[draft] Software Requirements Specification

for

Metro Calendar

Version 1.0.0

Prepared by Antares

Group Name: Antares

|  |  |  |
| --- | --- | --- |
|  |  |  |
| |  |  |  | | --- | --- | --- | | Le Hoang Hung | 01011 | Project Manager | | Vu Thanh Binh | 00919 | Developer | | Truong Trung Nghia | 01300 | Developer | | Le Xuan Nghia | 01322 | Designer | | Tran Thai Son | 00845 | Tester | |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| Instructor: | La Ngoc Quang |
| Specialties: | Information System |
| Date: | August 16, 2012 |

Table of Contents

[1. Overall Description 5](#_Toc333356496)

[1.1. Product Perspective 5](#_Toc333356497)

[1.2. Product Functionality 13](#_Toc333356498)

[1.3. Operating Environment 13](#_Toc333356499)

[2. Specific Requirements 14](#_Toc333356500)

[2.1. External Interface Requirements 14](#_Toc333356501)

[2.1.1. User Interfaces 14](#_Toc333356502)

[2.1.2. Software Interfaces 14](#_Toc333356503)

[2.1.3. Communications Interfaces 14](#_Toc333356504)

[2.2. Functional Requirements 14](#_Toc333356505)

[2.3. Behaviour Requirements 15](#_Toc333356506)

[2.3.2. Use Case View 15](#_Toc333356507)

[3. Other Requirements 15](#_Toc333356508)

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.0.0 | Hung Le | Create first draft of the document | 08/17/2012 |
| 1.0.1 | Hung Le | Update Product perspective | 08/20/2012 |
| 1.0.2 | Hung Le | Update Product functionality, OS environment | 08/22/2012 |
| \* |  |  |  |

# Overall Description

## 1.1. Product Perspective

Metro calendar is an application aim to help people to manage their task. The basic function of it are create new task, edit and delete them also. For the first phase, this application will be able to import, export task to file, share them through email. Authenticate and synchronize data with server will be implemented in phase 2. When completed, the application will have follow functions:

* **Display calendar:**

Content is the heart of Metro style apps and putting content before chrome is fundamental to the design of Metro style apps. Everything else is accessory—or chrome—that helps present and enables interaction with the content. There will be 2 possible task-calendar display types:

* + Grid view: Display calendar by month with a table. It provides an overall view of the calendar and the task.
  + Timeline: Display calendar by week or day on a line. It takes advantage of the UI in Metro style apps and brings user an visualization view of the calendar. Time line view by day can be used to add task.
* **Core function:**

There are core functions of Metro calendar:

* Add task.
* Edit task.
* Delete task.
* Reminder.
* Import/export.
* Share task.
* Authenticate.
* Synchronize data.
* Localization.
* **Other feature:**

Metro style apps are the focal point of the user experience on Windows 8, and great Metro style apps share an important set of traits that provide a consistent, elegant, and compelling user experience. The entire feature below may never appear on previous build of Windows. That the point Metro calendar should focus to:

* **Snap view.**

Metro calendar must design for view states. People naturally multitask, and they can run two apps side by side through snapping. All apps must implement a snapped state, and a great snapped state invites users to keep your app on screen longer.

* Design a useful Snap view that provides a functional view of the app. Preserve users' context when they switch between view states.
* When snapped, the width of the app is fixed at 320 pixels but the height is variable. Pan vertically to avoid conflict with the edge and snap gutter.
* Other apps can be snapped as well when your app is on screen, so design your app to reflow fluidly from a minimum width of 1024 pixels.

Snap view sample from **Weather** app:

|  |  |
| --- | --- |
| Filled state: | Snapped state (1/4): |
|  |  |

* **Semantic zoom view**.

Semantic Zoom is a touch-optimized technique used by Metro style apps in Windows 8 for presenting and navigating large sets of related data or content within a single view.

