Software Requirements Specification
NotifyMe

Group 10
Deepanker Mishra - 2013CS50282,
Garvit Jain - 2013CS50284,
Shivanshu Gupta - 2013CS50298

March 15, 2017

1. Introduction

1.1 Purpose

The purpose of this document is to present an innovative android application, NotifyMe, which takes the burden of sifting through all notifications - like discounts, cashbacks, etc. - away from the user. This document will explain the features and interfaces of the app, what the app does and the constraints under which it must operate. It will provide the software requirement specification for the application NotifyMe.

This project is developed as a course project but is useful to anyone who uses a smartphone these days.

1.2 Scope

This project will consist of developing an Android application which will have permissions to read incoming SMS/notifications. It will then employ text processing to extract various characteristics related to offer such as:

- source - which application did it originate from?
- category - what the offer is for? eg. Food, movies etc.
- discount - how much discount/cashback if any is being offered?
- validity - for how long is the offer valid?
- code - coupon code if any?
- contact info if any?
- and more.

The information is stored in a database and periodically updated as new offers come and old ones expire. This information will be used to let the user while in-app, view the best and most recent offers from any category.

1.3 Definitions, acronyms and abbreviations

- **User:** Someone who interacts with the mobile phone application
- **App:** is abbreviation for Application and may occur in this document at many places.
- **SMS:** Short Message Service, which is the medium most companies use these days to update their customers on recent coupons, discounts and offers.
- **DB:** stands for Database of the application.
- **Promo:** is any promotional offer received via SMS or as a notification from an e-commerce application.

- **Vendor:** is the source of the promotional offer. In most cases it will be an e-commerce application.
- **ERD:** stands for Enterprise Relationship Diagram.

1.4 References

- IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.
- https://docs.moltin.com/promotions/
- https://www.diffbot.com/dev/docs/product/

1.5 Overview

The next section, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third section, i.e. the Specific Requirements section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

2. Overall Description

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe how the stakeholders interact with the system. At last, the constraints and assumptions for the system will be presented.

2.1 Product Perspective

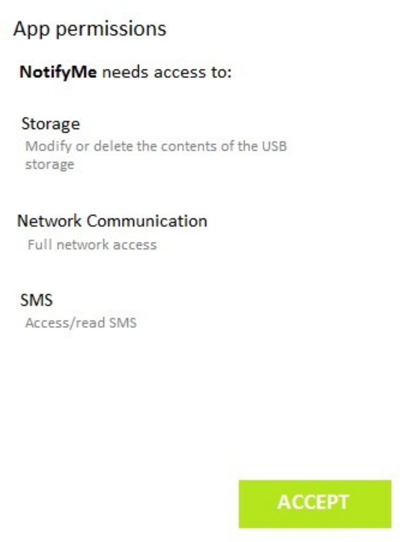
This product is independent and totally self-contained, i.e., this product is not a component of any other large project. This project in itself is complete. The project consists of developing an Android app, whose various components will be described in the following sections.

2.1.1 System Interface

The application uses text processing and pattern recognition to classify messages according to their source of origin, purpose (offers, personal, etc.), type of offer, etc and then score them to generate a sorted list of offers depending on their usability. Using this sorted list the app tries to make suggestions to the user as and when he/she opens the application - our app (NotifyMe) or the app for which he/she wants the offers. If a user opens NotifyMe app then he/she can see the sorted list of offers generated using our scoring system else if the user opens the application that he wants best offers for, then we list them in the notifications tab.

2.1.2 User Interface

User provides our app, NotifyMe, permissions to access/read SMS when he/she downloads the app or can be later changed from Settings>Apps>NotifyMe>Permissions. Without this permission the app will not work.



App permissions

NotifyMe needs access to:

Storage
Modify or delete the contents of the USB storage

Network Communication
Full network access

SMS
Access/read SMS

Figure 1: Permissions UI

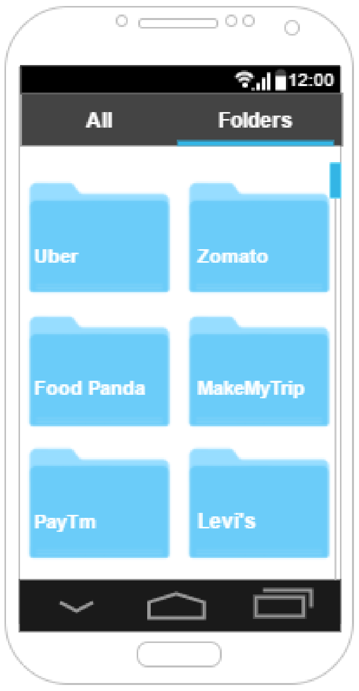


Figure 2: Tab with folders for different apps

Once this is done, the app can then processes the information in each SMS and classify them as explained earlier. The user also needs to state the apps (eg. Uber, Dominos, etc.) for which offers have to be sorted, otherwise our application would not know what are the sources of origin that are of actual interest to the user. Any person using the app for the first time will have to do the aforementioned to get the app working.

