Android Application for SMS Spam Detection in Roman Urdu

Abstract:

In today's world where technology is rapidly changing, we have many applications in the smartphone world that can detect spam. These applications use many heuristics to determine whether a message is spam, along with the fact that they only use messages encountered during the training phase. What separates our application is the fact that it has a knowledge base containing all words in the Roman Urdu domain along with its multiple representations. It uses the knowledge base to check whether the word we are using to train the filter has any other representation.

Contents

Project Proposal

Project Title:

“Android Application for SMS Spam Detection in Roman Urdu”

Project Scope:

The project’s scope includes implementing a system and a mechanism that ensures that any unwanted SMS received by the user is transferred to a spam folder. By implementing this application it will definitely save the user’s personal time which is wasted reading through these unwanted messages. These unwanted messages can come from different sources like promotional companies who want to advertise their new products, messages from different mobile networks and people sending forward messages every day. The application will keep to keep the interaction levels between the user and the application at a minimum otherwise the objective of this application will be futile. Another important aspect of our project is that our application is capable of filtering messages that come in Roman Urdu, this feature is not being provided currently by applications available on Android and thus gives our application a distinguishing edge. At the initial stage, the user will be asked to provide sample messages that he categorizes as spam and based on those messages our application will learn and categorize other incoming messages as spam or not.

Project Objectives:

The aim of this project is to provide a viable solution that is not offered by applications currently available.

- The primary objective of this project is to provide a filtering mechanism that will ensure that spam SMS are moved to a spam folder without involving the user in the process.
- Another objective of this project is to make sure that our application runs silently in the background and be fault tolerant.
- The application also want to ensure that our user’s critical time is saved which can only be done by keeping the interaction levels between the user and the application as minimum as possible.

Problem Statement and Description:

- The currently available applications in the spam detection category use keywords to categorize messages as spam, which is a pretty naïve approach plus they only are available for most common languages like American English, British English, Chinese and etc. Our application will be the first of its kind and there is an urgent need for an application that caters for an Urdu speaking community like Pakistan. In Pakistan, every day people receive countless messages which are of no interest to us which in turn leaves us frustrated when people take out our smartphones to check the message.
- In today’s fast paced world there is a strong demand for an application that solves this kind of problem and our application will do that exactly.

Motivation:

There are many spam applications available in the market for most common languages like American English, British English, Chinese and etc. There are no applications currently in the market for Roman Urdu. Our application will be the first of its kind to implement and detect spam in roman Urdu. As our application is the first of its kind so there is a lot of research involved. The challenge applocation will face in this place is that there are multiple representations of the same word in Roman Urdu. Hence this project is both a challenge and huge motivation for us.

SRS

List of Features:

- F-01: Intercept all incoming messages and inspect them based on an algorithm.
- F-02: Easy to Install, no complex procedure (just have to accept permissions).
- F-03: Meet the needs of the user.
- F-04: Classify messages as spam with hundred percent confidence.

Functional Requirements:

User Services

- R-01: System will satisfy the needs of the user by classifying the messages as stated by him after installing the application.
- R-02: Recover from any system failure without notifying the user.
- R-03: Only ask for the user's attention if the system is not able to recover from failure.
- R-04: System will silently run in the background without prompting the user.
- R-05: System will only ask the user about the classification of a message as spam or not when it is unsure.

Miscellaneous Services:

- R-06: System will generate reports on how many messages were classified as spam and with how much confidence they were classified.
- R-07: System will keep a backlog of all messages that were moved to the spam folder, if the user ever wishes to go through those messages; otherwise, he/she can delete them in batch mode.

Non-Functional Requirements:

- NFR-01: User will be provided with a settings menu if he/she wishes to customize the application's behavior.
- NFR-02: User will be provided with a simple interface from where he can view the spam messages.
- NFR-03: An option will also be given to user if he/she pleases to change the readability mode of the application.

Software Quality Attributes

Security:

The system needs permission to access user's SMS messages and it is the system's responsibility to ensure that messages are not used for any heinous activity.

Hardware interface:

Client Side:

Operating System: Android version 2.3 and up.

Processor: 800 MHz minimum

RAM: 256 Mb or more

Space: 5 Mb

User interface:

Native resolution of android device.

Usability:

The interface and the software usage is made user friendly and interaction between application and user will be kept minimum.

