

Budget Tracker

System Requirement Statement (SRS)

**Content**

**Version: 0.1**

**the virtual solution**

1 Introduction 5

1.1 Purpose 5

1.2 Scope 5

1.3 Glossary 5

1.4 References 5

2 System Requirements 7

2.1 General User Interface Requirements 7

2.2 Functional Requirements 7

2.3 Technical Requirements 8

3 System Scope 8

4 Business Process and Data Definition 8

5 Functional Decomposition 9

5.1 System overview 9

5.2 Module: Registering new user 9

5.3 Module: Login/Logout 10

5.4 Module: Adding/Editing accounts 11

5.5 Module: Adding/Editing Incomes 14

5.6 Module: Adding/Editing Outcomes 15

5.7 Module: Adding/Editing Bills 19

5.8 Module: Watching/Analyzing Reports 20

5.9 Module: Reviewing quick summary 23

5.10 Module: Performing search by specified criteria 24

5.11 Module: Changing the app settings 25

5.12 Module: Writing review in AppStore 26

5.13 Module: Writing feedback 27

5.14 Module: Watching contact information 27

6 Operational Features 28

6.1 Performance 28

6.2 Implementation/Deployment 28

7 Business Impacts 28

8 Assumptions and Dependencies 28

9 Software Quality Attributes 28

10 Appendix 29

**Table of Figures**

Figure 1: Registration page 10

Figure 2: Login User 11

Figure 3: Logout User 11

Figure 4: Account properties 12

Figure 5: List of Accounts 12

Figure 6: Editing Account 13

Figure 7: Account details 13

Figure 8: Adding Income 14

Figure 9: List of Incomes 14

Figure 10: Income Details 15

Figure 11: List of Outcomes 16

Figure 12: Outcome properties 16

Figure 13: List of categories 17

Figure 14: Editing Categories 17

Figure 15: Adding category 18

Figure 16: Choosing location 18

Figure 17: List of Outcomes 19

Figure 18: Calendar view 20

Figure 19: Bill properties 20

Figure 20: Graphic View 21

Figure 21: Table View 22

Figure 22: Column View 22

Figure 23: Pie View 23

Figure 25: Settings of Reports 23

Figure 26: Quick Summary 24

Figure 27: Search Settings **Error! Bookmark not defined.**

Figure 28: Inserting time frame 25

Figure 29: Search Results 25

Figure 30: Settings 26

Figure 31: Writing Review 27

Figure 32: Contact us 27

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| *Date* | *Version* | *Description* | *Author* |
| 2012-02-20 | 0.1 | Created Draft Version | [Gohar Hovhannisyan](mailto:gohar.hovhannisyan@virtual-solution.com?subject=Concerning%20SRS%20of%20Budget%20Tracker%20project) |
| 2012-02-27 | 0.1 | Added 5.8, 5.9 and 5.13 sections | [Arevik Tunyan](mailto:arevik.tunyan@virtual-solution.com?subject=Concerning%20SRS%20of%20Budget%20Tracker%20project) |
| 2012-02-28 | 0.1 | Added 5.7 section | [Mihran Dovlatyan](mailto:mihran.dovlatyan@virtual-solution.com?subject=Concerning%20SRS%20of%20Budget%20Tracker%20project) |
| 2012-03-07 | 0.1 | Added 5.10 section | [Gohar Hovhannisyan](mailto:gohar.hovhannisyan@virtual-solution.com?subject=Concerning%20SRS%20of%20Budget%20Tracker%20project) |

# Introduction

## Purpose

The purpose of this document is to describe the requirements specifications for the Budget Tracker project for software development teams.

The intended audience of this document includes the prospective developers of the project and the technical assessment personnel of the client organization.

## Scope

The project is a iOS based application, which means that the app will run only on the iPhone and iPad devices.

***Note****: The first version of the app will include only iPhone version.*

The main goal of the app is to learn and use the new technologies provided iOS SDK.