When the user opens NotifyMe app he/she will see this messages organized into folders depending on the source of origin. The user can then open any folder of his/her choice and filter the promos as per his/her requirements. This is shown in figure 2.

2.1.3 Hardware Interface

The application will be a mobile app that will run on all Android phones using Android v5.0 or above. Since the mobile app does not have any otehr designated hardware, it does not have any direct hardware interfaces.

2.1.4 Software Interface

Table 1 shows the various software interfaces required.

Software Used	Description
Operating System	The app should run on all smart-phones running the Android OS. Minimum version: Lollipop v5.0 - API level 21.
Text Processing	The app will use TextRazor's Java api to extract phrases from notification texts. These will then be used to extract attributes of the offers.
Database	The app will use the Sqlite database provided by Android for storing offers and their attributes.

Table 1: Table for Software Interfaces

2.1.5 Communication Interface

The communication between the app and the database and between the app and the TextRazor servers is important. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying Android operating systems. The communication between app and the TextRazor servers will in particular be using HTTP.

2.1.6 Memory Constraints

As the Android OS puts a cap of 24MB memory per-app, the application should not exceed this limit. This however should not be a problem as much of the text processing will be off-loaded to the TextRazor servers.

2.2 Product Functions

2.2.1 Context Diagram

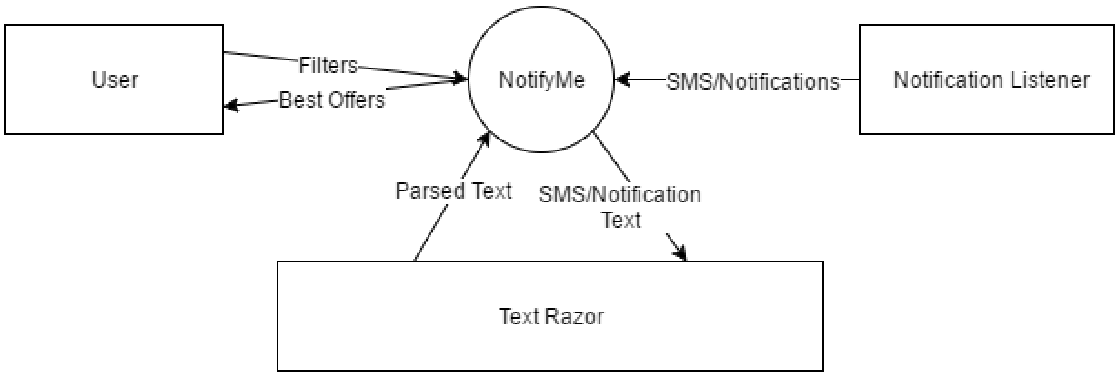


Figure 3: Context Diagram

As shown in figure 3 the outer environment to our application includes Text Razor, Notification Listener and the User of our application. Text Razor is an on-line NLP package that helps our application with text processing. Notification Listener is an Android service that handles notifications and we use it to get the notifications received by the user.

