Upgrading Mobile Applications Dynamically through Crowdsourcing for Including New Languages

Taygun Bulut Durmaz, Susana Muñoz Hernandez
Department de Lenguajes y Sistemas Informáticos e Ingeniería de Software, 
Facultad de Informática, Universidad Politécnica de Madrid, SPAIN
tb.durmaz@alumnos.upm.es, susana@ft.upm.es

Abstract
There are multiple workshops and markets available on the web where communities of programmers develop mobile applications for several purposes. Most of these applications are not restricted to a particular country or region. In fact, the users of a mobile application might speak many different languages. According to research, more than 99 percent of mobile applications are developed in 9 languages. However, there are more than 6000 languages spoken around the world. For those people who do not speak one of these 9 languages, how are they supposed to use most mobile applications on the market? The MilanApps project has an answer for this problem. Mobile applications usually have user interfaces that contain text in the form of words or sentences. In order to make the application usable by people who speak different languages, this text would need to be translated. The MilanApps project will make this feasible by providing translations of the user interface text of mobile applications to any language.

Keywords: Localisation, translation, mobile apps, TEDECO, MilanApps

1 Introduction

1.1 Mobile Applications
A mobile application, or mobile app, is a computer program designed to run on smartphones, tablet computers and other mobile devices.

Mobile apps were originally offered for general productivity and information retrieval, including e-mail, calendar, contacts, stock market and weather information. However, public demand and the availability of developer tools resulted in a rapid expansion into other categories, such as games, factory automation, GPS and location-based services, banking, order-tracking, ticket purchases and recently, medical apps. The explosion in the number and the variety of available apps made new app discovery a challenge, which in turn led to the creation of a wide range of sources for review, recommendation, and curation, including blogs, magazines, and dedicated online app-discovery services.

The popularity of mobile apps has continued to rise, as their usage has become increasingly prevalent across mobile phone users (Ludwig 2012). A comScore study in May, 2012 reported that in the previous quarter, more people used their mobile devices for apps than for browsing the web: 51.1% vs. 49.8% respectively (Perez 2012). Researchers found that usage of mobile apps strongly correlates with user context and depends on user’s location and time of the day (Böhmer et al. 2011).

1.1.1 Demands
We estimate that native English speakers account for only 34%, 39%, and 25% of iOS, Android, and Windows Phone downloads, respectively. (See Figure 1)

![Figure 1: Market demands according to majority platforms from Xyologic (Tethras 2013)]
Native English speakers currently make up approximately 33% of the total addressable market on average. If we consider the non-English market among the three major platforms (iOS, Android, and Windows) by concentrating on the top 8 languages, we have the following result (See Figure 2).

The results show that the English market has 33% (average) of total downloads, followed closely by the Chinese market with 23%, and then the Spanish market with 13%.

The situation is unlikely to change as current demand determines future supply. In other words, developers and companies attempt to meet their customers’ expectations according to their demand. This situation tends to maintain English, Chinese, and Spanish as the main languages of mobile apps.

1.1.2 Results

The market chain of mobile apps causes an imbalance between spoken languages versus app languages since 85% of developers publish in English addressing just 8% of the world population (around 500 million people) who speak English as a primary language. Meanwhile, Chinese, spoken by 22% of the world’s population, only attracts 16% of developers. The choice of English by the majority of developers happens not only on a global, but on a regional basis as well, putting local languages supply at a deficit. In South America, Spanish is used by 84% of developers, while English is only used by 48% (Kapetanakis 2012).

1.2 Minority Languages

According to UNESCO 6,000-7,000 languages are spoken in the world today. However, about 300 widespread or majority languages account for 90 percent of the world’s population. More than half of the remaining languages are endangered.

“Minority languages have been oppressed, denied, and neglected for a long time, and decline is accelerating. Whereas estimates show that half of the world’s languages disappeared from 1450 to 1950, half of the remaining 6000 to 7000 languages could disappear in this century alone” (Sheyholislami 2009).