The app will be used for tracking/monitoring the incomes as well as expenses for the specified time periods. It will help the clients to generate and analyze the reports. The easy to use and intuitive interface will assist the clients to do whatever they want. The project also will have a database where can be kept the user specific information, which is encrypted and stored locally in a device.

The app shall be carefully designed to help users to track daily, monthly, overall balances, dates, type of expense and income automatically expenses.

## Glossary

|  |  |
| --- | --- |
| **SRS** | Software requirements statement |
| **UI** | User interface |
| **App** | Short for application |
|  |  |

## References

* <https://svn.virtual-solution.de/internal/main/BudgetTracker/trunk>

# System Requirements

In the section below is represented all the requirements on the project both functional and non-functional.

As it is notated in the section above, the app is available only for iOS SDK and the supported SDK versions are 3.2 and later.

## General User Interface Requirements

The user interface requirements are the followings:

* UI shall be intuitive and easy to use by clients
* UI shall be implemented using the standard features/components of iOS SDK not to hurt user experience

## Functional Requirements

The app will have the following main functionalities:

1. New user creation and Login
   * In order to enter the application there is need to have an account. App will give an opportunity to register new users as well as login with already created accounts.
2. Quick summary
   * There should be possible to review the summary in the following time frames:
     1. Current day
     2. Current week
     3. Current month
     4. Current year
3. User accounts
   * User should have possibility to add/modify and delete an account. Account can have arbitrary type (e.g. savings, cash, debt and so on).
4. User incomes
   * User should have possibility to add/modify and delete an income. It also should be possible to connect income with any of accounts.
5. User outcomes
   * User should have possibility to add/modify and delete an outcome. It also should be possible to connect outcome with any of accounts.
6. User bills
   * User should have possibility to add/modify and delete a bill. It also should be possible to connect outcome with any of accounts.
7. Reporting
   * The reporting should be available in the following forms:
     1. Table view
     2. Pie view
     3. Graphic view
     4. Column view
   * It should be also possible to select the time frame for which the reporting is done.
8. Searching
   * It should be possible to perform search with specified criteria. E.g. one can perform search by category name, or outcome name.
9. Settings
   * All the user specific settings shall be gathered and easy reconfigurable. User should be able to easily configure his/her own view.
10. Contact information
    * Contact information should be available.
11. Writing feedback
    * An email should be opened that user can write feedback to the development team.
12. Writing review in a apple store
13. Impressum

## Technical Requirements

As the implementation of the app will be done for training purposes, the use of the following technologies is mandatory:

* iCloud
* iAd
* Core Date
* Core Graphics

# System Scope

# Business Process and Data Definition

The section will be empty for this project as it has training purpose and at this point no business process is defined.

# Functional Decomposition

## System overview

The whole app is implemented considering the following main functional units:

* **User**
  + Someone who wants to use the app.
* **Account**
  + Any kind of account for user, e.g. cash, savings, debt and so on.
* **Income**
  + The monetary payment received for goods or services, or from other sources, as rents or investments.
* **Outcome**
  + Any amount of money that user expenses
* **Bill**
  + A statement of money owed for good or services supplied. From the app point of view, the bills are especially recurring expenses (such as utility).
* **Category**
  + A category associated wit the outcome and/or bill, specifies the type of that items.

We’ll call beans the above-mentioned functional units for the rest part of the document.

The whole functionality of the app is the actions on the above-mentioned beans and their properties.

## Module: Registering new user

As the app is user oriented, there is need to register a user to be able to use the app. There should be separate page for registering user.

