Software Requirements Specification

for a

socket-based  
storage system

Version 1.3 approved

Prepared by Martin Disch

15.04.2015

Table of Contents

1. Introduction 1

1.1 Purpose 1

1.2 Product Scope 1

1.3 References 1

2. Overall Description 1

2.1 Product Functions 1

2.2 Operating Environment 1

2.3 Design and Implementation Constraints 1

2.4 User Documentation 1

3. External Interface Requirements 2

3.1 User Interfaces 2

3.2 Communications Interfaces 2

4. System Features 2

4.1 Pinging/alive signal 2

4.2 Getting backup list 2

4.3 Getting backup 3

4.4 Storing data 3

5. Other Nonfunctional Requirements 3

5.1 Performance Requirements 3

5.2 Safety Requirements 3

5.3 Security Requirements 3

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Martin Disch | 08.04.15 | Phrasing | 1.1 |
| Martin Disch | 15.04.15 | Introduced JSON communication | 1.2 |
| Martin Disch | 15.04.15 | Handling of encoding over different platforms | 1.3 |

# Introduction

## Purpose

This document describes a socket-based storage system, running on a Raspberry Pi microcomputer.

## Product Scope

The software specified provides functionality for KnowledgeBase, a note-taking application for Android. The system described here will act as a remote backup storage.

## References

Being open source, the current version of KnowledgeBase, the application using the described service, can always be found on GitHub

<https://github.com/martindisch/KnowledgeBase>

The server software lives in a subfolder called “server” and is therefore also available in the same repository.

# Overall Description

## Product Functions

The server will listen to incoming connections and allow the Android application to check on its status, get a list of all available backups, get a certain backup’s data and store new data.

## Operating Environment

The software will run in a screen instance on a Raspberry Pi microcomputer, running a current build of Raspbian with Python 2.7.

## Design and Implementation Constraints

As there is need for port forwarding, only a single specified port can be used.

## User Documentation

Documentation will be provided in the form of comments in code and a detailed git history.

# External Interface Requirements

## User Interfaces

The user will only interact with the Android application.

Status information of the server will be printed to standard console output and there is only one console command for directly controlling the server.

## Communications Interfaces

Communication between the Android application and the server will be using the TCP/IP protocol and will be formatted in JavaScript Object Notation (JSON).

Due to the cross-platform nature of the project, different character encodings will pose a problem. This will be dealt with by rigorously implementing the Unicode ‘airlock’, which means that all internal data will be in Unicode and encoded in UTF-8 for storage or transmission and all incoming traffic will be decoded back into Unicode for internal usage.

# System Features

## Pinging/alive signal

4.1.1 Description & priority

When sending the ping command, the Android application will get a response to verify the server’s ability to save sent data.

4.1.2 Stimulus/Response Sequences

When receiving  
{“command”: “ping”}  
the server will respond with  
{“response”: “pong”}

## Getting backup list

4.2.1 Description & priority

When sending the entries command, the android application will get a list of all stored backups.

4.2.2 Stimulus/Response Sequences

When receiving  
{“command”: “entries”}  
the server will respond by returning a list of saved backups like  
{“entries”: [  
 {“date”: “2015-03-25 1330” },  
 {“date”: “2015-04-03 1415” }  
 ]  
}

## Getting backup

4.3.1 Description & priority

When sending the get command followed by the date/time of storage, the Android application can get a certain backup’s data.

4.3.2 Stimulus/Response Sequences

When receiving a date like   
{“command”: “get”, “date”: “2015-03-25 1330”}  
the server will respond by returning all encrypted data of said date like  
{“data”: “la8j9dskf9as8j2lkjsd8f9j2lkasdf32”}

## Storing data

4.4.1 Description & priority

When sending the store command followed by the date/time and data, the Android application can store the backup on the server.

4.4.2 Stimulus/Response Sequences

When receiving a store command like   
{“command”: “store”,  
 “date”: “2015-03-25 1330”,  
 “data”: “la8j9dskf9as8”}  
the server will respond with an ok once everything has been safely stored  
{“response”: “ok”}

# Other Nonfunctional Requirements

## Performance Requirements

The server does not need to handle multiple users at the same time.

The server does not need to be able to act on console input at all times, doing this between handling network requests is sufficient.

## Safety Requirements

As the server acts as backup, no further backup is required. The newest version will always be on the device running the Android application and failure of both systems is extremely unlikely.

## Security Requirements

As all data is already encrypted on the Android device. It remains in this state during transmission and while being stored, requiring no further security measures.

There is only so much harm that can be done by connecting to the server and storing data until storage is full, so there is no need for any authentication before storing.