Software Requirements Specification

for

myPomodoro

Version 1.0 approved

Prepared by P. Barton, R. Forscht, B. Wetzel, and J. Smith

ITEC 4860

1.28.2010

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

\*\*Revision History for this document will be handled [online](http://code.google.com/p/mypomodoro/) using Subversion\*\*

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| N/A |  |  |  |
| N/A |  |  |  |

# Introduction

## Purpose

The purpose of this project is to develop Time Management Software that follows the Pomodoro Technique. The overall purpose is to develop software to allow a user to manage their time by focusing on tasks or activities in order to increase return.

This software, myPomodoro, will be a Time Management Tool based upon the time management technique called The Pomodoro Technique. The objectives of this software are to automate The Pomodoro Technique, which are otherwise preformed manually. The goal of this software is to allow an end user to manage their time by focusing on tasks and activities in order to increase returns, save time, work smarter not harder, and reduce stress through the automation of The Pomodoro Technique. This tool will follow a Keep It Simple, Stupid (KISS) design principle, which will maximize efficiency, task completion, and meet the needs of the end user while remaining easy to use and understand.

## Document Conventions

* IEEE Guide to Software Requirements Specifications (Std 830-1993)
* Font used: Arial, Font Size: 11
* Comments will be encapsulated by two carrots and italic (<*Comment*>). Comments should be removed before final version
* Anything listed in a red font is new since previous version
* Anything listed in a gray font is planned for deletion

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

## Intended Audience and Reading Suggestions

This document is intended for all stakeholders and developers of the myPomodoro Time Management Software and will be proposed to The Pomodoro Technique Timer Group on Google Groups for approval.

Before reading further into this document, it is suggested that you read either [The Pomodoro Technique](http://www.pomodorotechnique.com/resources/cirillo/ThePomodoroTechnique_v1-3.pdf) by Francesco Cirillo or [The Pomodoro Technique Cheat Sheet](http://www.pomodorotechnique.com/downloads/pomodoro_cheat_sheet.pdf) on the Pomodoro Technique website in order to better understand the time management technique this software will be based upon.

This document is organized below, in order, by providing the Scope and referenced documents of the software, an Overview of the software, types of Interfaces, Functional Requirements, Non-Functional Requirements, Glossary, UML Diagrams, and a To be Determined (TBD) List.

It is suggested for stakeholders to read the Scope, the Overview of the software, Functional and Non-Functional Requirements, and to refer to the Glossary if one has any questions concerning definitions, acronyms, and abbreviations.

It is suggested for developers to read the Overview, Functional and Non-Functional Requirements, Interfaces, UML diagrams, and TBD List.

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

## Product Scope

The development of this software will be divided into two phases, whereby the first phase will include the automation of The Pomodoro Technique and the second phase will include extensions typically seen in traditional time tracking software in order to provide import/export capabilities, report generation, etc… This document only covers the SRS for the first phase.

## Objectives

More specifically, this software will be designed to allow an end user to place all activities/tasks on an Activity Sheet, generate daily ToDo Sheets from the Activtity Sheet, prioritize tasks in a ToDo sheet, start/stop working on a daily ToDo Sheet, combine tasks on ToDo sheet into one Pomodoro (work increment of 25 minutes), break down tasks on a ToDo Sheet into multiple Pomodoro’s, adjust timer for breaks within the end user constraints, and provide feedback for interuptions to the ToDo Sheet and Activity Sheet. This software will be designed to provide a timer for each Pomodoro, each short break (3-5 minutes), and each long break (15-30 minutes); provide an Activity Sheet, provide ToDo Sheets, and provide a Records Sheet. This software will include a local relational database containing all the necessary relations and attributes as described in the Functional Requirements below.

The software will allow the user to access the following functions:

* Create a Timer - Bobby
* A to do list that is available across sessions - Brian
* Activity Sheet must be able to be updated and prioritized - Brian
* Backlog of Activities - Brian
* Notification at end of timer - Jordan
* Break counter - Bobby
* Way to update and make changes to activity sheets – Jordan
* Combine tasks on ToDo sheet into one Pomodoro - Paul
* Break down tasks on a ToDo Sheet into multiple Pomodoro’s - Paul
* Provide a Records Sheet – Jordan

Initially the application needs to have a GUI with an intuitive "look and feel"; furthermore, it should include a counter mechanism for timing the "pomodoros", and include a way to create/modify an activity list as activities are accomplished through the day.

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

## References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

Cirillo, Francesco. (2006). *The Pomodoro Technique* [v1-3]. (Adobe PDF), Retrieved from <http://www.pomodorotechnique.com/resources/cirillo/ThePomodoroTechnique_v1-3.pdf>

The Pomodoro Technique (2010). *The Pomodoro Technique Cheat Sheet*. Retrieved from <http://www.pomodorotechnique.com/downloads/pomodoro_cheat_sheet.pdf>

Barton, P., Forscht, R., Smith, J., & Wetzel, B. (2010). *myPomodoro Time Management Software Proposal*. Geogria Gwinnett College. ITEC 4860. Spring Semester 2010. Retreived from <http://code.google.com/p/mypomodoro/source/browse/trunk/myPomodoro%20TM%20Project%20Proposal.docx>