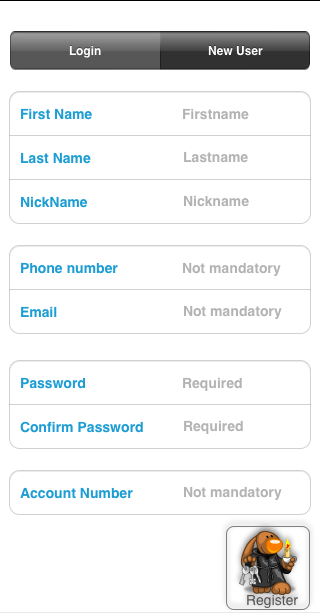


Figure 1: Registration page

It should be possible to create several users and the user profiles will be stored in database. The user should be able to easily differentiate mandatory and not mandatory fields for registration.

## Module: Login/Logout

The login/logout functionality should be available for each user. There shall be possible to enter nickname and password and login into the app and vice versa logout one user and login another one. The page as presented below shall be available:

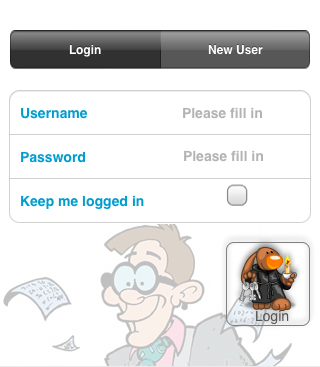


Figure 2: Login User

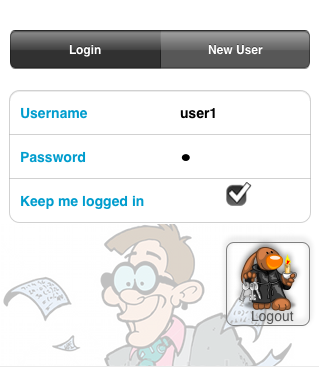


Figure 3: Logout User

## Module: Adding/Editing accounts

As soon as user enters the app, user shall be able to add the accounts, delete them and watch the list of added accounts with their properties. It also should be possible to view the details of each account and the total count of incomes/outcomes associated with it.

Below is represented the pages for each action with account:

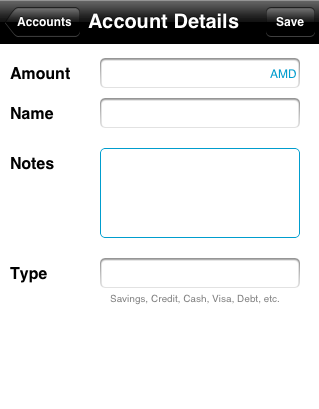


Figure 4: Account properties

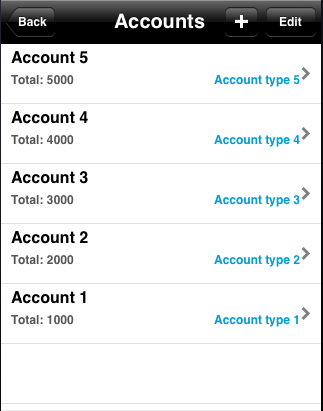


Figure 5: List of Accounts

It shall be possible to delete any account from list. The figure below represents the account deletion.

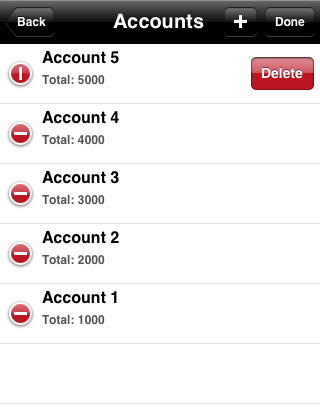


Figure 6: Editing Account

The information on account details (such as total amount of incomes, bills and outcomes associated wit that account, and current balance) shall be presented in a separate view. It shall be possible to easily attach income/outcome and bill to the current account.

The important information shall be correspondingly highlighted. E.g. when balance is negative, it should be highlighted in red, and vice versa in green.