Language plays an important role in learning. Since language is the main medium of communicating meaning in most learning activities, it is essential to use in education a language that learners understand and speak. Usually people understand their first language best, and are most comfortable speaking it. Multilingual people may be equally proficient in several languages. The first language is also often called the mother tongue, or the home language. Generally, the first language is a language one has learned first; one identifies with or is identified as a native speaker of by others; or one knows best (Unesco 2005).

Some observers consider new media technologies such as satellite, television and the Internet, as the salvation of minority languages. They believe it is important to maintain and develop minority languages and see the media as having a crucial role in maintaining and developing minority languages. Minority language media are deemed important for:

- their symbolic role in acknowledging that minority cultures can deal with the contemporary world;
- their ability to legitimate the existence of the language that they use;
• their potential to provide an “economic boost” for those who are interested in working in the minority language;
• their instrumentality in engendering a public sphere within a language community;
• their resourcefulness in enabling minorities to represent their community, not only within itself but also to outsiders instead of being represented by “others”;
• their capability to be conveyers of cultures and producers of cultural products; and
• their capability to magnify discursive practices of identity construction (Cormack and Hourigan 2007).

1.3 Proposal
There are multiple forges and markets available on the web where communities of programmers develop mobile applications for several purposes. Most of these applications are not restricted to a particular country or region. In fact, the users of a mobile application might speak many different languages. The most common scenario is that software is only available in one or two popular languages, such as English, French, Spanish or Chinese. Another possibility is that the software is available only in the language of the developers.

But what is the consequence of not having software adapted to local languages? That the software is in many cases restricted to people who know foreign languages. What about applications for the common population? What about applications for children?

The consequences for software applications, that try to be popular in the market, are dramatic because its target market is limited by the language of its user interface messages and tags.

It is often the case that the developers are the ones writing and translating the user interface text of mobile applications. But why not open the work to the collaboration of non-technical staff in the development team? They could aid in translating the user interface text of software applications to local languages.

The idea behind this proposal is, as in other issues where user collaboration is assumed (for fulfilling opinion surveys, to evaluate efficiency of services, to provide feedback, etc), to take into account the collaboration of users to load new languages to mobile applications.

The effective way of implementing this is not technically difficult but implies logistic planning, including a review process and an automatic way of loading new languages, which is currently not a global recommendation though it probably should be one. Here we provide a proposal for doing this in a simple and automatic way that would work for any software project intended to be multilingual.

2 State of the Art

2.1 MilanDi Project
The MilanDi project can be considered a “baby step” towards one of our aims: saving the minority languages. MilanDi (Multilingual dictionary for minority languages) is a dictionary for the minority languages that are spoken around the world. The goal was to create a mobile application using Universal Natural Language (UNL) and develop it together with other languages.

This open source project is composed of two main parts. The first part is a webpage where people around the world can contribute with translations to their own languages and by doing this, enrich the MilanDi system database. The second part is a mobile application that serves as a client for the MilanDi system in Android mobile devices.

The MilanDi webpage contains around 6000 UNL words, currently including English, French, Spanish, Swahili, and Kirundi. Translators can login to the system and translate words and sample sentences to their own language. Thanks to the contribution of local translators, the MilanDi project expects these 6000 words to be translated to many languages, especially minority languages. The MilanDi webpage will be published on TEDECO servers after acquiring the necessary permissions.

The MilanDi mobile application is an Android application that uses the collected data on the webpage. Users can use the application to translate words according to their needs. They can also update the word database through the application. If the mobile phone is connected to the Internet, the application fetches the new database from the MilanDi webpage and updates the offline database of the application.

MilanDi is a successful project that expresses the power of collaborative contribution. Instead of searching for the translation of a word, which is almost impossible for some minority languages as they are not available on the Internet, local translators
can contribute to this dictionary to save their own language.

3 Prototype: MilanApps

3.1 Description
The MilanDi project made us realize that collaborative contribution could be a very powerful tool in preserving and even spreading minority languages. We then decided to get contributions from local translators for language support of mobile applications.

Mobile applications usually have user interfaces that contain text in the form of words or sentences. In order to make the application usable by people who speak different languages, this text would need to be translated. The MilanApps project will make this feasible by providing translations of the user interface text of software applications to any language.