Semantic Zoom uses two distinct modes of classification (or zoom levels) for organizing and presenting the content: one low-level (or zoomed in) mode that is typically used to display items in a flat, all-up structure and another, high-level (or zoomed out) mode that displays items in groups and enables a user to quickly navigate and browse through the content.

The Semantic Zoom interaction is performed with the pinch and stretch gestures (moving the fingers farther apart zooms in and moving them closer together zooms out), or by holding the Ctrl key down while scrolling the mouse scroll wheel, or by holding the Ctrl key down (with the Shift key, if no numeric keypad is available) and pressing the plus (+) or minus (-) key.

Semantic sample from **Store** app:

|  |  |
| --- | --- |
| Semantic zoom in | Semantic zoom out: |
|  |  |

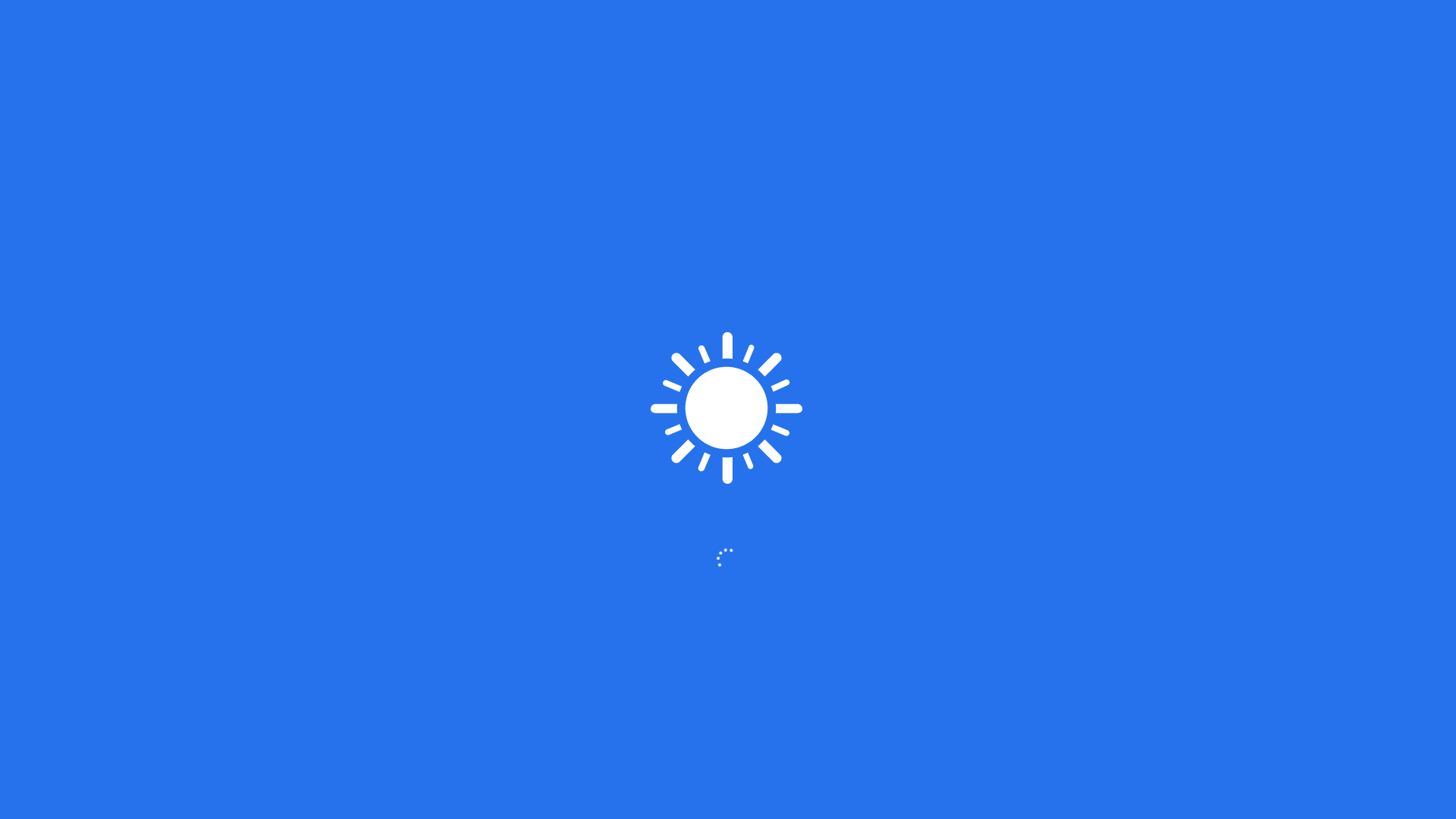
* **Splash screen**

All Metro style apps must have a splash screen, which is a composite of a splash screen image and a background color, both of which can be customized.

Windows displays this splash screen immediately when the user launches an app. This provides immediate feedback to users while app resources are initialized. As soon as the application is ready for interaction, Windows dismisses the splash screen.

A well designed splash screen can make Metro calendar more inviting.

Sample from Weather app:



* **Live tile**

Draw people in continually with dynamic, relevant, personalized content on Metro calendar’s tile. Fresh tile content gives people a reason to place the tile in a prominent position on the Start screen, and to launch the app time and again.

* + A live tile can cycle through up to 5 updates. For example, a news app's live tile can cycle through multiple stories each day.
  + Content shown in live updates should be accessible from the home page of Metro calendar. Remove outdated notifications if they are no longer relevant or accessible from the home page.