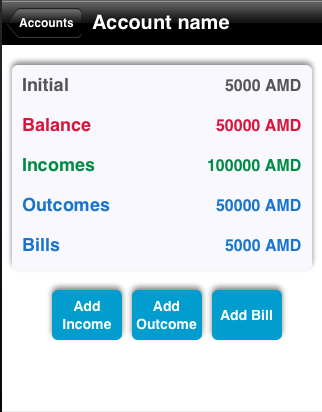


Figure 7: Account details

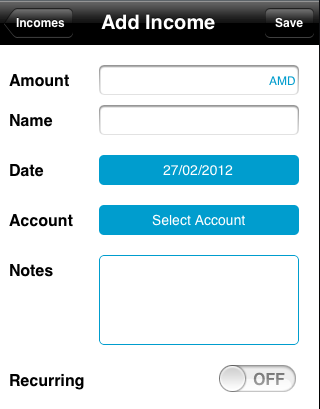
## Module: Adding/Editing Incomes

User shall be able to add incomes and connect it with already created accounts by filling in the predefined fields. The added incomes with the most important properties shall be shown in a list sorted by months.

E.g User received 20.000,00 AMD salary on February 2nd and added it to the Account 1.

In the list view, it should be possible to navigate into the months.

The figure below shows how to add a unique income.



As soon as an income is added, it should be visible in a list, under the corresponding month.

Figure 8: Adding Income



Figure 9: List of Incomes

There should be navigation through months in the list that user can see the incomes for each month.

There shall be possible to delete income from the list and navigate to the income update page and update any property of income.

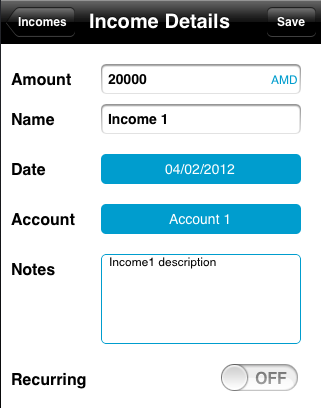


Figure 10: Income Details

It should be possible to add recurring income (such as salary).

## Module: Adding/Editing Outcomes

It shall be possible to add new outcome and attach it to the account. It should be also possible to edit the created outcome and update any property of outcome. The list of outcomes shall be shown with the most important properties.



Figure 11: List of Outcomes

As for incomes list, here also should be navigation through months in outcomes list that user can see the outcomes for each month.

The outcome details can be seen in a separate page, where user can easily add/modify any property. The figure below illustrates the outcome details page.

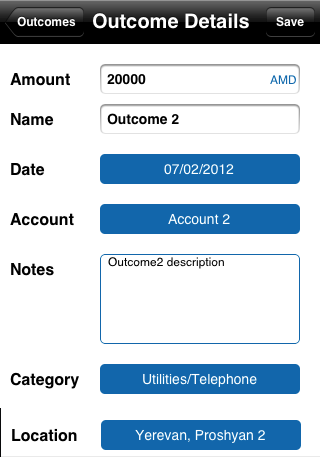


Figure 12: Outcome properties

As outcome has a category property, user shall be able to choose a category from the predefined list as well as create his/her own category. The figure below represents the categories List.

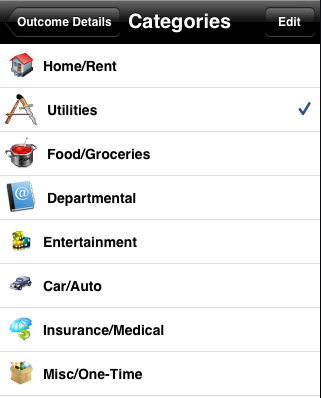


Figure 13: List of categories

There shall be possible to delete category and reorder the list of categories. The figure below illustrates the editing mode of categories list:

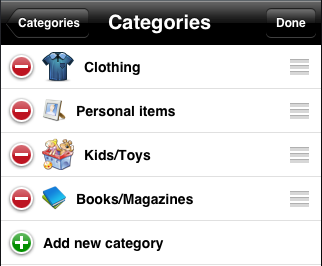


Figure 14: Editing Categories

There shall be possible to add new category and attach icon to it.