2.2.2 Use Case Diagram

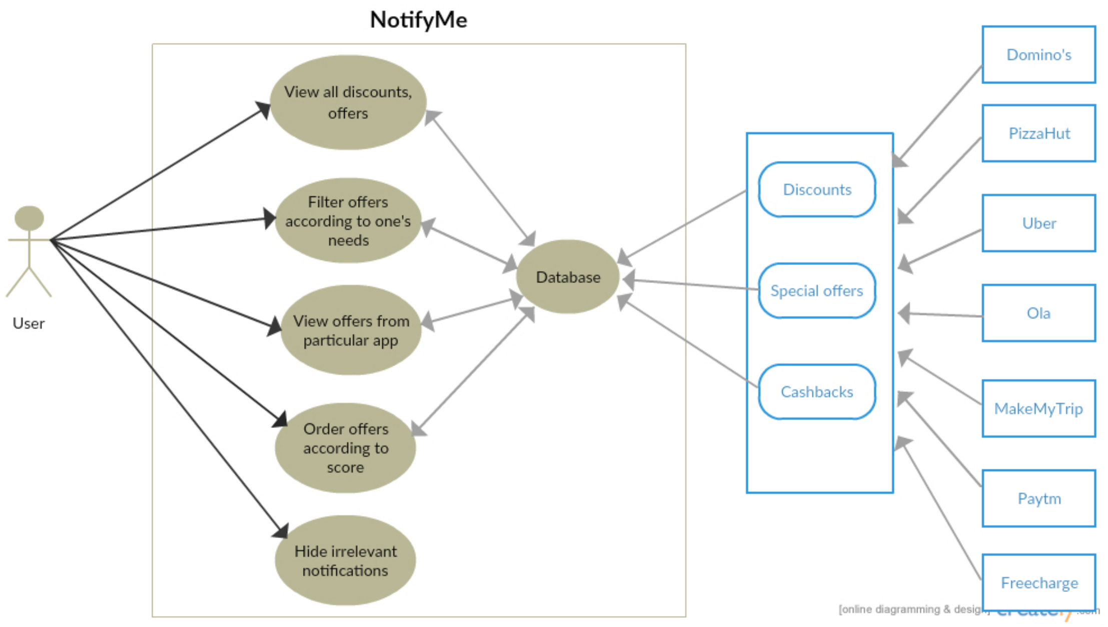

NotifyMe
User
View all discounts, offers
Filter offers according to one's needs
View offers from particular app
Order offers according to score
Hide irrelevant notifications
Database
Discounts
Special offers
Cashbacks
Domino's
PizzaHut
Uber
Ola
MakeMyTrip
Paytm
Freecharge
[online diagramming & design] creately.com

Figure 4: Use Case Diagram

2.3 Use Case Description

The Use Case diagram shows how the user interacts with the NotifyMe app. The user can use the NotifyMe app to view all promos and apply filters to view promos satisfying a particular criteria. User can view promos from a particular app by opening it's respective folder in the NotifyMe app. User can also order the promos according to score or hide irrelevant notifications by setting the appropriate options in the app. Each use case has been discussed in detail in the next section.

2.4 User Characteristics

This app should not require any level of technical expertise from the user. As everyone likes a good discount, this app will be useful for anyone who uses smart-phones especially those who use e-commerce applications on their phones.

The users of this app should be able to view offers from a particular seller, view offers on a particular item, and order them by any of the supported criteria.

2.5 Constraints

The Internet connection is a constraint for the application. Since the application uses the Text Processing services provided by TextRazor over the Internet, it is crucial that there is an Internet connection for the application to be able to process any new offers. This should, however, not hinder the application from providing the services to the user with the previously processed offers.

The mobile application may be constrained by the capacity of the Sqlite database provided by Android. By periodically deleting redundant offers from the databases, this constraint may be overcome.

2.6 Assumptions and Dependencies

We have assumed that the user has provided the application the permissions to read user's SMSes and to listen to notifications from other applications.

Another assumption about the product is that it will always be used on mobile phones that have at least 24MB of memory available for the app. Also we have assumed that the smart-phone will not be completely devoid of Internet connection.

Moreover we have assumed that the promos will have a general design and contain some keywords as mentioned earlier.

2.7 Apportioning of Requirements

In future we would like to make the app more reliable by improving the text processing algorithms and using machine learning to tweak the message scoring system.

3. Specific Requirements

3.1 External Interface

This section provides a detailed description of all inputs into and outputs from the system.

3.1.1 Notification Listener

Characteristics	**Description**
Name of Item	Notification Listener API
Description of Purpose	Listens for incoming notifications
Source of Input	The notifications from different applications
Destination of output	NotifyMe application's text processing API running in the background
Valid range	All the notifications coming on the smart-phone

3.1.2 Text Processing

Characteristics	**Description**
Name of Item	Text Processing API
Description of Purpose	Parses and extracts information from the message/notification
Source of Input	Notifications coming from the Notification Listener API
Destination of output	NotifyMe's database
Accuracy	Fairly accurate assuming that the offer messages are of generic design
Data format	String
Timing	Gives output within one second