2 Column use case

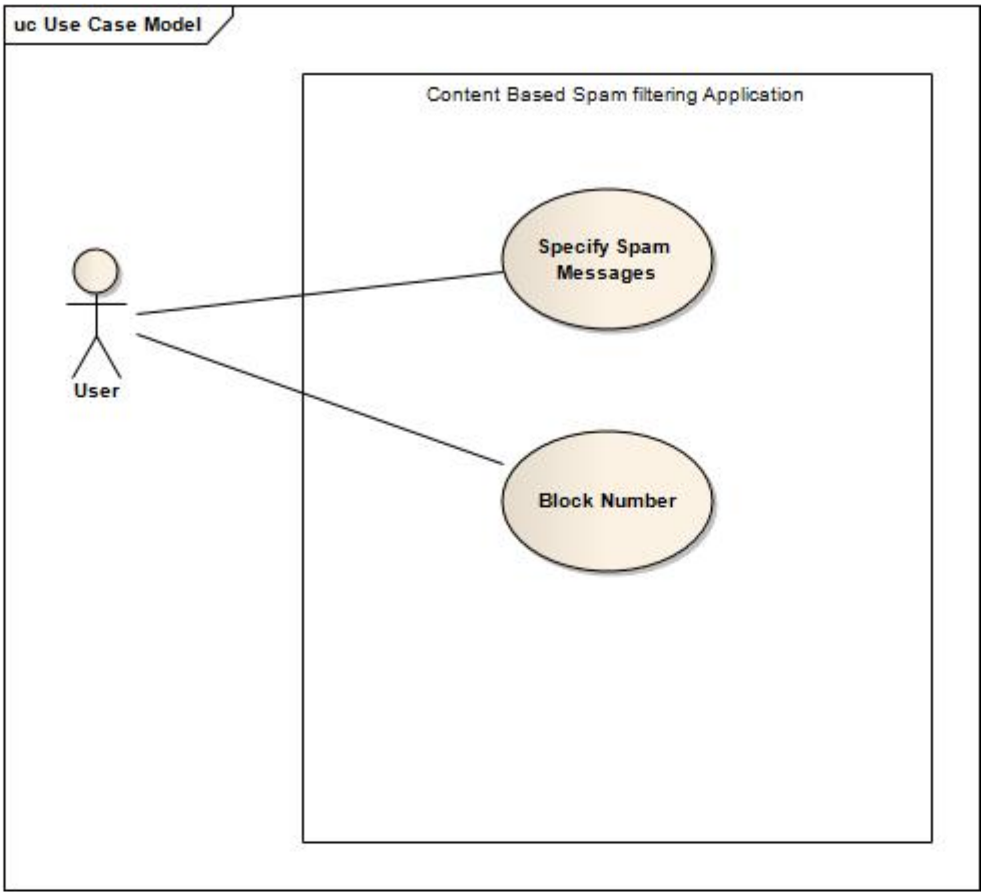
1. Specify Spam

Preconditions: N.A.

Success Guarantee (or Post conditions): Spam messages and words/phrases will be categorized as spam and put in the spam repository.

Actor	System
1. User navigates to the Incoming SMS section.	
	2. System displays all Incoming SMS
3. User selects the message considered as spam.	
	4. System moves the message to spam section.
	5. System extracts the word/phrases from the specified spam message and add them to spam words repository.
User repeats steps 3-5 until indicates done.	

Use Case Diagram



Full dressed Use Case

Use Case UC1: Specify spam

Scope: SMS Spam Detection in Roman Urdu.

Level: User Goal.

Primary Actor: User

Stakeholders and Interests:

-**User:** wants the spam to be correctly specified as spam.

Preconditions: N.A.

Success Guarantee (or Post conditions): Spam messages and words/phrases will be categorized as spam and put in the spam repository.

Main Success Scenario (or basic flow):

1. User navigates to the Incoming SMS section.

2. System displays all Incoming SMS

3. User selects the message considered as spam.

4. System moves the message to spam section.

5. System extracts the word/phrases from the specified spam message and add them to spam words repository.

User repeats steps 3-5 until indicates done.

Extensions

***a: At any time, System fails:**

1. The system displays the error message and restarts

1a. User wants to enter words/phrases to categorize as spam.

1. User enters the words/phrases to categorize as spam.
2. System responds by putting the words/phrases in the spam words repository.

***b: User performs an invalid operation:**

1. System prompts the user if the system state is affected.

3a. User categorizes a message as spam, unintentionally:

1. User navigates to the spam folder.

2. System displays the spammed messages

3. User selects message and request removal.

4. System responds by removing the message from spam folder.

User repeats steps 3-4 until indicates done.

4b. User categorizes a word/phrases as spam, unintentionally:

1. User navigates to the spam words/phrases folder.
2. User selects the word/phrase and request removal.
3. System removes the Word/phrase from the spam folder.

User repeats steps 2-3 until indicates done.

Use Case UC2: Block Number

Scope: SMS Spam Detection in Roman Urdu.

Level: User Goal.

Primary Actor: User

Stakeholders and Interests:

-**User:** wants the spam words/phrases to be correctly specified as spam.

Preconditions: N.A.

Success Guarantee (or Post conditions):

- Future SMS from the specified number will not be shown.

Main Success Scenario (or basic flow):

1. User navigates to the specify block number option.

2. System asks the user to enter the number to block.

3. User enters the number, he wishes to be blocked.

4. System saves the number and blocks all future incoming messages from that number.

Extensions

***a: At any time, System fails:**

1. The system displays the error message and restarts

***b: User performs an invalid operation:**

1. System prompts the user if the system state is affected.

3a. User blocks a number, unintentionally:

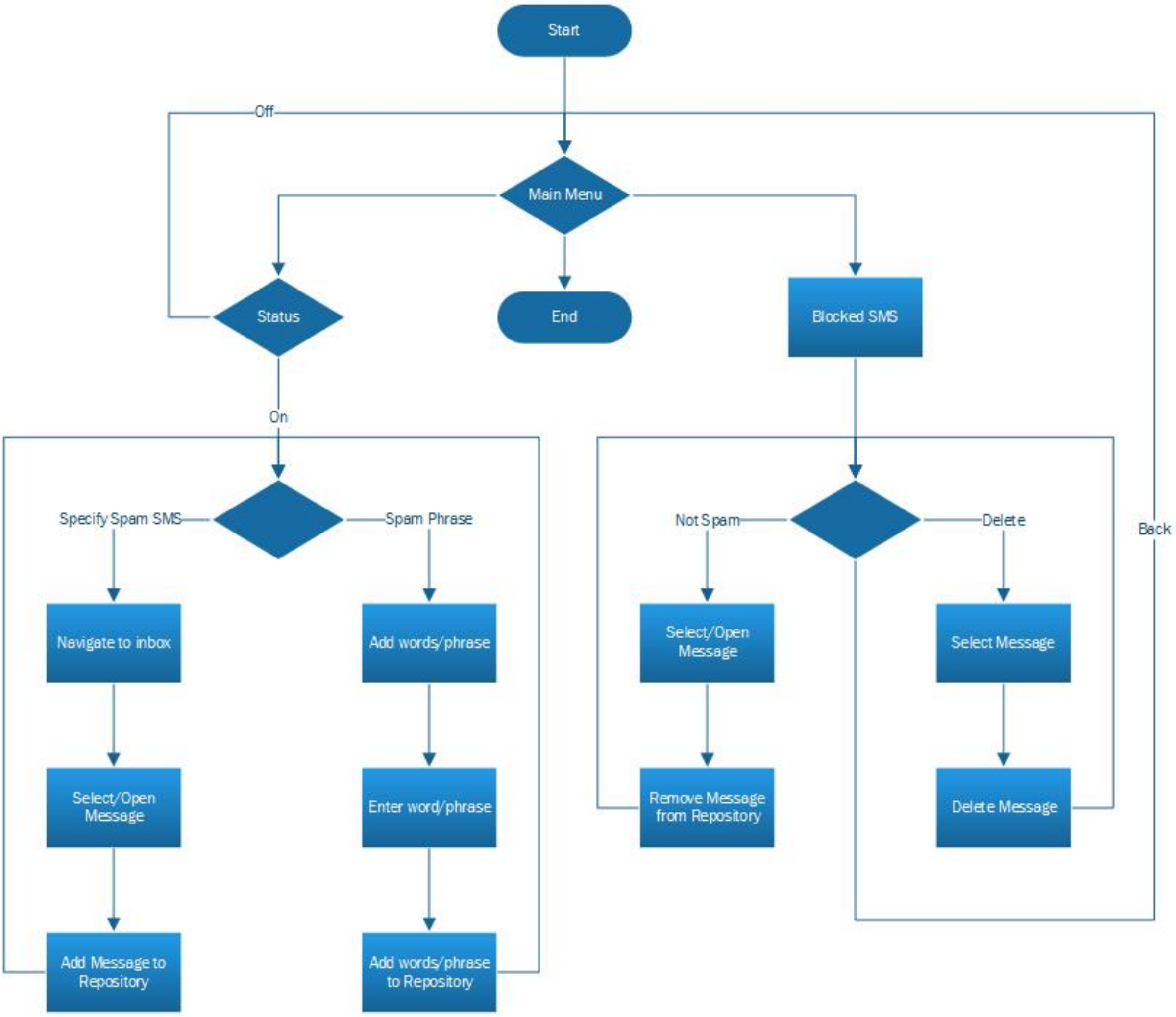
1. User navigates to blocked numbers page.

2. System displays all the blocked numbers.

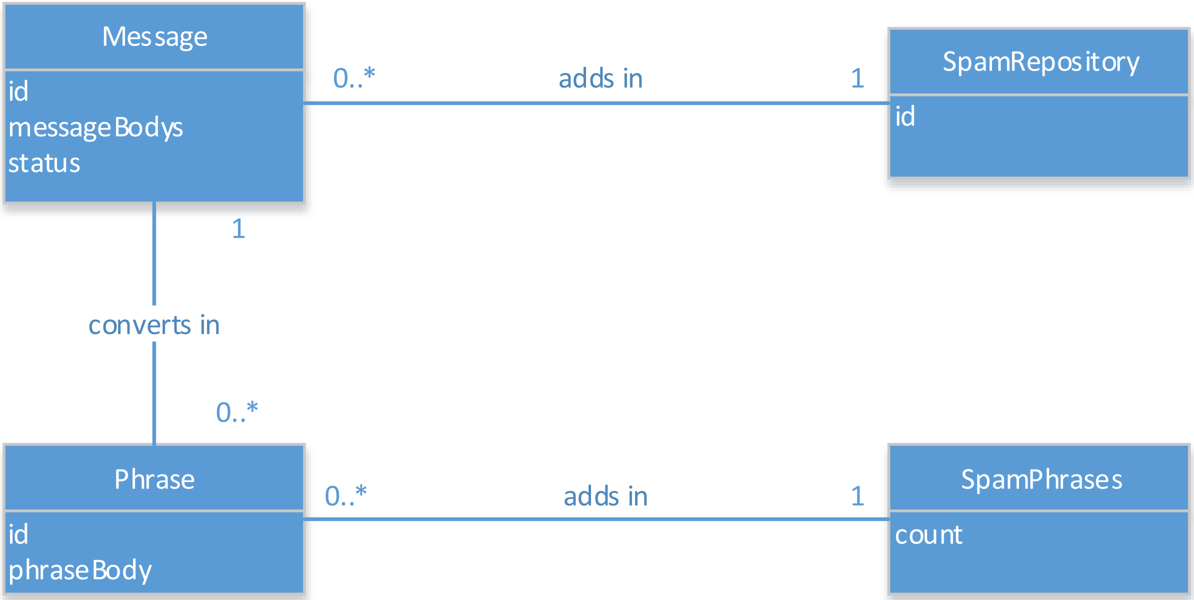
3. User selects the number and requests removal.

