CUSTOMER REWARDS FOR SOFTWARE APPLICATION RECOMMENDATION

This provisional patent application is intended to describe one or more embodiments of the present invention. It is to be understood that the use of absolute terms, such as “must,” “will,” and the like, as well as specific quantities, is to be construed as being applicable to one or more of such embodiments, but not necessarily to all such embodiments. As such, embodiments of the invention may omit, or include a modification of, one or more features or functionalities described in the context of such absolute terms.

Embodiments of the invention may be operational with numerous general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

Embodiments of the invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer and/or by computer-readable media on which such instructions or modules can be stored. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

Embodiments of the invention may include or be implemented in a variety of computer readable media. Computer readable media can be any available media that can be accessed by a computer and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by computer. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term “modulated data signal” means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

According to one or more embodiments, the combination of software or computer-executable instructions with a computer-readable medium results in the creation of a machine or apparatus. Similarly, the execution of software or computer-executable instructions by a processing device results in the creation of a machine or apparatus, which may be distinguishable from the processing device, itself, according to an embodiment.

Correspondingly, it is to be understood that a computer-readable medium is transformed by storing software or computer-executable instructions thereon. Likewise, a processing device is transformed in the course of executing software or computer-executable instructions. Additionally, it is to be understood that a first set of data input to a processing device during, or otherwise in association with, the execution of software or computer-executable instructions by the processing device is transformed into a second set of data as a consequence of such execution. This second data set may subsequently be stored, displayed, or otherwise communicated. Such transformation, alluded to in each of the above examples, may be a consequence of, or otherwise involve, the physical alteration of portions of a computer-readable medium. Such transformation, alluded to in each of the above examples, may also be a consequence of, or otherwise involve, the physical alteration of, for example, the states of registers and/or counters associated with a processing device during execution of software or computer-executable instructions by the processing device.

As used herein, a process that is performed “automatically” may mean that the process is performed as a result of machine-executed instructions and does not, other than the establishment of user preferences, require manual effort.

One or more embodiments of the invention, and/or components thereof, may be referred to herein by the term “Refer Engine,” and may include an online application that integrates application stores of different platforms with social networks and enables app developers to integrate their apps in a powerful way.

DETAILED DESCRIPTION OF ONE OR MORE EMBODIMENTS

Refer Engine consists of several customer facing experiences.

### ReferEngine.com

This is the main Refer Engine website which shows the information about applications in different platforms like iOS App Store, Google Play, Windows Store, etc… The user can browse these applications by category/subcategory and sort by different criteria like ratings, number of recommendations, etc… Once logged in to Facebook, the user can also browse the apps that different friends use and sort those by category, platform, most liked, etc…The user can also search for applications in different stores or in specific app stores.

Refer Engine may include different modes of browsing apps. The user can sort and view using known methods such as sort by rating, sort by number of likes, etc… Refer Engine also offers different viewing modes: browse 1 app at a time, browse a grid of apps (default), and browse by app screenshots only. Another browsing mode is to sort randomly and show one random app at a time. This helps with discovering apps that may otherwise not come up to the surface.

ReferEngine.com may fully integrate with the social graph of Facebook. Every Store, Category and App page may be a Facebook Graph object that support native Facebook graph actions like sharing, liking and posting on timelines, etc. The user can also choose to perform specific social actions like ‘liking’ or sharing and custom ReferEngine social actions like ‘use this app several times a day’, ‘use this app every day’, ‘use this app once a week’. Other social actions may be to specify when you use an app, for example: ‘use this app when I’m bored’, ‘use this app to listen to coldplay music’, etc.

ReferEngine may enable app developers to fully control the listings of their apps. Once app developers register and verify their ownership of an application, they can modify all text, add high resolution screenshots and videos and connect their twitter and Facebook feeds on the page. This creates very powerful landing pages for these applications that can be controlled by the developer to a large extent.

ReferEngine.com/Developer

This may be the developer portal where application developers can register their ownership of applications and modify content that shows up within their application’s page on ReferEngine.com. The developers can also get the code packages that they need to install REAPEX (below) within their applications.

To verify that a user is really the developer of a specific application, the developer may perform these steps:

* Login to ReferEngine.com and select the app that they own
* Install the ReferEngine.Verification code library in the app (for example, in the Windows platform they may be asked to install a NuGet package)
* Enter the Refer Engine Verification Passcode in a specific configuration section inside the app
* Run the app from their machine. The app may now launch and run the Refer Verification code which collects the app receipt and the passcode and sends them to the Refer Engine servers. The Refer Engine servers can verify the app receipt and that the passcode is correct. If all is good, the developer is verified.
* The developer can then remove the verification code from their app.