The MilanApps project has two main aims in its essence. The first aim is for people living in developing countries to use mobile applications in their own language. The second aim is to save minority languages through increasing awareness of them.

In order to accomplish these aims, we have created a portal for developers to work in collaboration with reliable translators. Developers of mobile applications will upload their user interface text to the MilanApps portal and reliable translators will translate them into their language of expertise. Therefore, developers will have the capacity to create multilingual applications for mobile apps users.

People understand their first language best, are most comfortable speaking it, and for this purpose, learning activities should mainly be done in their own language. The MilanApps project gives them this opportunity. People will be able to use software in their own languages. Consequently, people will be able to cope with new technologies, at least mobile applications, much more easily.

Additionally, MilanApps will help to spread minority languages in mobile applications. According to Portio Research, there were 1.2 billion mobile application users worldwide in 2012 and this number was expected to reach 4.4 billion users at the end of 2017 (Portio Research 2013). If minority languages can enter a market with such vast potential as that of mobile apps, the awareness of them in the world would surely increase.

So how will this system work?

1. Developers around the world will send their application name with the text they have used on their user interface.
2. After their request is approved, the text will be recorded on the MilanApps system and will be visible to translators around the world.
3. Reliable translators will translate the text to their language of expertise.
4. Finally, developers will fetch the translated text.

We have also created an Android-based test application for MilanApps to translate its text into a variety of languages. Reliable translators have connected to our local server and contributed to the project by translating the UI text of this test application to their local languages. We received the results in one day and our test application currently supports English, Turkish, Spanish, and Kurdish. You can see the screenshots in the “Translation Forms” section.

The MilanApps project prototype is ready and working appropriately. It will be published on TEDECO servers after acquiring the necessary permissions.

Our expectation for this project is for both developers and local translators to contribute to the system. With the help of more people, more applications can be translated into local languages and our project will achieve its goals. If this project is able to spread around the world and become a standard, people would be capable of using mobile applications in their own languages. As a result, language barriers would decrease and minority language speaking populations could reach technology much easier.

3.2 Open Source and Free to Use
The MilanApps project is developed for the people who do not have access to technology in their own languages. We want to increase language support for mobile applications not only for majority languages, but also for minority ones.

It is possible to find similar platforms to MilanApps on the Internet. They intend to provide a solution for the localization of mobile applications. However, one
cannot solve a minority language problem without knowing the expectations of minority language speaking users. These platforms only support majority languages or request money for translations. This situation severely hampers localisation. First of all, it is useless for minority languages if the platform only supports translation of majority languages. Secondly, developers are less likely to give language support for minority languages if they need to pay for the translations.

The idea behind the MilanApps project is that everyone can contribute with translations to help other people. Since this is a free-to-use platform, both developers and translators can contribute without having any extra expenses. Developers can upload their UI text to the platform and wait for translators to translate those into local languages. Therefore, developers can easily add extra language support without much effort. Moreover, this project is Open Source so any developer around the world can contribute to improve the platform according to user needs. As a result, MilanApps gets its power from being Open Source and free to use. Collective collaboration has an amazing potential for translations. MilanApps taps into this potential as part of a non-profit organization, TEDECO.

3.3 SourceForge Repository
SourceForge is a web-based source code repository. It acts as a centralized location for software developers to control and manage free and open source software development. It was the first platform to offer this service for free to open source projects (James 2007).

MilanApps has a repository on SourceForge where developers around the world can contribute to our prototype. This promising open source project hopefully will help to universalize the translation of the mobile applications.

4 Conclusion
UNESCO states that there are around 6500 languages in the world. However, 90% of the languages do not appear on the Internet. In our globalized, modern world the tendency is to use the common languages of the developed world, such as English or French, for business, commerce, education, and any other information interchange.

The market for mobile applications is similar to the above scenario. Developers and companies attempt to meet their customers’ expectations according to their demand. Customers of mobile applications are mainly distributed among developed countries, a factor which ultimately determines the languages of mobile applications. More than 99 percent of mobile apps are developed in English, Chinese, Spanish, Japanese, German, French, Korean, Portuguese, Italian, or Russian.