4. System responds by removing the number from the blocked list.

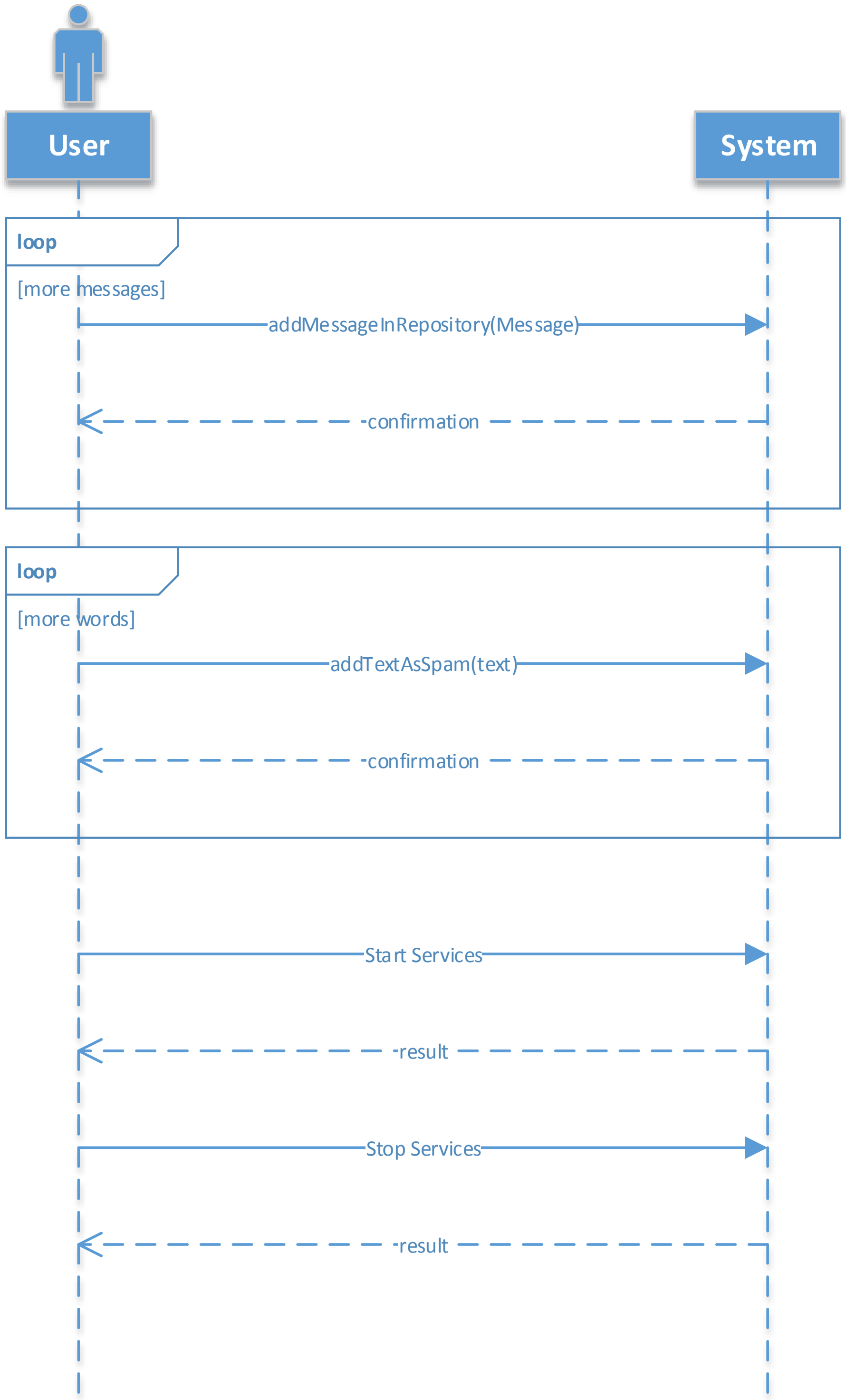
Activity Diagram



Domain Model



System Sequence Diagram



Operation Contracts

Use Case UC1: Specify spam

Contract C01: AddMessage

Operation: AddMessage(string:message)

Cross Reference: Use Case: Specify Spam

Preconditions: Message has been received.

Post conditions:

- A Message Collection mc instance was created.
- mc was associated with the specified spam repository.
- A Spam Phrases sp instance was created.
- sp was associated with the specified spam repository.
- mc.message became message.
- sp.message became message.

Contract C02: AddPhrasesFromMessage

Operation: AddPhrasesFromMessage(string:message)

Cross Reference: Use Case: Specify Spam

Preconditions: Message has been received.

Post conditions:

- Message m instance was created.
- m was associated with SpamPhrases.
- Phrase Collection pc and phrase p instances were associated with SpamPhrases.

Contract C03: AddPhrase

Operation: AddPhrase(string:text)

Preconditions: Message has been received.

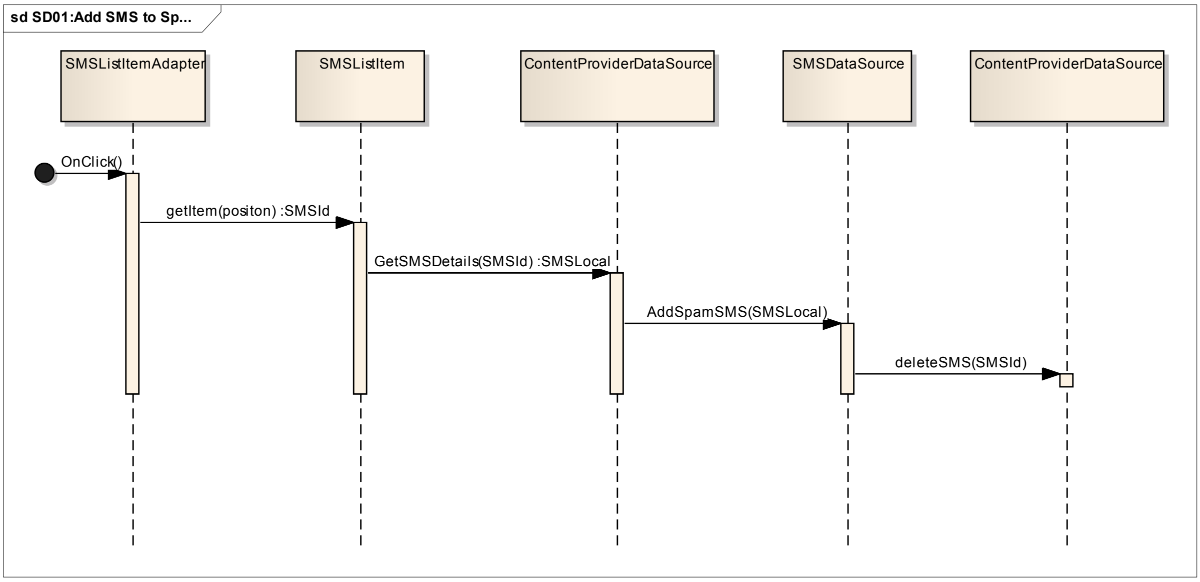
Post conditions:

- Phrase p instance was created.
- p was associated with SpamPhrases.
- p.text was modified to text.
- PhraseCollection pc instance was created.
- pc was associated with SpamPhrases
- pc.text was modified to text.