Wiegers, K.E. (1998). *IEEE Software Requirements Specification Template* (Microsoft Word), Retrieved from [www.csc.villanova.edu/~tway/courses/csc4181/srs\_template-1.doc](file:///C:\Documents%20and%20Settings\Dad\Local%20Settings\Temp\www.csc.villanova.edu\~tway\courses\csc4181\srs_template-1.doc)

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

# Overall Description

## Product Perspective

The application will provide an easy to use interface that is analogous to manual method of managing time prescribed by Francisco Cirillo in his book, [*The Pomodoro Technique*](http://www.pomodorotechnique.com/resources.html).  In addition to emulating those manual tasks of recording time, this application will include additional features for tracking time and activities and storing them in a database.  This data can be used for risk assessments, budgeting, employee evaluations, calculating employee wages, etc.  Making the software open source will allow for the development of more features to fit individual needs, thus making the overall application more robust and usable.  The software should be considered intuitive to use by non-technical personnel, and easily administered by non-technical project managers or individual users.  The data should be exportable, meaning it should be accessible in a variety of forms, including but not limited to, print and web forms.  This software needs to be able to run in a variety of systems, which could include Windows, Mac, Linux, Mobile, etc.

## Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

## User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

## Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

## User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

# 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., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, 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. Refer to documents that describe detailed application programming interface protocols. 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.>

## 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.>

# System Feature

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## Create a Timer

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Create a to do list that is available across sessions

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Activity Sheet must be able to be updated and prioritized

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Backlog of Activities

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Notification at end of timer

4.1.1 Description and Priority

At the end of a Pomodoro, the user must be notified that the Pomodoro has ended so that the user can begin their break. This feature is of High importance: without this notification, the software will be useless. Luckily, it should be straightforward to implement and the implementation carries low risk, as there are many ways to successfully implement this feature.

4.1.2 Stimulus/Response Sequences

This feature is not triggered by user action. It is triggered by the timer feature (see 4.1) . When triggered by the expiration of the current Pomodoro timer, the system will play an audible tone and visually notify the user (through a suitable operating system feature such as a Windows taskbar notification). The user will be expected to manually restore the program and input metrics about the newly completed Pomodoro (preferable the program interface has changed so that these input fields are highlighted, perhaps in a modal dialog?). Once this data is input, the user will manually close the previous Pomodoro, starting the break (see 4.6) or ending their work session.

4.1.3 Functional Requirements

This feature depends on being successfully invoked by the Pomodoro timer. Additionally, appropriate notification facilities must be available. If the volume is muted or it is impossible for the Pomodoro application to use visual notifications (or bring itself to the front), this program will fail to successfully alert the user. Responses to user interactions are not time-critical but should not be noticeably delayed.

## Break counter

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Way to update and make changes to activity sheets

*<This appears to be a duplicate of feature 4.3… possibly combine? – Jordan>*

## Combine tasks on ToDo sheet into one Pomodoro

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Break down tasks on a ToDo Sheet into multiple Pomodoro’s

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Provide a Records Sheet

4.1.1 Description and Priority

This feature allows to user to permanently store and view completed Pomodoros. The user should be able to view a reverse-chronologically ordered table with one completed Pomodoros per row. It will have at least these columns: date, time, description, number of Pomodoros used and user notes. It is possible that columns could be needed to display type codes, categories or other fields stored by our system. Finally, additional columns could be needed for various estimation metrics.

Those capabilities are a high priority since they are used for three of the stages of the Pomodoro technique. However, it is a low risk component since most of the Pomodoro display capabilities can be re-used from feature 4.2 and 4.4

4.1.2 Stimulus/Response Sequences

The user (while not in the middle of a Pomodoro) will (click button, select menu item, etc) to enter into records view. There must be a way to return home from this view. Other user inputs are not required but may be needed. Specifically, future features might include allowing the user to search, re-order or filter this list.

4.1.3 Functional Requirements

Any user actions that take noticeable time to execute (filtering the archived results, etc) should display some sort of “busy signal” to the user, giving an time estimate for completion if possible.

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# Other Requirements

<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.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

As the scope of the project suggests this project should be broken up into two major parts.  Firstly, a fully functional program must exist by the end of the first half of the semester.  During this time we will learn build a project following the Waterfall SDLC.  Secondly, by the end of the semester we will learn agile programming methods, specifically the *SCRUM* methodology.  We will divide new features to the functional prototype into stories that will be developed over the remainder of the semester.

The potential benefit of this software includes:

* Improving the work environment and company culture by reducing work overload and stress.
* Reducing interruptions, distractions, and delays through providing a work-break-work philosophy.
* Reducing time spent on activities and tasks.
* Time tracking features by simply recording the number of Pomodoro’s, which would reduce the need to record the actual time spent on work (In phase two).
* Importing/exporting from or to another file type for the purposes of reporting (In phase two).

This project will be completed of the course of one semester, and the scope of the project should reflect this.  The first half of the semester will follow a traditional Waterfall SDLC, and the second half will follow agile programming methods.  Therefor, a working foundation of the application should be completed by the end of the first half, and additional features can be included as *stories* in the second half.  This will insure that a fully functional application will have been developed by the end of the semester.

Some additional features to the application may include:

* Method to save/restore activities lists
* Store time tracking/activity log data in a database
* Produce metrics based on time tracking/activity log
* Store user profiles/sessions
* Link documents to actions completed
* Allow administrative access to employee progress
* Allow commenting on Activity logs.
* Allow the establishment of standardized activities.
* .there are many possibilities.
* Task Archive/template