Pass
Ranking Mechanism	To test the accuracy of sentiment scores generated and evaluate the ranking mechanism	1. 2. 3.	a) Select any offer from the offers tab. b) Provide a textual user review as well as ratings. c) Click on submit. a) Send a postivite text review to the sentiment engine. a) Generate a random text and send it for evaluation to the sentiment engine.	The user review is sent to sentiment analysis engine and eventually change the rankings of the sellers. Postive score must be greater than 0.5 Positive and negative scores must be balanced as the text doesnt have a meaning.	Ranks of the sellers are changed accordingly. Postive score is greater than negative score. Positive Score approximately equal to negative score.	Pass Pass Pass

Registration and Login	To test seller registration and login.	1.	a) Select registration tab on the seller website. b) Provide the information required. c) Click on Register Button.	Unique Seller Client Key should be generated.	Client key obtained.	Pass
		2.	a) Name and/or handle provided already is used by another seller. b) Register	Registration should be unsucessfull as the constraints are not met.	Error message displayed indicating the same Name or Handle or both.	Pass
		3.	a) Enter valid username and password on login page b) Press Login	User redirected to the Homepage.	Same as expected.	Pass
		4.	a) Enter invalid username and password. b) Press Login	Error message should be displayed and user stays on login page.	Same as expected.	Pass

- Test cases in Automated Testing

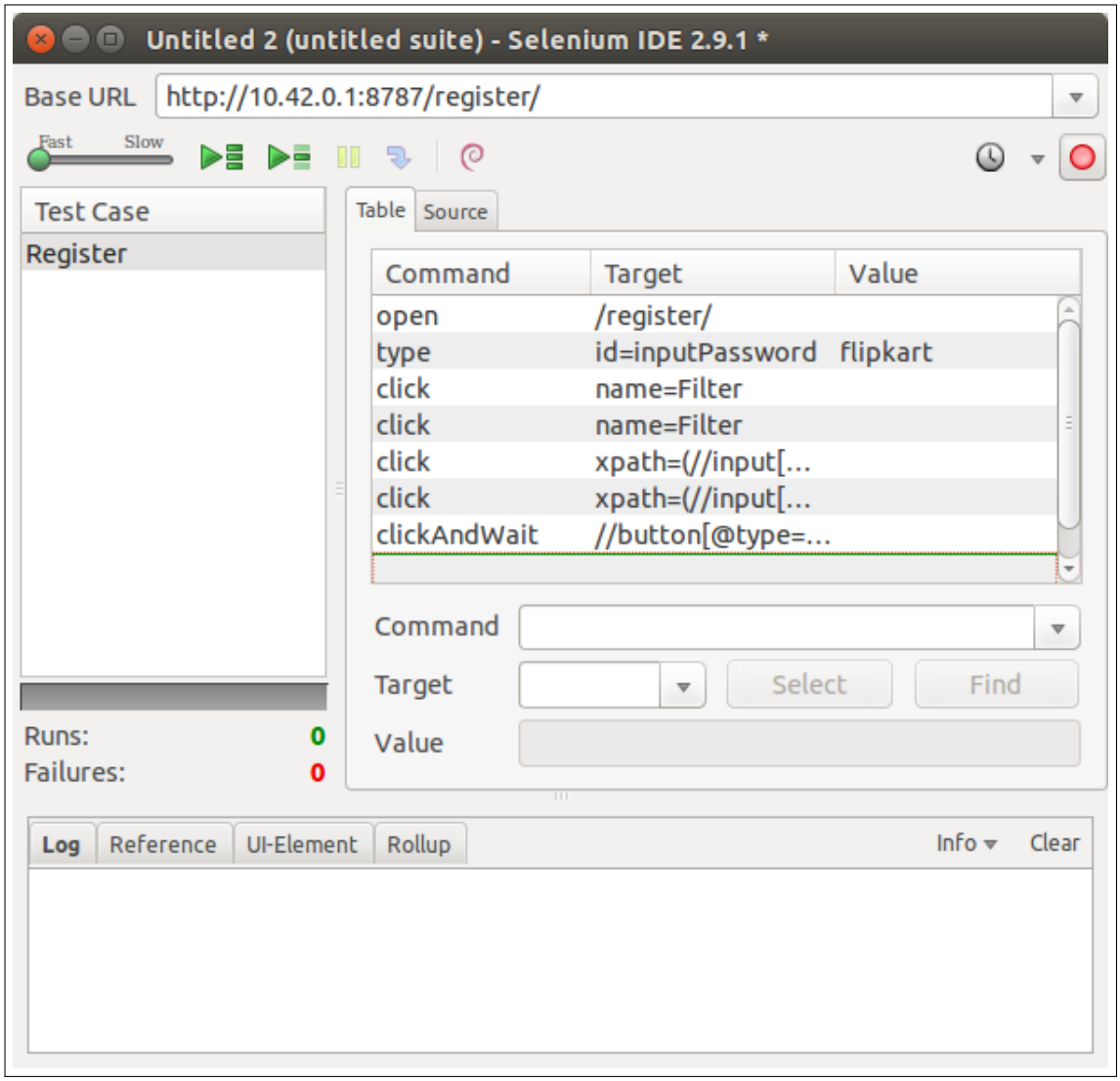


Figure G.1: Selenium Testing Tool for Registration

- Test cases in Validation Testing

  Test Case 1:

  Aim: Registration of Seller.

