# **1.** **Executive Summary**

|  |  |  |
| --- | --- | --- |
|  | The purpose of the project is to create a device to enable a chef to view the inside of an oven and monitor temperature and timers remotely. The project uses a Raspberry Pi and a range of modules to provide information to the user. The user will be able to view still images, temperature and timers via a browser or an Android app. |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 

# **2.** **Product/Service Description**

In this section, describe the general factors that affect the product and its requirements. This section should contain background information, not state specific requirements (provide the reasons why certain specific requirements are later specified).

## 2.1 Product Context

PI checker the essential product with the basic functionalities of users login, online brodcasting and photos taking for cooker. It can send constantly changed image throw home internet till specific equipment such as TV, computer or mobilephone.

How does this product relate to other products? Is it independent and self-contained? Does it interface with a variety of related systems? Describe these relationships or use a diagram to show the major components of the larger system, interconnections, and external interfaces.

## 2.2 User Characteristics

The user will be anyone who uses an oven but cannot or does not want to stand in front of it in order to monitor the food.

## 2.3 Assumptions

- User has a wireless network available

- Smart phone with Android os

## 

## 2.4 Constraints

- Pi must be in a splash proof box

- Must operate remotely (no cables while operating)

## 2.5 Dependencies

- Requires a constant server connection

# **3.** **Requirements**

- The Pi should send an image every 5 seconds

-

· Describe all system requirements in enough detail for designers to design a system satisfying the requirements and testers to verify that the system satisfies requirements.

· Organize these requirements in a way that works best for your project. See Appendix DAppendix D, Organizing the Requirements for different ways to organize these requirements.

· Describe every input into the system, every output from the system, and every function performed by the system in response to an input or in support of an output. (Specify what functions are to be performed on what data to produce what results at what location for whom.)

· Each requirement should be numbered (or uniquely identifiable) and prioritized.

See the sample requirements in Functional Requirements, and System Interface/Integration, as well as these example priority definitions:

Priority Definitions

The following definitions are intended as a guideline to prioritize requirements.

· Priority 1 – The requirement is a “must have” as outlined by policy/law

· Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits

· Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS **must be** either A or I" or "It **would be nice** if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

· A good requirement is:

· Correct

· Unambiguous (all statements have exactly one interpretation)

· Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)

· Consistent

· Ranked for importance and/or stability

· Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms and specify measurable quantities)

· Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)

· Does not specify any particular design

· Traceable (cross-reference with source documents and spawned documents).

## 3.1 Functional Requirements

In the example below, the requirement numbering has a scheme - BR\_LR\_0## (BR for Business Requirement, LR for Labor Relations). For small projects simply BR-## would suffice. Keep in mind that if no prefix is used, the traceability matrix may be difficult to create (e.g., no differentiation between '02' as a business requirement vs. a test case)

The following table is an example format for requirements. Choose whatever format works best for your project.

For Example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Req# | Requirement | Comments | Priority | Date Rvwd | SME Reviewed / Approved |
| BR\_LR\_05 | The system should associate a supervisor indicator with each job class. | Business Process = “Maintenance | 3 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_08 | The system should handle any number of fees (existing and new) associated with unions. | Business Process = “Changing Dues in the System”  An example of a new fee is an initiation fee. | 2 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_10 | The system should capture and maintain job class status (i.e., active or inactive) | Business Process = “Maintenance”  Some job classes are old and are no longer used. However, they still need to be maintained for legal, contract and historical purposes. | 2 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_16 | The system should assign the Supervisor Code based on the value in the Job Class table and additional criteria as specified by the clients. | April 2005 – New requirement. It is one of three new requirements from BR\_LR\_03. | 2 |  |  |
| BR\_LR\_18 | The system should provide the Labor Relations office with the ability to override the system-derived Bargaining Unit code and the Union Code for to-be-determined employee types, including hourly appointments. | April 2005 – New requirement. It is one of three new requirements from BR\_LR\_04.  5/11/2005 – Priority changed from 2 to 3. | 2  3 |  |  |

## 3.2 User Interface Requirements

In addition to functions required, describe the characteristics of each interface between the product and its users (e.g., required screen formats/organization, report layouts, menu structures, error and other messages, or function keys).

## 3.4 Performance

Specify static and dynamic numerical requirements placed on the system or on human interaction with the system:

· Static numerical requirements may include the number of terminals to be supported, the number of simultaneous users to be supported, and the amount and type of information to be handled.

· Dynamic numerical requirements may include the number of transactions and tasks and the amount of data to be processed within certain time period for both normal and peak workload conditions.

All of these requirements should be stated in measurable form. For example, "95% of the transactions shall be processed in less than 1 second" rather than “an operator shall not have to wait for the transaction to complete”.

### 3.4.1 Capacity

Include measurable capacity requirements (e.g., the number of simultaneous users to be supported, the maximum simultaneous user load, per-user memory requirements, expected application throughput)

### 3.4.2 Availability

Include specific and measurable requirements for:

· Hours of operation

· Level of availability required

· Coverage for geographic areas

· Impact of downtime on users and business operations

· Impact of scheduled and unscheduled maintenance on uptime and maintenance communications procedures

· reliability (e.g., acceptable mean time between failures (MTBF), or the maximum permitted number of failures per hour).

### 3.4.3 Latency

Include explicit latency requirements, e.g., the maximum acceptable time (or average time) for a service request.

## **4.** Use case bake from recipe in Pi:

**Precondition:** System is turned on and in starting state.

User selects bake from database mode

User selects receipe from the database and shows it on the web UI

Place the probe in the oven that is turned on.

When temperature is at the right level, alert the user that the temperature has been reached.

User bakes the pie according to the steps in the recipe.

The interface lets the user step through the bakning process and shows the user where in the process he is.

1a. The user places the temp probe in the pie.

The pi reaches the right temp before the time is out.

User is alerted.

User can choose if the baking is finished or if to continue until time is up.

1b. The user places the temp probe in the pie.

The pi reaches the end time before the right temp is reached.

User is alerted.

User can choose if the baking is finished or if to continue until temp is up.

User shuts down the system or chooses to restart with another recipe or in another mode.

User should be able to interrupt the process at any step. The system requires confirmation that the user wants to interrupt once the baking has started.

Provide a summary of the major functions that the product will perform. Organize the functions to be understandable to the customer or a first time reader. Include use cases and business scenarios, or provide a link to a separate document (or documents). A business scenario:

· Describes a significant business need

· Identifies, documents, and ranks the problem that is driving the scenario

· Describes the business and technical environment that will resolve the problem

· States the desired objectives

· Shows the “Actors” and where they fit in the business model

· Is specific, and measurable, and uses clear metrics for success