What about applications for the populations of less developed countries? How can they use mobile applications in their own languages?

The MilanApps project facilitates translations of the user interface text of software applications to local languages. We have created a portal for developers to work in collaboration with reliable translators. Developers of mobile applications will upload their user interface text to the MilanApps portal and reliable translators will translate them into their own language of expertise. Therefore, developers will have the capability of creating multilingual applications for mobile apps users.

These multilingual applications will, of course, mainly, be tailored for people living in less developed countries and written in their local languages. Since people understand their first language best, and are most comfortable speaking it, learning activities should mainly be carried out in this own language. The MilanApps project gives them this opportunity. People will be able to use software in their own language. Consequently, people will be able to cope with new technologies, at least mobile applications, much more easily.

Moreover, this project will not only help people who are living in developing or non-developed countries, but also help preserving minority languages. As minority languages are more widely spread in mobile applications, the awareness about them will also increase proportionally among communities.

The MilanApps project prototype is currently ready and working properly. The MilanApps portal will be published on TEDECO servers for developers and translators use. Our expectations for the future are that increasingly more reliable translators will join the portal to expand the language translation capacity. As the portal becomes more popular among developers and translators, our project will reach more users and achieve its goals.
4.1 What next?
In the future we expect that more translators will join the project. However, this situation may cause
reliability problems related to the control mechanism of translators and translated text.

Translators will translate text into their own language of expertise. However, with an increased number of
translators, it is possible that a ‘fake’ translator could ruin the translations with dummy text. In order to
address this issue, we have created another role called “Reviewers”. Reviewers will be responsible for
checking the correctness of the translations. After their approval, we will then publish the translated
text. This safety mechanism will keep the portal reliable by avoiding ‘fake’ translators.

A second problem could be caused by an excessive number of requests by translators. Currently, we have
to check the CVs of translators to give them permission to translate text into their language.
However, if a large number of requests come through the system, it would cause the permission mechanism
to progress very slowly. Therefore, we are going to allow translators to login to the system through a
reliable CV webpage, such as LinkedIn. This will help us check the translators’ CV’s and their reliability more easily.

References


IDE, A. (2013) PHP Just Grows & Grows | Netcraft [online], available: http://news.netcraft.com/archives/2013/01/31/php-
just-grows-grows.html [accessed 22 July 2014].

James, M. (2007) The SourceForge Story - Datamation [online], available:


Ludwig, S. (2012) Study: Mobile App Usage Grows 35%, TV & Web Not so Much [online], VentureBeat, available:


Sqlite (2015) About SQLite [online], available:

Tethras (2013) Into Which Languages Should I Localize My Mobile App | Blog [online], available:

Unesco (2005) First Language First: Community-
Based Literacy Programmes for Minority Language Contexts in Asia | MTB-MLE Network [online], available: http://mlenetwork.org/content/first-language-first-community-based-literacy-programmes-minority-language-contexts-asia [accessed 22 July 2014]

W3C (2011) HTML5 Differences from HTML4 [online], available: http://www.w3.org/TR/html5-diff/ [accessed 22 July 2014]

Appendices

A. Technology Decisions

1. HTML
HTML5 is a core technology markup language used for structuring and presenting content for the World Wide Web. It is the fifth revision of the HTML standard (created in 1990 and standardized as HTML 4 as of 1997) (W3C 2011).

2. PHP
PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. As of January, 2013, PHP was installed on more than 240 million websites (39% of those sampled) and 2.1 million web servers (IDE 2013).

3. Bootstrap
Bootstrap is a free collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. In March 2014 it was the No.1 project on GitHub with over 65,000 stars and 23,800 forks (GitHub 2014).

4. SQLite3
SQLite is an in-process library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine. The core for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is currently found in more applications than we can count, including several high-profile projects (SQLite 2015).

5. AndroidSdk
The Android SDK provides API libraries and developer tools necessary to build, test, and debug apps for Android.