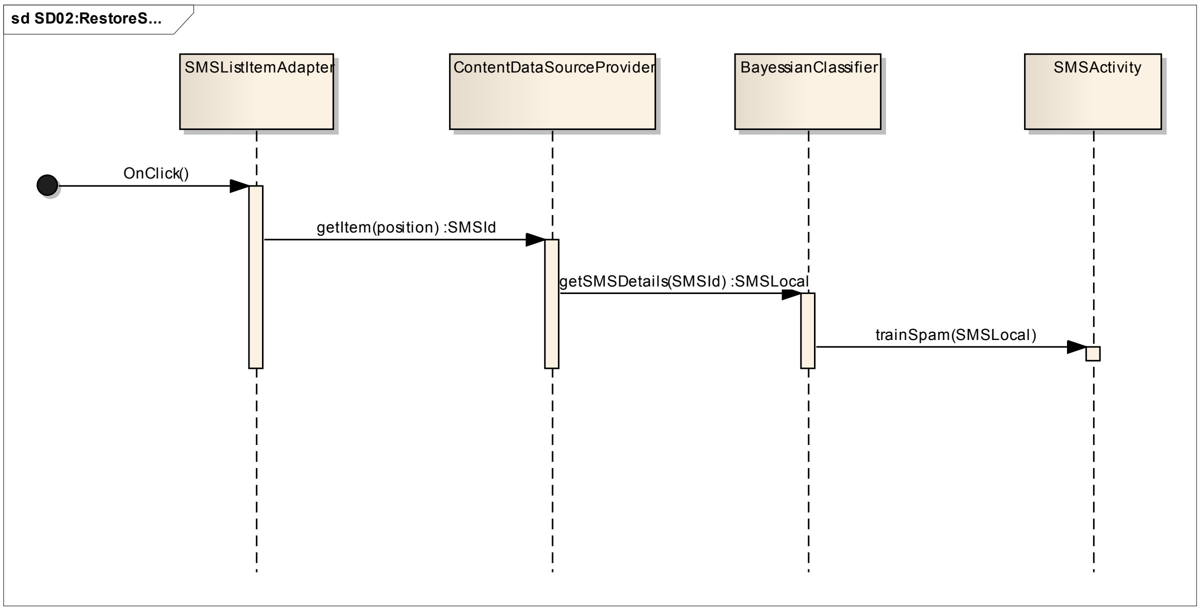
Sequence Diagrams

Use Case UC1: Specify spam

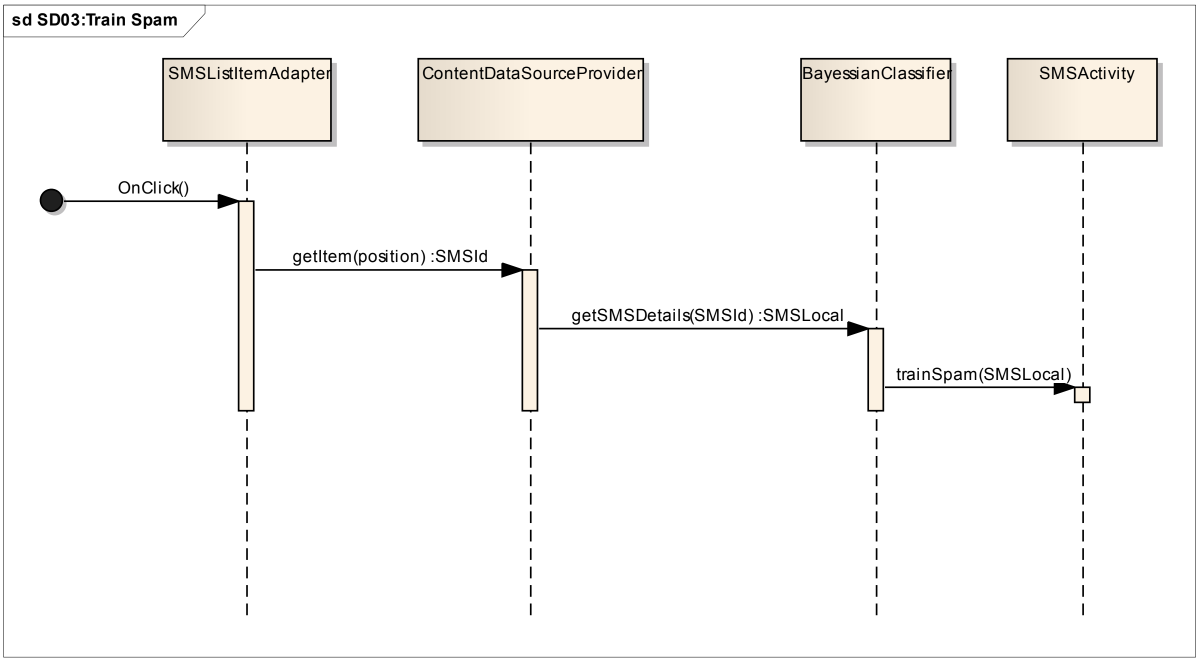
SD01: Add SMS to Spam



SD02: Restore SMS

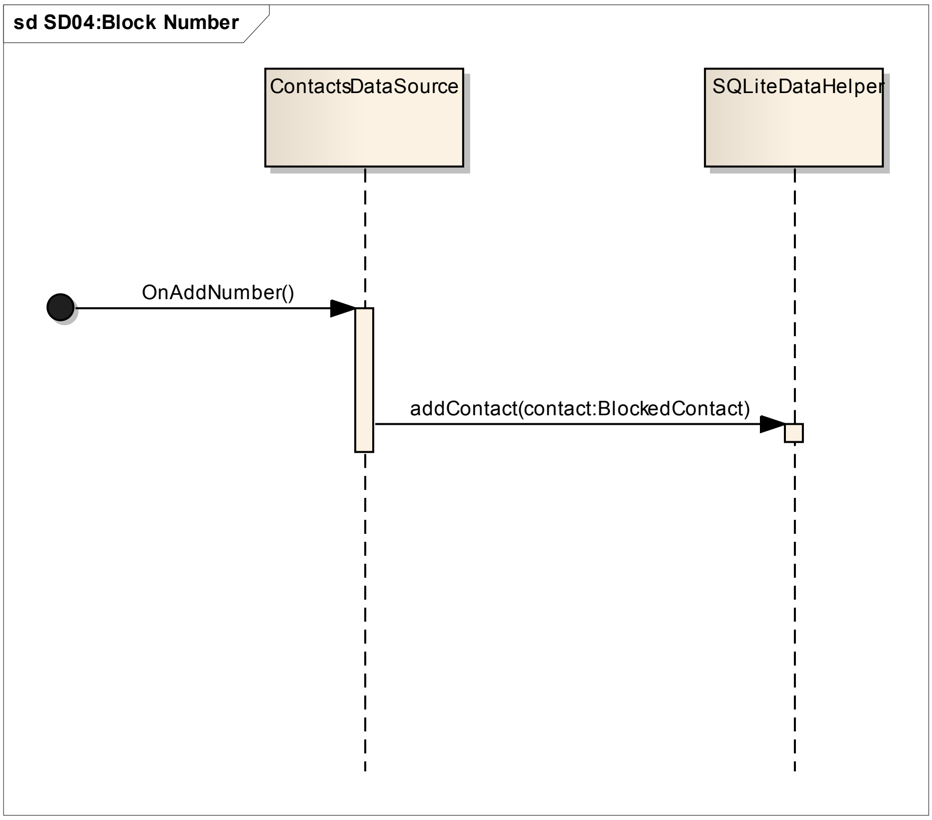


SD03: Train Spam

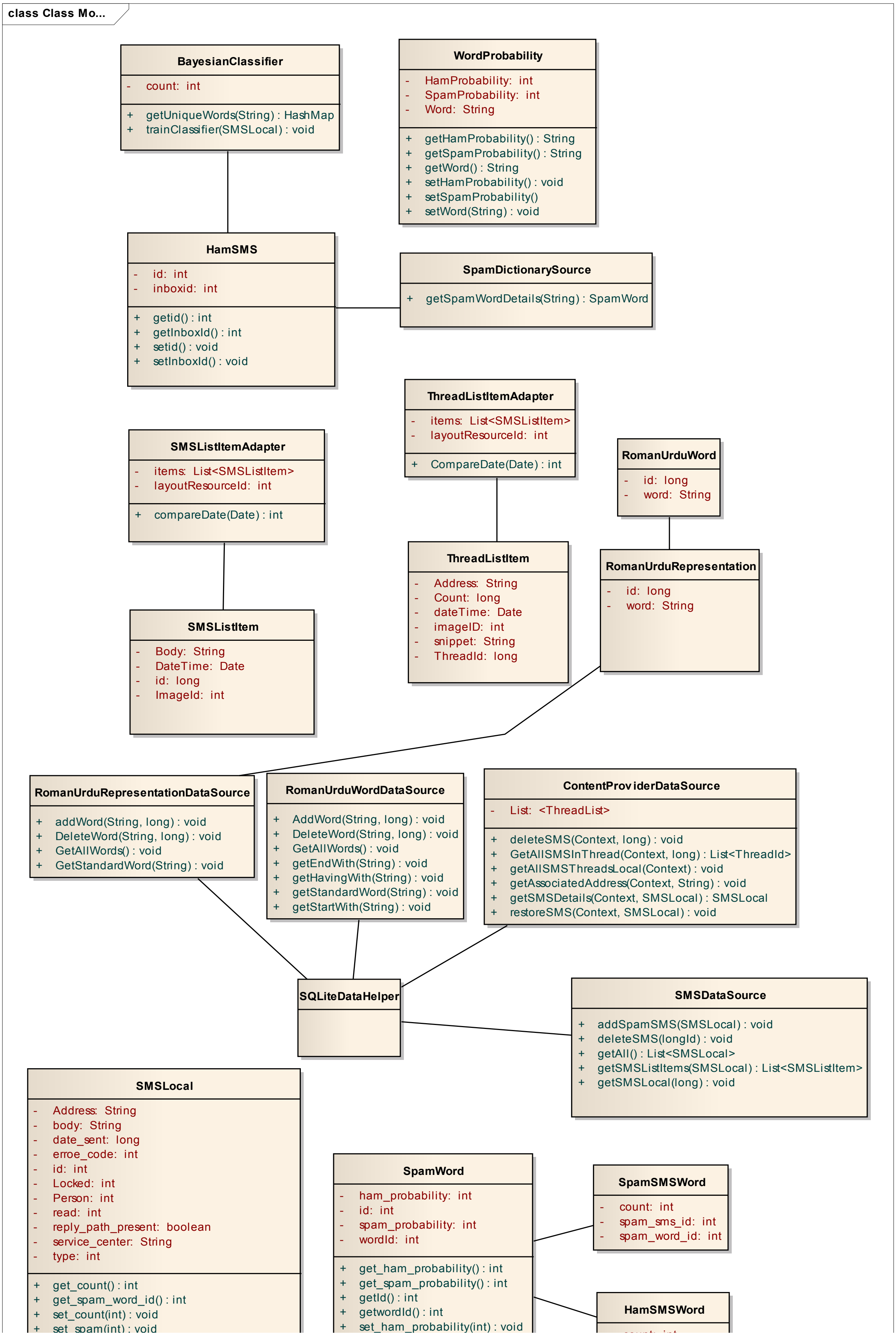


Use Case UC2: Block Number

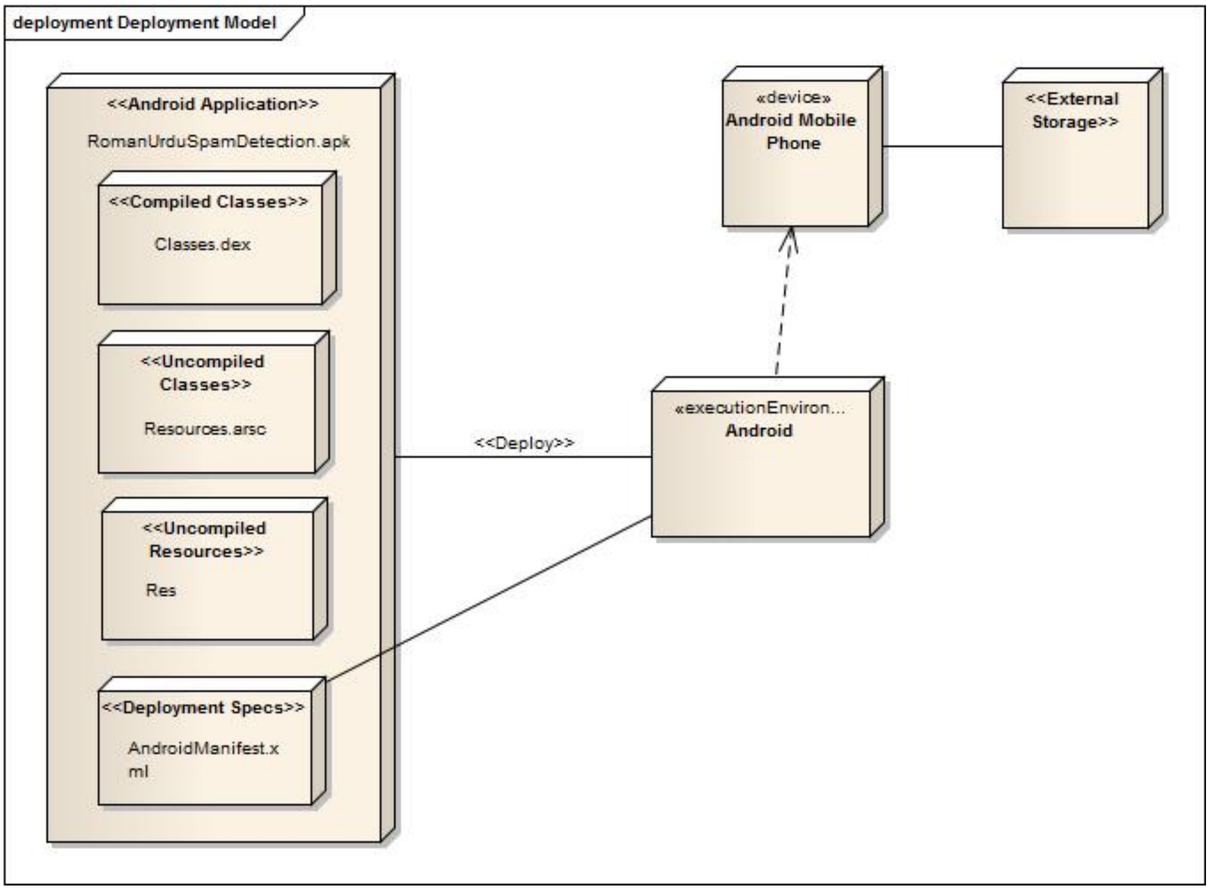
SD01: Train Spam



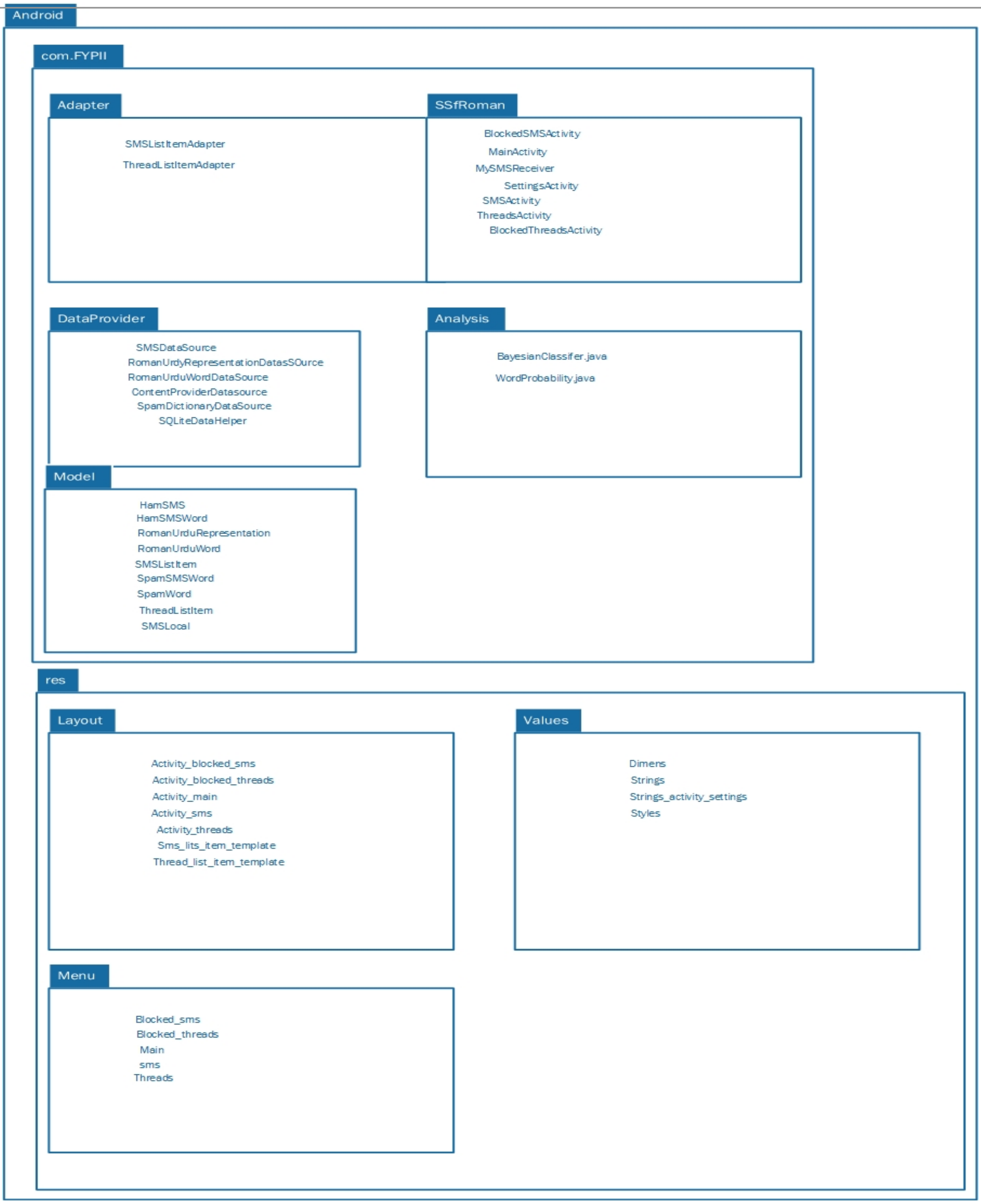
Class Diagram


class Class Mo...
BayesianClassifier
- count: int
+ getUniqueWords(String) : HashMap
+ trainClassifier(SMSLocal) : void
WordProbability
- HamProbability: int
- SpamProbability: int
- Word: String
+ getHamProbability() : String
+ getSpamProbability() : String
+ getWord() : String
+ setHamProbability() : void
+ setSpamProbability()
+ setWord(String) : void
HamSMS
- id: int
- inboxid: int
+ getid() : int
+ getInboxId() : int
+ setid() : void
+ setInboxId() : void
SpamDictionarySource