  + Use badges to show simple numeric or glyph information.

|  |  |
| --- | --- |
| Sample from Store app: | Sample from Sports app: |
|  |  |

* **App bar**

App bars provide the user with easy access to commands when they need them. The user can swipe the bottom edge of the screen to make app bars appear and can interact with their content to make app bars disappear. App bars can also be used to show commands or options that are specific to the user's context, such as photo selection or drawing mode.

If you have a command that is necessary for a user to complete a workflow (such as buying a product), place those commands on the canvas instead of in app bars.

Do place commands consistently, and organize them into command sets.

1. Start with your commands on the right.
2. If you have distinct sets of commands (such as a set for creating new content, and a set for filtering your view), then divide those sets up on either side of the screen.
3. If you have more than two sets, then put the command sets most like each other on the same side, separated by a separator.
4. Ensure that commands always show in the same relative position, and on the same side of the screen, whenever they appear inside of your app.

App bar sample from Metro IE app:



* **Settings Charm**

The Settings charm provides a single access point to all settings that are relevant in the user's current context. The Settings charm is always available. Regardless of context, settings include system settings that always apply, system-brokered settings that enable users to control an app's access to system devices and capabilities, and settings that are specified by the current app.

You swipe from the side of the screen to display the charms. Press the Settings charm to display the settings window. The settings window includes both app and system settings.

The app may provide Settings Command entry points, which appear at the top of the settings window.

Sample from Weather app:



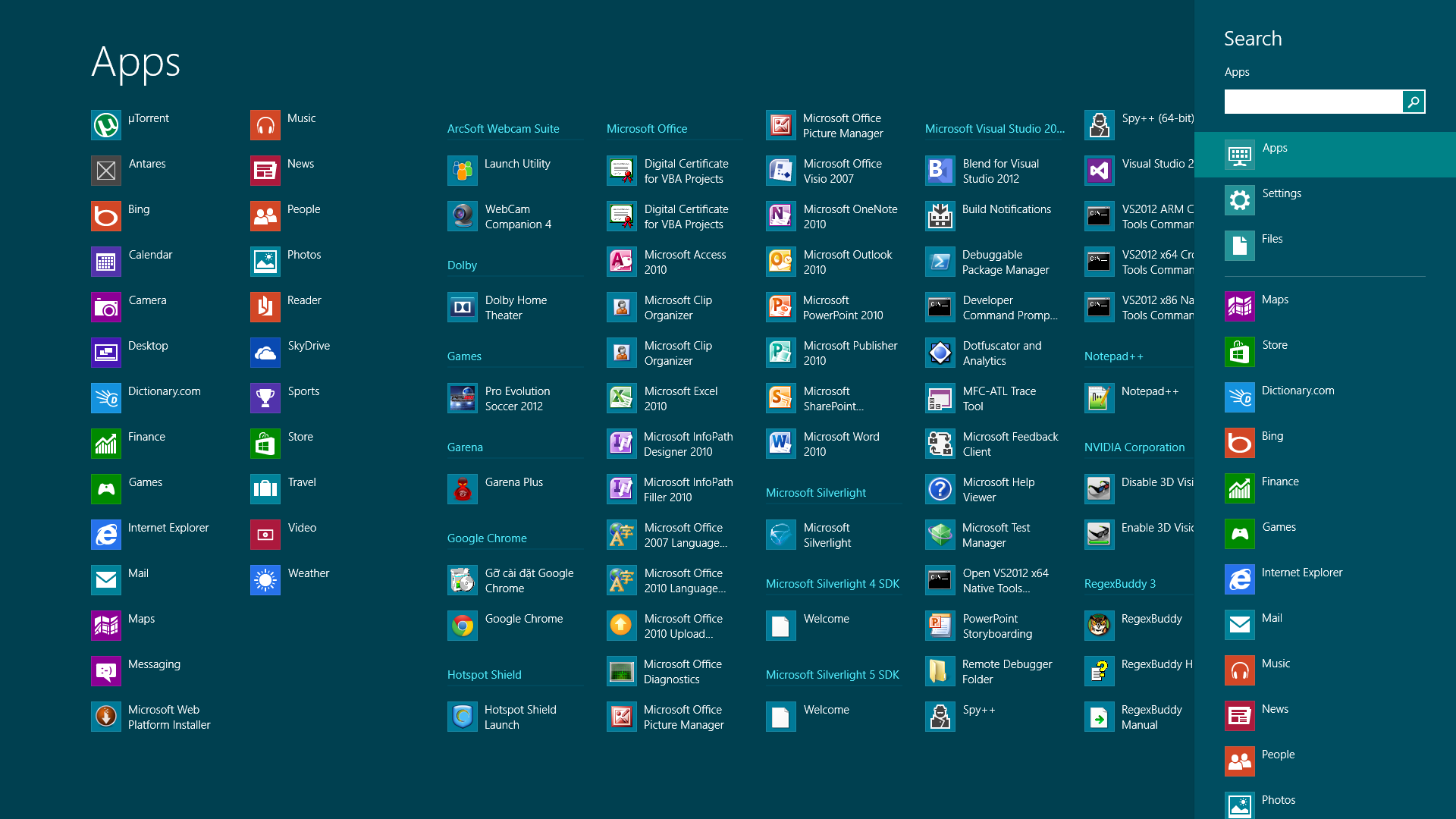
* **Search contract**

Let users search with Metro calendar from anywhere in their system, including from its self. Users will be able to use the Search charm to open a search pane where they can enter search queries and select your app to display their search results. Application must participate in the Search contract. The search pane that is opened by the Search charm consists of:

* + - A touch-friendly search box
    - Touch-friendly search suggestions
    - An integrated IME for east Asian languages
    - A list of apps that user can get search results from

You can use shortcut keys to access charms, including the Search charm. The Windows Logo Key + C lets you select any of the charms, and the Windows Logo Key + Q selects the Search charm.

This is the search contract:



* **Lock screen**

The lock screen is designed to show information that a user can take in with a glance. Users should not have to dwell on this screen to receive information. The best app candidates to be placed on the lock screen can convey understandable, context-free, summary information through only a badge and a logo icon. For example, a mail app that shows a badge with a count of unread messages would work well on the lock screen.