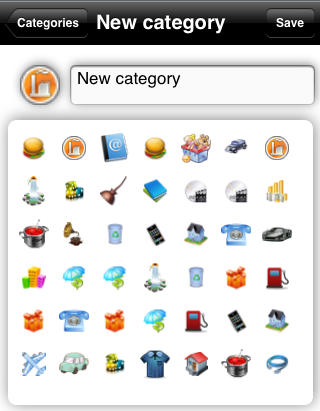


Figure 15: Adding category

The user shall have possibility to attach the location to the outcome. For filling in the address where that expense has taken place user shall be able to open map view, choose the corresponding address and then save it. After all the chosen address shall be shown in outcome details page.

The figure below illustrates the map view, by default the current location is opened.

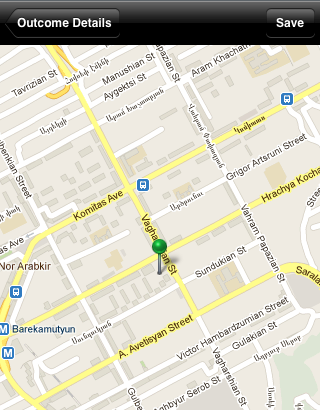


Figure 16: Choosing location

## Module: Adding/Editing Bills

It shall be possible to add new bill and attach it to the account. It should be also possible to edit the created bill and update any property of it. 2 Views shall be available for displaying the bills:

* List of bills
* Calendar

The list of bills shall be shown with the most important properties.

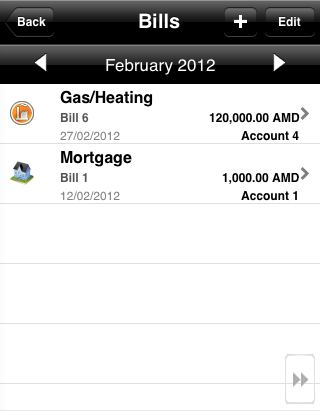


Figure 17: List of Outcomes

As for outcomes list, here also should be navigation through months in bills list that user can see the bills for each month.

In the calendar view should be marked the bills for each day.

The figure below illustrates the basic calendar view:



Figure 18: Calendar view

The bill details can be seen in a separate page, where user can easily add/modify any property.

The bill shall have the same properties as outcome, please refer to the Figure 12: Outcome properties for more details. And in addition to the outcome properties, bill shall have some more properties, which are illustrated in the figure below:

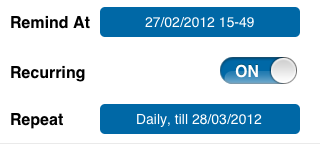


Figure 19: Bill properties

User should be able to create recurring bill and define periodicity for it. It should be also possible to setup reminder for it.

## Module: Watching/Analyzing Reports

There should be button on home page in the application tapping on user can see transaction reports. The application should analyse incomes and expenses, taking into account some criteria (e.g. categories - how much was spent for each category) and generate graphical views. There should be several types of reports. E.g. user should be able to see relative expenses for each category.

The following reports shall be available:

* Graphic view
* Table View
* Column View
* Pie View

There should be legends in graphical views bringing on details about that graphic.

The user should be able to choose from the report settings that report he/she wants to and for which time frame. The following time frames shall be available:

* 3 months
* 6 months
* 12 months

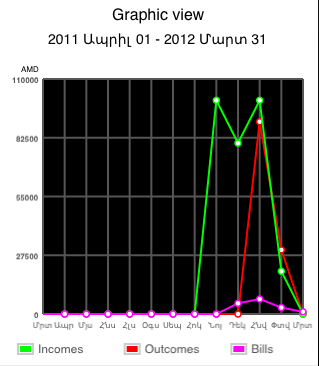


Figure 20: Graphic View

Table view – should present generated report in table view mode. The moth field should show for how much months (3, 6, 12) was report generated. The balance field should show the difference between incomes and expenses.