The developer can specify what kinds of rewards they wish to offer their customers and how many recommendation points each of those rewards cost. Examples could be:

* 4 game points for every 1 recommendation point
* 10 game points for every 2 recommendation points
* 20 game points for every 3 recommendation points  
    
  ReferEngine In-App Experience (REAPEX)

The ReferEngine In App Experience (REAPEX) is an additional feature that app developers can sign up for and integrate with their apps. Once integrated, REAPEX offers the customers the ability to:

* Sign up as a Refer Engine Verified User of the app
* Submit a Verified Recommendation of the app

The Verified User is a Refer Engine concept for customers to identify themselves as really using this app. If the customer particularly likes this app, they can submit a verified recommendation. Verified recommendations are recommendations that can only happen from within the applications. The developer can optionally choose to offer an incentive or an in-app reward based on the result of the customer’s recommendation. Refer Engine may track these recommendations and calculate what rewards every customer earns. In order to maximize their chances of collecting Recommendation Points, the user can submit multiple recommendations. To minimize spam, the user can only submit one recommendation per app every certain period of time (currently set at one month).

Developers who participate in the In-App experience have additional capabilities such as contacting their customers directly through Refer Engine.

Another task that REAPEX does is to offer the customer the ability to redeem the rewards they earned from their recommendations. Whenever the app launches, REAPEX checks with ReferEngine.com servers and determines whether the user has earned a reward. If the user earned a reward, it pops up and notifies the user of the reward. If the user chooses to claim the reward, REAPEX may call a pre-defined function inside the app to notify the app of the reward to be claimed. The app gives the user the reward (game points, etc…) and calls back into REAPEX that the reward has been claimed successfully. REAPEX may then send this info to ReferEngine.com servers and the handler may deduct the appropriate points from the user’s Recommendation Points.

## Implementation

Refer Engine may be implemented as a native cloud application with several cloud service endpoints, background workers and data stores.

### ReferEngine.Cloud

ReferEngine.Cloud may be the main cloud service that responds to all requests made to any part of [www.ReferEngine.com](http://www.ReferEngine.com). This includes all three services mentioned above. It includes those components:

#### ReferEngine.Web Web Worker

This may be the main worker that all requests made to [www.ReferEngine.com](http://www.ReferEngine.com) are services. It may be implemented as an ASP.NET MVC Application.

#### ReferEngine.DataWriter

This may be a background worker that the Web Worker sends database updates to. This may be used so web requests are not blocked on database writes which could be costly.

### ReferEngine.Workers

This may be a cloud service that contains several background workers that are optionally advantageous to make Refer Engine work as desired.

#### ReferEngine.Workers.WinApps

This may be an automated background worker that wakes up once every 24 hours. It starts by pinging the Windows Store Sitemap for the links to all apps ([http://apps.microsoft.com/windows/sitemap/sitemap\_{0}.xml](http://apps.microsoft.com/windows/sitemap/sitemap_%7b0%7d.xml)) where {0} is the page number. All the links to the apps are stored in the database.

Once the worker has collected all the links, it then pings every link one by one and parses the information for every app. This information includes Name, Age Rating, Publisher Name, Description, Website Link, Privacy Policy, Screenshots, Supported Languages and Architectures, etc…

The information for all these apps are stored in the database. If an app link fails to load, we store the number of failures and once it fails to load 5 days in a row, we consider it as a deleted application and delete its information from the database.

Once the process is finished, the worker sleeps for 24 hours then wakes up and repeats.

#### ReferEngine.Workers.iOSApps

This worker works in a very similar way to the REFERENGINE.WORKERS.WINAPPS worker but instead of collecting information about the Windows 8 apps, it collects information about iOS (iPhone and iPad) apps.

#### Additional Collector Works

Addition app collection workers are planned to collect information about applications in the Android (Google Play) store, Windows Phone, Blackberry and other application stores.

#### ReferEngine.Workers.RewardGenie

This may be the worker that scans the latest activity that is happening on Refer Engine and determines which users have earned which rewards from which app developers.

When this worker wakes up, for every app that is signed up with the Refer Engine In App Experience: it scans the database for any new verified users and verified recommendations added since it last operated.

* For every User A who registered as a verified user or submitted a recommendation for App-1
* How many Users B are there who: are a friend of A and who submitted a recommendation of App-1 within the last X weeks? (where X is a predetermined variable)
  + If there is zero Users B
    - User A is here organically.
  + If there is one User B
    - User B earns a recommendation point for App-1
  + If there is more than one Users B
    - User B who submitted a recommendation first may earn the recommendation point
    - All other Users B don’t earn a recommendation point. This is to avoid rewarding more than one recommenders recommendation points for one new user. Note that the recommendation “expires” after X # of weeks and is considered ineffective.