3.1.3 Database - Storage

Characteristics	**Description**
Name of Item	Sqlite Database
Description of Purpose	Stores promos and gives query results on them
Source of Input	Output of the Text Processing API
Destination of output	NotifyMe application window
Accuracy	Accurately outputs the messages meeting the query requirements
Data format	List of Messages (Strings)
Timing	Gives query results within one second

3.1.4 Database - Ordering

Characteristics	**Description**
Name of Item	Sqlite Database
Description of Purpose	Orders the promos on the basis of their score
Source of Input	Output of the Text Processing API
Destination of output	NotifyMe application's folders which would contain ordered messages corresponding to a particular app.
Accuracy	Accuracy of ordering depends on scoring mechanism. Good accuracy for messages of generic design
Data format	Ordered List of Messages (Strings)

3.2 Functional Requirements

3.2.1 Use Case 1

3.2.2 Description

- **Brief description -** This use case describes how the user uses the NotifyMe app to view all promos.
- **Actors -** App User
- **Preconditions -** User has given permissions to the app.
- **Basic flow of events**
  1. The user opens the NotifyMe app.
  2. App displays the current offers to the user.
  3. User views all the promos displayed on the screen.

- **Alternative flows**

  1. *No offer present-* If there are no offers in the database, the app displays an error message.

- **Post-conditions**

  a *Successful operation* - User can view all the offers at once.

  b *Failed operation* - The system state remains unchanged.

- **Special Requirements -** None

3.2.3 Process Flow Diagram

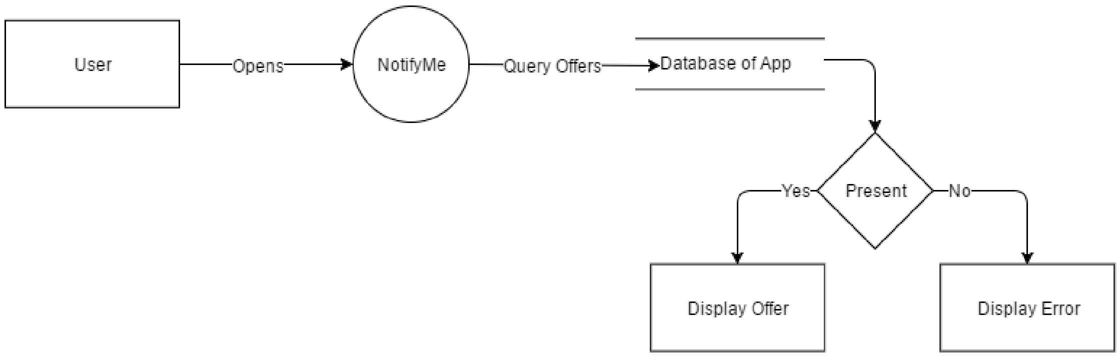


Figure 5: Process Flow Diagram for use case 1

3.2.4 Use Case 2

3.2.5 Description

- **Brief description -** This use case describes how the user uses the NotifyMe app to filter notifications.

- **Actors -** App User

- **Preconditions -** User has given permissions to the app.

- **Basic flow of events**

  1. The user opens the NotifyMe app.
  2. App displays the current offers to the user.
  3. User specifies a filter to view particular offers.
  4. The filter is applied on the database of offers and the required offers are displayed.

5. User notes the coupon code and other relevant information about his offer of interest.

- **Alternative flows**

  1. *No such offer* - If no offer meets the filter requirements, or there are no offers in the databse, the app displays an error message. The user may specify a different filter or cancel the operation

- **Post-conditions**

  a *Successful operation* - User gets information about his offer of interest.

  b *Failed operation* - The system state remains unchanged.

