# Wavetable Functional Specification

## Introduction

### Summary

A few sentences summarising the project: what it is, who it is for (customer or internal), is it a bespoke project, a product, a demo.

### Requirements

See requirements documentation for detailed description of the functional requirements this document will implement.

### Existing System

This section should include an explanation of the system we are replacing, even if it’s an old manual system.

What problems does the current system have? Which of these problems do we solve?

What useful functions of the current system will we not provide (Constraints)?

Depending on the depth of analysis required, this section may also describe the root causes of each problem. “Root cause” analysis is a systematic way of uncovering the underlying cause of an identified problem:

“It’s amazing how much people *do* know about the problem behind the problem; it’s just that no-one – by which we usually mean management – had taken the time to *ask* them before. So, *ask* them and then *ask* them *again*.”  
Source: *Managing Software Requirements: A Unified Approach* by Dean Leffingwell, Don Widrig – Chapter 4, “The Five Steps in Problem Analysis”

### Terminology

This section should contain all words or phrases having a special meaning for this project with a clear, concise, unambiguous statement on their meaning.

### References

List any document references with numbers, remembering to include issue numbers and/or dates so that the actual version is identified and refer to them as ref[n] in the rest of the document.

## Functional Description

**Use Cases**  
Most likely these will be kept in a separate document or CASE tool, referenced from the functional specification. Development of the use cases and functional specification should happen in parallel, where information from one feeds the other incrementally.

**Always avoid repetition.** The amount of detail in the rest of the functional specification will depend on the number of use cases that have been written.

Although important, use cases do not capture all functional requirements: this is why we need an encompassing functional specification. The availability of a separate document also discourages use case authors from putting too much detail in the use case (e.g. functional requirements instead of usage scenario text) or the wrong detail (e.g. boundary conditions), which are both common mistakes.

(Note this is a similar approach to the Unified Process “Supplementary Spec” which captures additional detail that should be kept separate from the use case).

Where the functional specification references a use case, always use the unique use case name (e.g. “Perform Order Entry”). Depending on the size of the system being modelled, you might also need to include the package name.

Similarly, if the use case references an item in the functional spec, always use the section and number of the functional item (e.g. “User Community, item 1.2.3.4”). If possible (given the constraints of the word processor or CASE tool being used) provide a hyperlink that takes the reader directly to the referenced item.

**User Community**  
Identification of who the system is aimed at. There may be more than one group of people and each group may have slightly different requirements. Are we providing different functions to fulfil these or not?

These groups of people are normally identified as use case roles (i.e. actors), and the functions assigned to each role as individual use cases. Where this information does not fit into the use case model, it should be captured in the main functional specification instead.

**Error Handling**

How errors should be handled should be stated. Identify the different types and reasons for the classification.

**Help**  
Help will be accessible through the design/implementation documents and comments in code.

**Interface**  
User  
This could be a chapter in its own right if it is a full definition. If it is deferred to the design specification stage, this should be stated.

Software  
We may be interfacing to existing software. This should be stated, e.g. toolkits, back ends of existing packages. State versions. Do interface documents exist?

**Platforms**

We should list which platforms we will be supporting. Name a reference platform or platforms plus appropriate operating system versions.

**Portability**  
Although we may only be supporting one platform initially, we almost certainly will want to be able to port developments to other platforms. This should be stated here.

**Documentation**  
List the documents that will be produced. This could refer to the project plan if that exists and contains such a list, otherwise it should be stated here.