SOFTWARE REQUIREMENTS SPECIFICATION

for

General Network Access Tester

Version 1.0

Prepared by Michael Vessia Jr.

February 11, 2016

Contents

Revision History

Name	Date	Reason For Changes	Version
21	22	23	24
31	32	33	34

1 Introduction

1.1 Purpose

General Network Access Tester is an Android application which helps IT departments, network administrators, or anyone that wants to monitor a network that they have access to. The application should be free to download on the Google Play store, and the application will be open source. General Network Access Tester is an Android application which tests network accessibility. The application provides information that will make it easier for someone to troubleshoot their network. The General Network Access Tester reduces the workload on IT departments by allowing them to troubleshoot more quickly.

This document is meant to give an overview of the features of the General Network Access Tester, making the process of using the application for one's own network simple, and making contributing to the project easy and painless.

1.2 Document Conventions

The General Network Access Tester will hereafter be referred to as GNAT. Users will be someone that interacts with the mobile application. An administrator will be the person who owns the network and is likely having the logs sent to them. The administrator can and likely will also be the user.

1.3 Intended Audience and Reading Suggestions

This document is meant for administrators who would like to have their network monitored by GNAT, or developers who would like to make contributions to GNAT. It is not required that users read this document, but if they would like further clarification about the settings they they are configuring within the application, they will find that information in this document. Developers will be interested in the source code. As of version 1.0 , the source code can be found at https://github.com/MichaelVessia/gnat.

1.4 Project Scope

None as of version 1.0 .

1.5 References

None as of version 1.0 .

2 Overall Description

2.1 Product Perspective

GNAT is targetting network administrators who have a considerable amount of mobile devices connecting to their network, and would like more information about failed connections. GNAT should integrate well into an existing monitoring/dashboard system, but ultimately should be a stand-alone application and not depend on any other software.

2.2 Product Functions

Within the mobile application, users will be able to enter a number of configuration options. The network connection that will be attempted will be based upon these configuration options. The result of this connection attempt will be logged if anything goes wrong. Although the application tests network access, it ultimately assumes that the user has access to the internet. If there is an initial connection error, logs will be stored locally until a connection is made, at which point it will send the logs via the specified means of communication.

The application needs both internet and GPS connection to generate the desired logs and send them to the specified location. The GPS will allow the logs to contain relevant geographical information, like where the device was when it attempted to connect from.

An administrator will be allowed to create a web page, likely containing some kind of file. The app will attempt to connect to this web page. The file will give the administrator some flexibility. GNAT can simply check if the file was able to be downloaded, or it can be a script that does something else.

If the administrator uses a dashboard service that has an API, GNAT will be able to send the logs via the API to be viewed in the dashboard service.

2.3 User Classes and Characteristics

The user of the application is expected to either be the administrator, or be a knowledgeable user that recieved the relevant information from the administator. End users of the network should not have to interact with GNAT, but assuming they had the relevant information nothing would be stopping them.

2.4 Operating Environment

GNAT will run on the Android operating system. The application will require the user to have at least Android 4.0.3 (Ice Cream Sandwich). It also assumes there is a network to be tested.

2.5 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).>

2.6 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.>

2.7 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).>

3 External Interface Requirements

3.1 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.>

3.2 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.>

3.3 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.>

3.4 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.>

4 System Features

<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.>

4.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

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

Temporary info to be revised:

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

REQ-1:REQ-2:

4.2 System Feature 2 (and so on)

5 Other Nonfunctional Requirements

5.1 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.>

5.2 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.>

5.3 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.>

5.4 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.>

5.5 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.>

6 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.>

6.1 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.>

6.2 Appendix B: Analysis Models

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

6.3 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.>