  Test data: A list of values to be used for registration.

  Expected Result: Unique Name and Handle should be provided. If not then Registration fails otherwise it is successfull and a seller client key is generated.

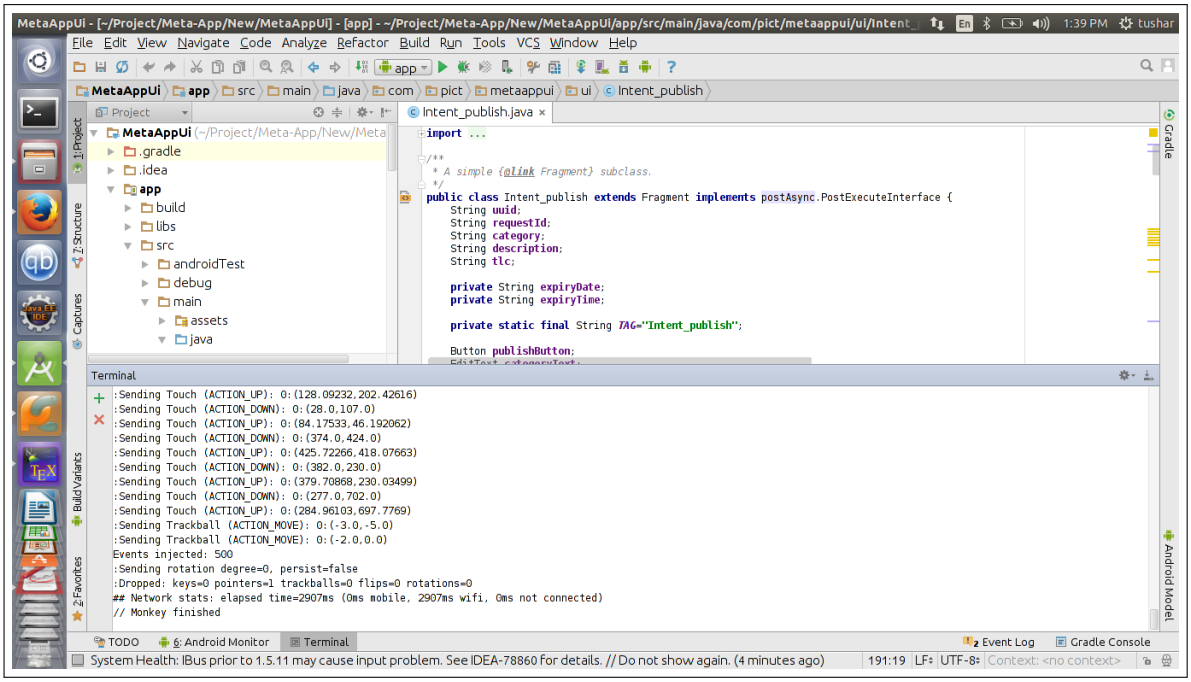


Figure G.2: Monkey Testing Tool for Android App

Actual Result: Same as expected.

Test Case 2:

Aim: Validation of sellers.

Test data: Any function from the SDK is executed.

Expected Result: The client key provided by the seller should be validated and accordingly the response is sent.

Actual Result: Service is denied if the client key is incorrect.

- Test cases in High order Testing

  Test Case 1:

  Aim: Complete end to end testing.

  Test data: Any function from the App/Seller Website is executed.

  Expected Result: The intents from a user are published to athe respective sellers and the sellers are able to respond to the intents with their offers. The users can provide a review to rank the sellers.

  Actual Result: Same as expected.

ANNEXURE H

INFORMATION OF PROJECT GROUP MEMBERS

1. Name : Tushar Jagdish Badgu
2. Date of Birth : 30/08/1994
3. Gender : Male
4. Permanent Address : A-304, Mansarovar Annexe, Sukhsagar Nagar, Pune-411048
5. E-Mail : tushar.badgu@gmail.com
6. Mobile/Contact No. : 8446992752
7. Placement Details : Morgan Stanley
8. Paper Published : Meta-App : A Pull-Based Approach

1. Name : Abhidnya Amod Patil
2. Date of Birth : 13/11/1994
3. Gender : Female
4. Permanent Address : A2-601, Regency Cosmos, Baner-Mhalunge Road, Near orchid School, Pune-411045
5. E-Mail : patil.abhidnya@gmail.com
6. Mobile/Contact No. : 8446992752
7. Placement Details : SAP
8. Paper Published : Meta-App : A Pull-Based Approach

1. Name : Snehal Shrirang Rasakar
2. Date of Birth : 05/03/1995
3. Gender : Female
4. Permanent Address : A-401, Castalia, Near Swaraj Garden, Pimple Saudagar, Pune-411027
5. E-Mail : snehalraskar5@gmail.com
6. Mobile/Contact No. : 8149097787
7. Placement Details : Persistent
8. Paper Published : Meta-App : A Pull-Based Approach

1. Name : Omkar Patil
2. Date of Birth : 11/12/1994
3. Gender : Male
4. Permanent Address : A-1, Aalap apartment, 391E-ward, Shahupuri, Kolhapur.
5. E-Mail :omkarpatilkop@gmail.com
6. Mobile/Contact No. : 7507519643
7. Placement Details : ACI Worldwide
8. Paper Published : Meta-App : A Pull-Based Approach