B. Use Cases

Reliable Translators: Reliable translators contribute to the MilanApps languagedatabase through the webpage.

MilanApps System: The MilanApps webpage.

Developers: Developers of mobile applications.
  • They introduce UI text of their applications to the MilanApps System.
  • They fetch the translated text after translators translate them.

Market: Server to download applications.

Users: Users of mobile applications. They can use the applications in their own language. (See Figure 3)

1. MilanApps Use Case Diagram
See Figure 4

Use Case UC1: Register with the System
Scope: MilanApps System
Level: Sub-function
Stakeholders & Interest:
  - Translator: Wants to register with the system without any problem.

 Preconditions: The translator is not registered.

Success Guarantee: The translator’s record is saved

Figure 3. Basic Workflow Scheme of MilanApps
and the translator is ready to login to the system.

**Main Success Scenario:**
1. The translator enters his/her e-mail address.
2. The translator enters his/her password.
3. The translator re-enters his/her password.
4. The translator submits the form.
5. The system records the form and sends an e-mail with an activation link to the translator’s e-mail address.
6. The translator clicks on the link from his/her e-mail account.
7. The system updates the information for the translator.

**Use Case UC2: Request for translating text**

**Scope:** MilanApps System  
**Level:** Sub-function  
**Stakeholders & Interest:**  
- Translator: Wants to request a language to translate text.  
- Admin: Wants to receive requests from translators.  
**Preconditions:** The translator’s record is saved and the translator is ready to login to the system.  
**Success Guarantee:** The translator’s request is saved and sent to the admin.  

**Main Success Scenario:**
1. The translator logs in to the system.
2. The translator enters a request for translating a language text.
3. The translator selects a language to translate.
4. The translator submits the form.
5. The system records the form and sends the information to the admin.
6. The system informs the user that the request is done.

**Use Case UC3: Translate Text**

**Scope:** MilanApps System  
**Level:** User goal  
**Stakeholders & Interest:**  
- Translator: Wants to translate a language text.  
**Preconditions:** The translator’s record is saved and the translator is ready to login to the system.  
**Success Guarantee:** The translator’s changes are saved to the database.  

**Main Success Scenario:**
1. The translator logs in to the system.
2. The translator enters the “translate a language” menu.
3. The translator selects the application.
4. The translator selects the reference language.
5. The translator selects the target language.
6. The translator submits the selections.
7. The system records the form and presents the table for translating.
8 The translator translates text.
9 The translator saves the changes.
10 The system updates the database according to the changes.

Use Case UC6: Edit translated text
Scope: MilanApps System
Level: User goal
Stakeholders & Interest:
- Admin: Wants to edit translated text.
Preconditions: None.
Success Guarantee: The admin’s changes are saved to the database.
Main Success Scenario:
1 The admin logs in to the system.
2 The admin enters the “translate a language” menu.
3 The admin selects the application.
4 The admin selects the reference language.
5 The admin selects the target language.
6 The admin submits the selections.
7 The system records the form and presents the table for translating.
8 The admin edits translated text.
9 The admin saves the changes.
10 The system updates the database according to the changes.

Use Case UC7: Add a new language
Scope: MilanApps System
Level: User goal
Stakeholders & Interest:
- Admin: Wants to add a new language to the system.
Preconditions: None
Success Guarantee: The new language is saved to the database.
Main Success Scenario:
1 The admin logs in to the system.
2 The admin enters the “add a new language” menu.
3 The admin enters the name of a new language.
4 The admin submits the form.
5 The system records the form and updates the database.

6 The system informs the admin that the process is done.

Use Case UC8: Apply Requests from Translators
Scope: MilanApps System
Level: User goal
Stakeholders & Interest:
- Admin: Wants to see the requests from translators and apply them.
Preconditions: The translators have sent a request.
Success Guarantee: The admin’s response is saved and the translator is ready to translate the requested language.
Main Success Scenario:
1 The admin logs in to the system.
2 The admin enters the “requests from users” menu.
3 The admin reviews the requests.
4 The admin applies a request.
5 The system records the form and updates the database.