+ getSpamWordDetails(String) : SpamWord
ThreadListItemAdapter
- items: List<SMSListItem>
- layoutResourceId: int
+ CompareDate(Date) : int
SMSListItemAdapter
- items: List<SMSListItem>
- layoutResourceId: int
+ compareDate(Date) : int
RomanUrduWord
- id: long
- word: String
ThreadListItem
- Address: String
- Count: long
- dateTime: Date
- imageID: int
- snippet: String
- ThreadId: long
RomanUrduRepresentation
- id: long
- word: String
SMSListItem
- Body: String
- DateTime: Date
- id: long
- ImageId: int
RomanUrduRepresentationDataSource
+ addWord(String, long) : void
+ DeleteWord(String, long) : void
+ GetAllWords() : void
+ GetStandardWord(String) : void
RomanUrduWordDataSource
+ AddWord(String, long) : void
+ DeleteWord(String, long) : void
+ GetAllWords() : void
+ getEndWith(String) : void
+ getHavingWith(String) : void
+ getStandardWord(String) : void
+ getStartWith(String) : void
ContentProviderDataSource
- List: <ThreadList>
+ deleteSMS(Context, long) : void
+ GetAllSMSInThread(Context, long) : List<ThreadId>
+ getAllSMSThreadsLocal(Context) : void
+ getAssociatedAddress(Context, String) : void
+ getSMSDetails(Context, SMSLocal) : SMSLocal
+ restoreSMS(Context, SMSLocal) : void
SQLiteDataHelper
SMSDataSource
+ addSpamSMS(SMSLocal) : void
+ deleteSMS(longId) : void
+ getAll() : List<SMSLocal>
+ getSMSListItems(SMSLocal) : List<SMSListItem>
+ getSMSLocal(long) : void
SMSLocal
- Address: String
- body: String
- date_sent: long
- erroe_code: int
- id: int
- Locked: int
- Person: int
- read: int
- reply_path_present: boolean
- service_center: String
- type: int
+ get_count() : int
+ get_spam_word_id() : int
+ set_count(int) : void
+ set_spam(int) : void
SpamWord
- ham_probability: int
- id: int
- spam_probability: int
- wordId: int
+ get_ham_probability() : int
+ get_spam_probability() : int
+ getId() : int
+ getwordId() : int
+ set_ham_probability(int) : void
SpamSMSWord
- count: int
- spam_sms_id: int
- spam_word_id: int
HamSMSWord

Deployment Diagram



Package Diagram


Android
com.FYPII
Adapter
SMSListItemAdapter
ThreadListItemAdapter
SSfRoman
BlockedSMSActivity
MainActivity
MySMSReceiver
SettingsActivity
SMSActivity
ThreadsActivity
BlockedThreadsActivity
DataProvider
SMSDataSource
RomanUrdyRepresentationDatasSOurce
RomanUrduWordDataSource
ContentProviderDatasource
SpamDictionaryDataSource
SQLiteDataHelper
Analysis
BayesianClassifer.java
WordProbability.java
Model
HamSMS
HamSMSWord
RomanUrduRepresentation
RomanUrduWord
SMSListItem
SpamSMSWord
SpamWord
ThreadListItem
SMSLocal
res
Layout
Activity_blocked_sms
Activity_blocked_threads
Activity_main
Activity_sms
Activity_threads
Sms_lits_item_template
Thread_list_item_template
Values
Dimens
Strings
Strings_activity_settings
Styles
Menu
Blocked_sms
Blocked_threads
Main
sms
Threads