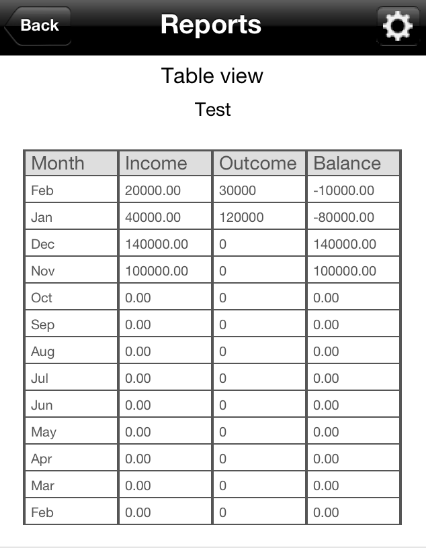


Figure 21: Table View

The view below represents generated report (only incomes and outcomes) in column view.

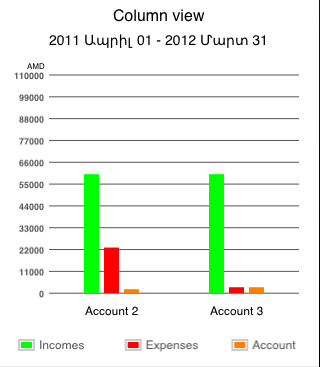


Figure 22: Column View

pie view – this view should be intended only for displaying categories report in pie view mode

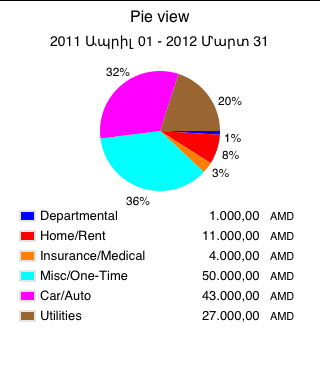


Figure 23: Pie View



Figure 24: Settings of Reports

## Module: Reviewing quick summary

There should be button in the application home screen, which should represent the quick summary of user transactions in some time frames. The quick summary page should display automatically generated summary from the incomes and expanses (outcomes, bills) for day, week, month and year time frames.



Figure 25: Quick Summary

## Module: Performing search by specified criteria

It should be possible to perform search in beans (except Category) and beans properties. It also should be possible to perform search by specified time period.

The figure below represents the view of the settings for advanced search:

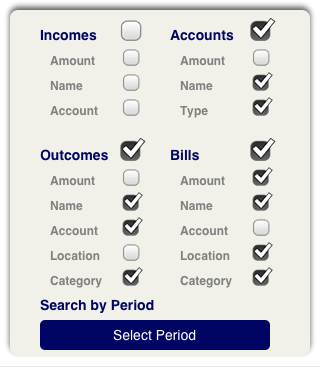


Figure 26: Search Settings

It should be possible to filter search result according to above-mentioned properties. E.g. if user unticks Income item, the search will be done only for Accounts, Outcomes and Bills items.

User should be able to select period for performing search. The figure below represents basic view for selecting time frame:



Figure 27: Inserting time frame

Finally, the search results should be available in a list, as illustrated in the figure below:

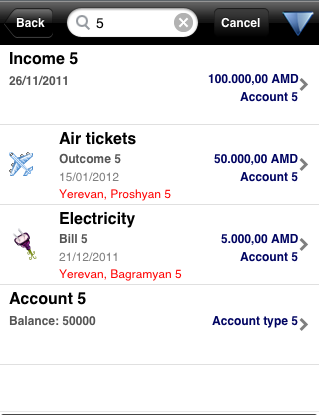


Figure 28: Search Results

Each item should have the same look&feel as in separate list. E.g. in the figure above is represented the search result by “5” keyword and there are 1 income, 1 outcome, 1 bill and 1 account. Each of the items has the same view as in the list of those items.