Use Case UC9: Add a new Application
Scope: MilanApps System
Level: User goal
Stakeholders & Interest:
- Admin: Wants to add a new application to the system.
Preconditions: None.
Success Guarantee: New application is saved to the database.
Main Success Scenario:
1 The admin logs in to the system.
2 The admin enters the “add a new application” menu.
3 The admin enters the name of a new application.
4 The admin submits the form.
5 The system records the form and updates the database.
6 The system informs the admin that the process is done.

Use Case UC10: Add a new text for translation
Scope: MilanApps System
Level: User goal
Stakeholders & Interest:
- Admin: Wants to add a new text to the system.
Preconditions: None.
Success Guarantee: A new text is saved to the database.
Main Success Scenario:
1. The admin logs in to the system.
2. The admin enters the “add a new text” menu.
3. The admin enters new text.
4. The admin submits the form.
5. The system records the form and updates the database.
6. The system informs the admin that the process is done.

C. MilanApps Database Structure

1. Table: Language
   Entity 1: _id
   • Description: Id number of the Language, each language has a unique Id number.
   • Data Type: INTEGER
   • Primary Key: YES
   • Foreign Key: NO
   • Auto Increment: YES

   Entity 2: Name
   • Description: Name of the Language.
   • Data Type: TEXT
   • Primary Key: NO
   • Foreign Key: NO
   • Auto Increment: NO

2. Table: Apps
   Entity 1: _id
   • Description: Id number of the Application, each application has a unique Id number.
   • Data Type: INTEGER
   • Primary Key: YES
   • Foreign Key: NO
   • Auto Increment: YES

Figure 5: Entity Diagram of MilanApps Translation Module
Entity 2: Name
- Description: Name of the Application.
- Data Type: TEXT
- Primary Key: NO
- Foreign Key: NO
- Auto Increment: NO

3. Table: Sentences:
   Entity 1: _id
   - Description: Id number of the Sentence, each sentence has a unique Id number.
   - Data Type: INTEGER
   - Primary Key: YES
   - Foreign Key: NO
   - Auto Increment: YES

   Entity 2: Text
   - Description: Name of the Text.
   - Data Type: TEXT
   - Primary Key: NO
   - Foreign Key: NO
   - Auto Increment: NO

   Entity 3: IDApp
   - Description: Id Number of the Application.
   - Data Type: INTEGER
   - Primary Key: NO
   - Foreign Key: YES
   - Auto Increment: NO

   Entity 4: IDLanguage
   - Description: Id number of the language.
   - Data Type: INTEGER
   - Primary Key: NO
   - Foreign Key: YES
   - Auto Increment: NO

   Entity 5: IDAppSentence
   - Description: ID number of IDAppSentence.
   - Data Type: INTEGER
   - Primary Key: NO
   - Foreign Key: YES
   - Auto Increment: NO

4. Table AppSentences
   Entity 1: _id
   - Description: Id number of the AppSentences/
   - Data Type: INTEGER
   - Primary Key: YES
   - Foreign Key: NO
   - Auto Increment: YES

Entity 2: Name
- Description: Name of the IDAppSentence.
- Data Type: TEXT
- Primary Key: NO
- Foreign Key: NO
- Auto Increment: NO

Entity 3: IDApp
- Description: Id Number of the Application.
- Data Type: INTEGER
- Primary Key: NO
- Foreign Key: YES
- Auto Increment: NO

5. Database of MilanApps Translators
See Figure 6.

6. Table: Users:
   Entity 1: ID
   - Description: Id number of the User, each User has a unique Id number.
   - Data Type: INTEGER
   - Primary Key: YES
   - Foreign Key: NO
   - Auto Increment: YES

   Entity 2: Email
   - Description: E-mail address of the User, each User has a unique e-mail.
   - Data Type: TEXT
   - Primary Key: YES
   - Foreign Key: NO
   - Auto Increment: NO

   Entity 3: Password
   - Description: Password of the User.
   - Data Type: TEXT
   - Primary Key: NO
Figure 6: Entity Diagram of MilanApps Translators