- **Special Requirements -** None

3.2.6 Process Flow Diagram

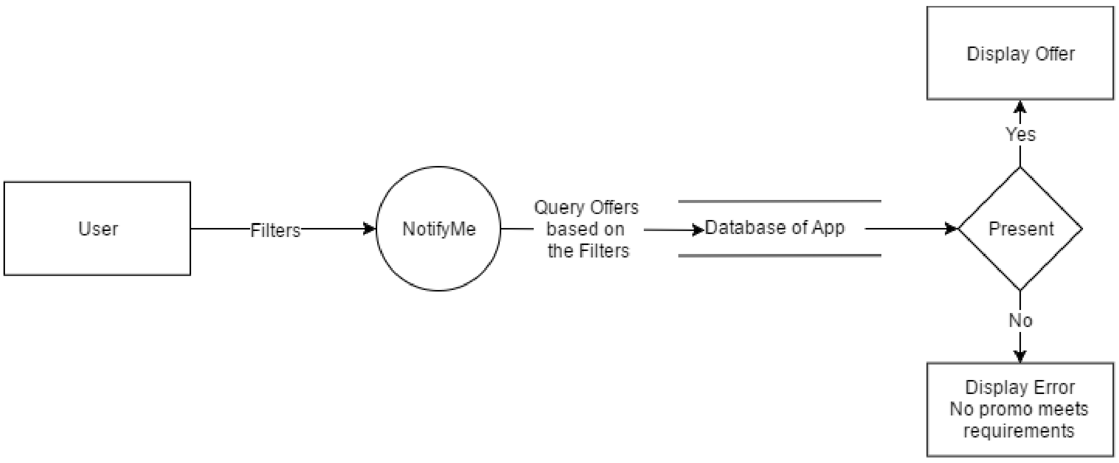

User
Filters
NotifyMe
Query Offers based on the Filters
Database of App
Present
Yes
Display Offer
No
Display Error No promo meets requirements

Figure 6: Process Flow Diagram for use case 2

3.2.7 Use Case Description-3

- **Brief description -** This use case describes how the user uses the NotifyMe app to view promos from a particular app.

- **Actors -** App User

- **Preconditions -** User has given permissions to the app.

- **Basic flow of events**

  1. The user opens the NotifyMe app.

2. App displays the current offers to the user.
3. User clicks on the 'Folders' tab. A list of folders is displayed containing promos from particular apps.
4. User selects one of the folders corresponding to his app of interest.
5. User searches for a particular offer by typing keywords of the offer.
6. User notes the coupon code and other relevant information about his offer of interest.

- **Alternative flows**

  1. *No such offer* - If no offer contains the keyword, or there are no offers in the database, the app displays an error message. The user may specify a different keyword, open a different folder or cancel the operation

- **Post-conditions**

  a *Successful operation* - User gets information about his offer of interest.
  b *Failed operation* - The system state remains unchanged.

- **Special Requirements -** None

3.2.8 Use Case Description-4

- **Brief description -** This use case describes how the user uses the NotifyMe app to order promos according to score.
- **Actors -** App User
- **Preconditions -** User has given permissions to the app.
- **Basic flow of events**

  1. The user opens the NotifyMe app.
  2. App displays the current offers to the user.
  3. User opens the 'Options' menu in the app.
  4. User can select/deselect the 'order' checkbox to view the offers in an ordered fashion according to the score.

- **Alternative flows**

  1. *No offer present*- If there are no offers in the database, the app displays an error message.

- **Post-conditions**

  a *Successful operation* - User can view the promos ordered according to the score.
  b *Failed operation* - The system state remains unchanged.

- **Special Requirements -** None

3.2.9 Use Case Description-5

- **Brief description -** This use case describes how the user uses the NotifyMe app to hide irrelevant notifications.
- **Actors -** App User
- **Preconditions -** User has given permissions to the app.
- **Basic flow of events**
  1. The user opens the NotifyMe app.
  2. App displays the current offers to the user.
  3. User opens the 'Options' menu in the app.
  4. User can select/deselect the 'Hide irrelevant notifications' checkbox to hide those notifications which have a score less than the user specified score. The score is specified in the 'Score threshold' textbox.
- **Alternative flows**
  1. None
- **Post-conditions**
  a *Successful operation* - Low score notifications will be hidden from the user.
  b *Failed operation* - The system state remains unchanged.
- **Special Requirements -** None

3.2.10 Data Flow Diagram 1

The diagram in figure 7 shows what happens when a message/notification is received and the required permissions are set.

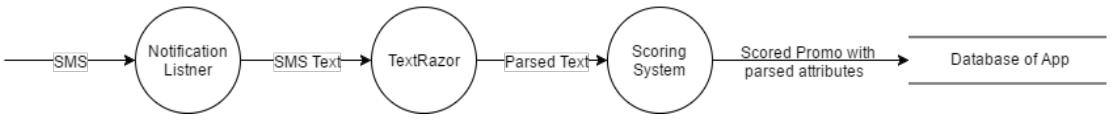


Figure 7: Data Flow on receiving notifications

1. Message is received.
2. Notification listner receives it and forwards the text to our application, NotifyMe. NotifyMe accesses the message and parses it using a Text Processing API (TextRazor).
3. If the message is a promo, it is forwarded to our scoring system where it gets a score according to its contents. Otherwise, the message is left untouched. Scored Promo is then forwarded to be stored in the database.