## Module: Changing the app settings

Once user enters the app, he/she should be able to define his own settings and/or change them. User can have both functional and interface specific settings. The chosen settings shall be saved in a database.

The following figure illustrates the basic view of settings page:



Figure 29: Settings

User shall be able to define the following settings:

* Start screen for the app
* The main colour for the app
* Month start date
* Currency for the transactions
* How long save the added data (incomes, outcomes, bills, accounts)
* Backup/Restore the user specific info in/from iCloud
* Fully reset the app by removing the user and associated info

## Module: Writing review in AppStore

There shall be button on home page for redirecting to the application info page in appStore, that user can rate the app and write a review. The following page shall be opened for writing review for app:

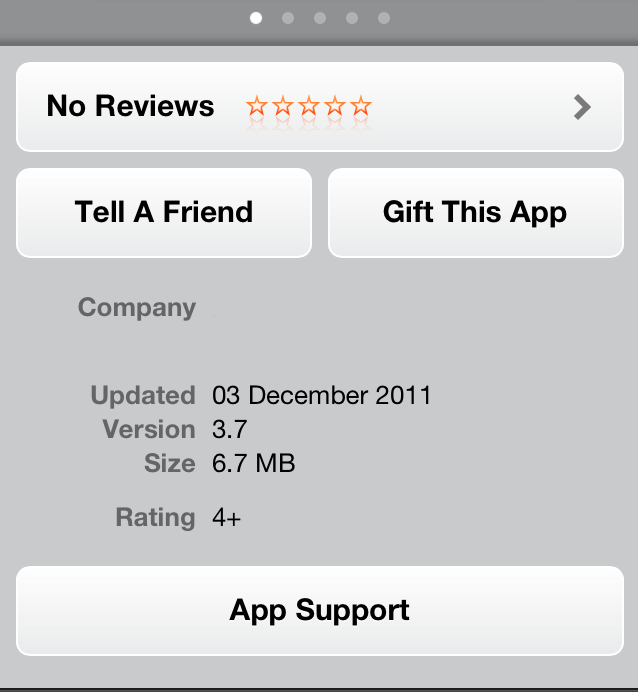


Figure 30: Writing Review

## Module: Writing feedback

In the home page should be button, which should allow user to write mail from registered mail to the development team. Taping on that button application should redirect user to the standard mail page.

## Module: Watching contact information

The contact information shall be available for the project, bringing up the company info such as address, tel. number, fax and email.

The following figure illustrates the contact information page:



Figure 31: Contact us

# Operational Features

The following non-functional requirements shall be supported:

* There shall be help and “how to use app” information
* No installation shall be necessary on end user side
* The development shall be done using XCode tool and the implementation shall be analyzed via “Static Analyzer Tool”.
* An easy understandable error messages shall be displayed to help the user to understand what is wrong and how to correct it.

## Performance

All the pages shall be loaded as fast as possible. The response time, utilization and throughput time shall be as little as possible.

The implementation shall be done as effective as possible.

The app shall use as little memory as possible.

## Implementation/Deployment

The implementation will require the following environments:

* The implementation shall be done in Objective C
* Xcode
* Testing environment

# Business Impacts

# Assumptions and Dependencies

* All the development workstations shall have the following tools installed:
  + Mac osX
  + Xcode
  + Any client of SVN

# Software Quality Attributes

The following attributes shall be taken into account when the app is tested:

* **Functionality**: are the required functions available, including interoperability and security
  + Completeness
  + Correctness
  + Security
  + Compatibility
  + Interoperability
* Efficiency
  + Time economy
  + Resource economy
* Reliability
  + Non-deficiency
  + Error tolerance
  + Availability
* Maintainability
  + Correctability
  + Expandability
  + Testability
* Usability
  + Understandability
  + Ease of learning
  + Operability
  + Comunicativeness

# Appendix