- Foreign Key: NO
- Auto Increment: NO

**Entity 4: Admin**
- Description: Shows if the User is an admin or not.
- Data Type: BOOLEAN
- Primary Key: NO
- Foreign Key: NO
- Auto Increment: NO

**Entity 5: Activation**
- Description: Activation number which is sent to an e-mail address.
- Data Type: TEXT
- Primary Key: NO
- Foreign Key: NO
- Auto Increment: NO

**Entity 6: Temporary**
- Description: Shows if the user’s account is activated or not.
- Data Type: BOOLEAN
- Primary Key: NO
- Foreign Key: NO

**Entity 7: Permission**

**Entity 1: ID**
- Description: Id number of the Permission, each permission has a unique Id number.
- Data Type: INTEGER
- Primary Key: YES
- Foreign Key: NO
- Auto Increment: YES

**Entity 2: LanguageID**
- Description: Language id of the request, each request has a language ID.
- Data Type: INTEGER
- Primary Key: NO
- Foreign Key: NO
- Auto Increment: NO

**Entity 3: Active**
- Description: Shows if the permission is given or not.
Entity 4: UserMail
- Description: E-mail of the user who makes the request.
- Data Type: TEXT
- Primary Key: NO
- Foreign Key: YES
- Auto Increment: NO

8. MilanApps Website Map Diagram

See Figure 7

D. MilanApps Forms

1. MilanApps Translation Module for Translators

Login Page
This is the main page of the MilanApps website. If the translator does not have an account, he/she has to register with the system first.

If the translator has an account on the MilanApps system, he/she will enter his/her e-mail address and password to login.

- The translators will enter their username and password to enter the MilanApps System.
- They can register with the system by entering their e-mail address and password.
- The system will send them an activation e-mail to verify they are human.

Request Page
After the translator logs in to the system, he/she can translate and record text to his/her language of expertise. However, he/she needs to obtain permission for that language first. The translator can request a language for translation through this page. The translator’s request will be accepted by the admin if his/her reliability is verified.

- The translators will send a request for a language to translate.
- The admin will verify the translator and if he/she is reliable, the admin will apply the request.

Editing Page
In this page, translators can translate and record text to his/her language of expertise. They need to obtain permission for that language from the admin in order to translate text.
Figure 8: MilanApps Login Page

Figure 9: MilanApps Request Page

Figure 10: MilanApps Edit Language Page

Figure 11: MilanApps Add a new Language Page
Figure 12: MilanApps Add a New Application Page

Figure 13: MilanApps Add new Text Page

Figure 14: MilanApps Edit Language Page

Figure 15: MilanApps User Request Page
translators.

**E. MilanApps Test Application**

Figures 16, 17 and 18 show screenshots that belong to the test mobile application. The UI text of this application was entered into the MilanApps system. Translators translated the text to their own local languages. This application then fetched this new data to support more languages. Currently, this sample mobile application supports English, Turkish, Kurdish and Spanish.

**Main Menu** (shown in figures 16)
- This is the main menu of the application. Text is in English.

![Figure 16: Main Menu of MilanApps Test Application](image)

**Options Menu** (shown in figures 17 & 18)
In the options menu, users can select the language they prefer.
- User selects his language and submits
- The language of the application has been changed and the sentences are in Turkish now.
Figure 17: Options Menu of MilanApps Test Application

Figure 18: Translated Main Menu of MilanApps Test Application
F. MilanDi Project

1. Network Scheme

Reliable Translators:
Reliable translators are contributing Language MilanDi database through webpage (MilanDi Language Editing System).

MilanDi System:
MilanDi webpage which is Milandi Language Editing System

Mobile Application Developers:
Developers of mobile applications. They should build applications with MilanDi support.

Market:
Server to download applications

Milandi Enabled Applications:
MilanDi enabled applications can fetch the data from MilanDi system and able to convert language data according to user’s requests.