## 1.2. Product Functionality

The major functions of Metro calendar are listed as below:

|  |  |  |
| --- | --- | --- |
|  | **Function** | **Description** |
| Phase 1 ~8 weeks | **Add schedule** | User can easily create a new task with title, description, start time, end time, type, reminder etc… After this task's been added, it will be present on timeline view or basic view. |
| **Edit schedule** | Allow user to modify existing task. |
| **Delete schedule** | Allow user to delete their task. |
| **Localization** | Metro calendar has capability to display on 3 languages: Vietnamese, English, Japanese |
| **Snapped state** | Develop snap function which allows user to view Metro calendar in snapped state (1/4  screen). |
| **Search** | Integrated with search contract of Windows. Allow user to search all the schedule in  Metro calendar. The result can be filtered by type, time… etc. |
| **Live tile** | Implement live tile to display the incoming task. |
| **Notification** | Notify when the task has started on both on screen notification and lock screen  notification. Time to display notification can be modified if user wants to notify before a period of time… |
| **Log to metric** | Store behavior of user to use later. Data will be stored cumulative. |
| **Intelligent calendar** | Metro calendar can provide some intelligent at:   * Base on the metric, it can predict the behavior of users then give them suggestion. * Optimized schedule on task with no started date. * Suggest task, give advice for user when they create/edit task base on some related task or tasks has close time to it. |
| **Import/Export schedule** | Allow user to import, export their task to other computer. There're potential some  option to import, export, such as: in period of time, type… etc. |
| Phase 2 ~4 weeks | **Share** | Provide a mechanism to share task online, mail is a tentative option. |
| **Authentication** | Provide a mechanism to authenticated when login to Metro calendar.  There are few potential options: Using Google account, Windows Live ID or  Authentication broker. |
| **Synchronized data** | Synchronized data with cloud server. Metro calendar may use the current authenticated account to store data to its server (by supplied API). |

Screen flow:



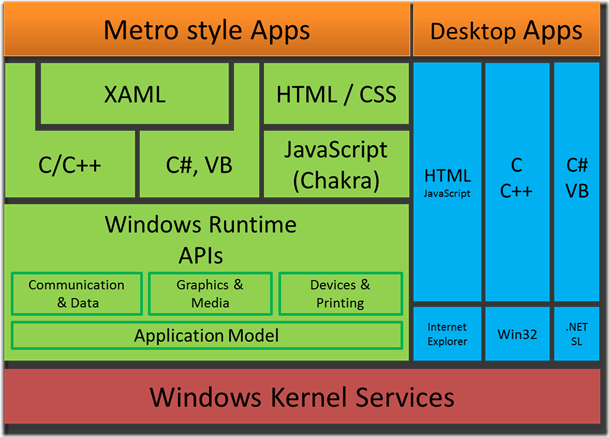
## 1.3. Operating Environment

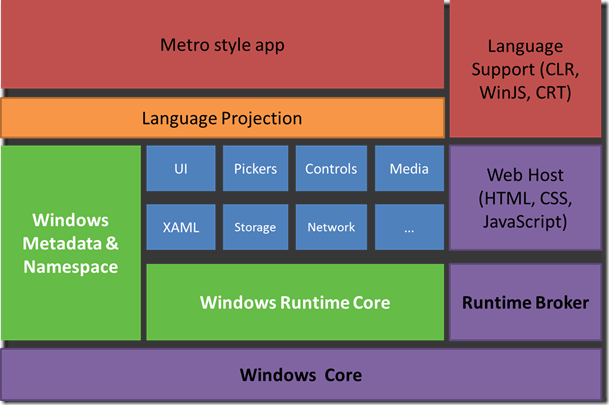
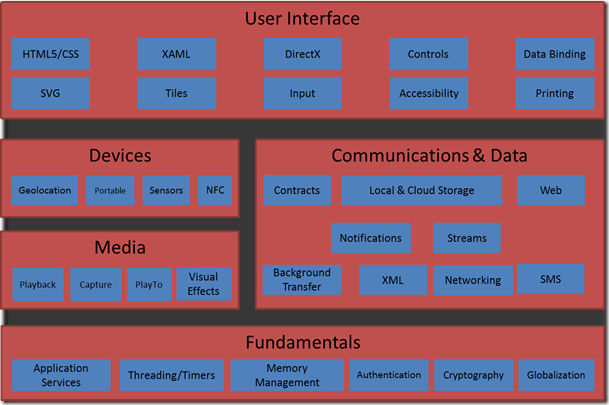
Metro Calendar will be run on Metro environment. Windows 8 employs a new user interface based on Microsoft's Metro design language. The Metro environment features a new tile-based Start screen similar to that of the Windows Phone operating system. Each tile represents an application, and can display relevant information such as the number of unread messages for an e-mail application or the current temperature on a weather application. These applications run in full-screen or in docked modes, and are able to share information between each other using "contracts". They will be available only through Windows Store. Apps in the new interface are developed with the new Windows Runtime platform using various programming languages, including C++, Visual Basic, C#, and HTML with JavaScript code-behind.

The traditional desktop environment for running desktop applications is accessed via a tile on the new Start screen. The Start button has been removed from the taskbar in favor of a Start button on the new charm bar, as well as a hotspot in the bottom-left corner. Both open the new Start screen, which replaces the Start menu.

Applications developed for this new environment were previously referred to as "Metro-style apps" in development materials, connecting it to Windows 8's use of a interface following the Metro design language as its primary desktop. However, reports surfaced that due to potential trademark issues with the German company Metro AG, Microsoft officials had begun to advise its Windows developers to stop using the term. A Microsoft spokesperson however, denied these reports and stated that the use of the term "Metro" to describe these apps was merely a codename that would be phased out. Following these reports, Microsoft started using "Modern UI Style" to refer to its tile-based interface and design language.

The WinRT is described as below:



# Specific Requirements

## 2.1. External Interface Requirements

### User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., Cancel) that will appear on every screen, error message display standards, and so on. Define the software components for which a user interface is needed.

TO DO: The least you can do for this section is to describe in words the different User Interfaces and the different screens that will be available to the user. Those who will be able to provide optional Graphical User Interface screenshots, will be rewarded by extra marks.>

### Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems (Windows? Linux? Etc…), tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

TO DO: The previous part illustrates some of the information you would usually include in this part of the SRS document. To make things simpler, you are only required to describe the specific interface with the operating system.>

### Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

TO DO: Do not go into too much detail, but provide 1-2 paragraphs were you will outline the major communication standards. For example, if you decide to use encryption there is no need to specify the exact encryption standards, but rather, specify the fact that the data will be encrypted and name what standards you consider using. >

## Functional Requirements

*< Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. This section is the direct continuation of section 2.2 where you have specified the general functional requirements. Here, you should list in detail the different product functions with specific explanations regarding every function.*

*TO DO: Brake the functional requirements to several functional areas and divide this section into subsections accordingly. Provide a detailed list of all product operations related to these functional areas.*

## Behaviour Requirements

### Use Case View

<A use case defines a goal-oriented set of interactions between external actors and the system under consideration. Since sometimes we will not be able to specify completely the behaviour of the system by just State Diagrams, we use use-cases to complete what we have already started in section 3.3.1.

TO DO: Provide a use case diagram which will encapsulate the entire system and all possible actors. Do not include detailed use case descriptions (these will be needed when you will be working on the Test Plan), but make sure to include a short description of what every use-case is, who are the actors in your diagram. For more information please refer to your UML guide and the MiniThermostat SRS example file.>

# Other Requirements

<This section is **Optional.** Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>