3.2.11 Data Flow Diagram 2

In figure 8 we demonstrate what happens when a user opens the app and applies a filter.

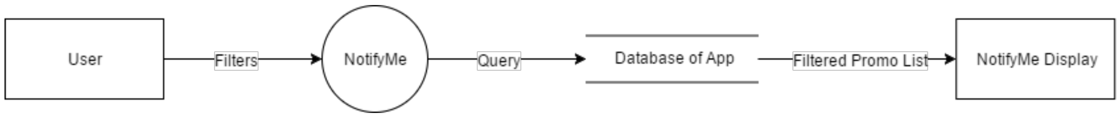


Figure 8: User Data Flow Diagram

1. User opens the NotifyMe application.
2. All the offers are displayed on the screen by the app.
3. User applies a filter according to his needs by interacting with the application.
4. The filter query is sent to the database which gives the results to the application.
5. The required results(promos) are displayed on the screen.

3.3 Performance Requirements

The application will handle information about all the messages and notifications from the apps listed by the user which correspond to discounts, cashbacks, special offers, etc. As a user applies filters to view the offers according to his needs, the required offers will be displayed on the screen within a second.

3.4 Logical Database Requirements

App and Offers are the two entities of our DB. Offers is related to App by a "belongs to" relationship. There is only one attribute for Application, its name. On the other hand Offers entity has several attributes: discount amount, terms and conditions, validity, coupon code, score, arrival date. These form the basic logical database requirements. The figure 9 shows the Chen ERD for the same.

3.5 Design Constraints

The application shall work on Android Lollipop (Version 5.0) and above. Use of versions below this may cause poor performance or some features may not work.

3.6 Software System Attributes

3.6.1 Reliability

The app may not always provide the best offer at all times, though we want it to be as reliable as possible. Our aim would be to guarantee that our suggested offer is at least the next best nonetheless. The app performs reliably when the promos being received possess keywords on which the text processing and scoring systems are based.

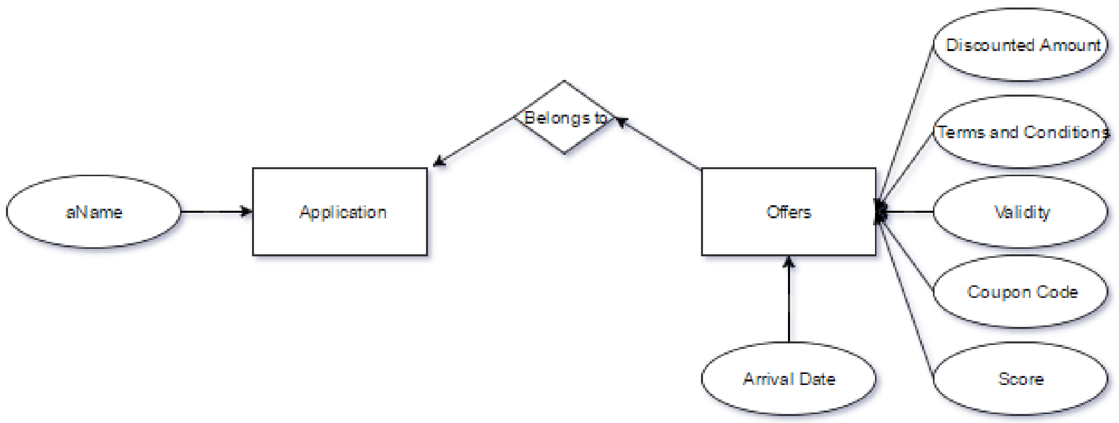


Figure 9: ER Diagram for the application

3.6.2 Availability

The application runs in background whenever a new promo arrives. This way it can process the promo and add it to the sorted list in its database if the promo meets the threshold. The application may not start on its own after a reboot so the user must take care of this.

3.6.3 Security

As the app doesn't hold any private data of the user, and thus won't require any authentication for use. The communication of the app over the Internet will be with the TextRazor servers. This communication also should not carry any sensitive information.

3.6.4 Maintainability

The app is easy to maintain as the code will be modular: UI, TextProcessing, DB, etc. are separately coded. This reduces the interdependence of these components and allows parallel development.

3.6.5 Portability

The app is targeted for Android platform and will not work on Windows or IOS devices. Though the TextProcessing module can be shared for those platforms, UI will have to be developed from scratch.

4. Supporting Information

4.1 Table of Content