2. MilanDi mobile dictionary

![Figure 20: Use case diagram of MilanDi mobile dictionary](image)

Use Case UC1: Translate Word
Scope: MilanDi mobile dictionary
Level: user goal
Stakeholders & Interest:
-User: Wants to translate the word
Preconditions: none
Success Guarantee: User reach the translated word
Main Success Scenario:
1. User picks the language he/she wants to translate from
2. User picks the language he/she wants to translate
3. User enters the word he/she wants to translate
4. System records the form and directs the result to the user

Use Case UC2: Update Database
Scope: MilanDi mobile dictionary
Level: sub function
Stakeholders & Interest:
-User: Wants to update the database of the application
Preconditions: mobile device is connected to Internet
Success Guarantee: Recent database is downloaded and embedded into application
Main Success Scenario:
1. User clicks update button on main page
2. System receives data from MilanDi server and embeds into application
3. System informs user

3 MilanDi Language Editing System

Use Case UC3: Register to System
Scope: MilanDi Language Editing System
Level: sub function
Stakeholders & Interest:
- Translator: Wants to register to system without any problem

Preconditions: Translator is not registered

Success Guarantee: Translator’s record is saved and translator is ready to login to system

Main Success Scenario:
1. Translator enters his/her e-mail address
2. Translator enters his/her password
3. Translator re-enters his/her password
4. Translator Submits his form
5. System records the form and sends a mail with activation link to translators e-mail address
6. Translator clicks the link from his/her e-mail account
7. System updates translator information

Use Case UC4: Request for editing a Language
Scope: MilanDi Language Editing System
Level: sub function

Stakeholders & Interest:
- Translator: Wants to request a language to edit
- Admin: Wants to receive requests from translators

Preconditions: Translator’s record is saved and translator is ready to login to system

Success Guarantee: Translator’s request is saved and sent to admin

Main Success Scenario:
1. Translator logs in to the system
2. Translator enters request for editing a language menu
3. Translator picks a language to edit
4. Translator Submits his form
5. System records the form and sends a information to admin
6. System informs user that request has been carried out

Use Case UC5: Edit allowed Language
Scope: MilanDi Language Editing System
Level: user goal

Stakeholders & Interest:
- Translator: Wants to edit languages

Preconditions: Translator’s record is saved and translator is ready to login to system

Success Guarantee: Translator’s changes are saved to database

Main Success Scenario:
1. Translator logs in to the system
2. Translator enters edit a language menu
3. Translator picks reference language
4. Translator picks target language
5. Translator submits
6. System records the form and presents the table for editing
7. Translator edits word and descriptions
8. Translator saves the changes
9. System updates the database according to changes

Use Case UC6: Edit Language
Scope: MilanDi Language Editing System
Level: user goal

Stakeholders & Interest:
- Admin: Wants to edit languages

Preconditions: none

Success Guarantee: Admin’s changes are saved to database

Main Success Scenario:
1. Admin logs in to the system
2. Admin enters edit a language menu
3. Admin picks reference language
4. Admin picks target language
5. Admin submits
6. System records the form and presents the table for editing
7. Admin edits word and descriptions
8. Admin saves the changes
9. System updates the database according to changes

Use Case UC7: Add a new language
Scope: MilanDi Language Editing System
Level: user goal

Stakeholders & Interest:
- Admin: Wants to add a new language to the system

Preconditions: none

Success Guarantee: New language is saved to database

Main Success Scenario:
1. Admin logs in to the system
2. Admin enters add a new language menu

![Figure 21: Use case diagram of MilanDi Language Editing System](image-url)
Localisation Focus

3 Admin enters the name of a new language
4 Admin submits his form
5 System records the form and updates the database
6 System informs admin that process is done

Use Case UC8: Apply Requests from Translator
Scope: MilanDi Language Editing System
Level: user goal
Stakeholders & Interest:
Admin: Wants to see the requests from translators and apply them
Preconditions: Translators had sent a request
Success Guarantee: Admin’s response is saved and translator is ready to edit requested language

Main Success Scenario:
1 Admin logs in to the system
2 Admin enters requests from users menu
3 Admin reviews the requests
4 Admin applies the request
5 System records the form and updates the database