# Software Requirements Specification

for

Institute Firewall Login Manager for Android

Mohammed Shahraaz Hussain B170252CS B Batch
Ishan Ghosh B170473CS B Batch
Kapil Gyanchandani B170583CS B Batch
Vinay Kumar Saxena B170338CS B Batch
Goutham Krishna K V B170214CS A Batch

*National Institute of Technology, Calicut*
*Computer Science and Enginnering*
*Bachelor of Technology (BTech)*
*Software Engineering (CS3004D)*

January 2020

# Contents

# Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 1.0 | 22.01.2020 | All | Revision 1 |
| 2.0 | 26.01.2020 | All | Revision 2 |
| 3.0 | 27.01.2020 | All | Revision 3 |
| 4.0 | 28.01.2020 | All | Revision 4 |

# Introduction

## 1.1 Purpose

The purpose of this document is to capture, in natural language and a functional level, the description and requirements of a Institute Firewall Login Manager Application. The focus here is to automate the process of connecting to the institute network. This is a functional description of those features required to address the current network connectivity issue. A short discussion accompanies each requirement, to add the background and framework necessary to explain the functionality. It also describes nonfunctional requirements and other factors necessary to provide a complete and comprehensive description of the requirements for the software.

## 1.2 Document Conventions

This document was created based on the IEEE template for System Requirement Specification Documents.

Abbreviations:

- FLMI: Firewall Login Manager for Institutes.
- WiFi: Wireless Fidelity, a local area wireless network.
- FOSS: Free and Open Source Software
- UI: User interface.
- URL: Uniform Resource Locator.
- CNC: Campus Networking Center.

## 1.3 Intended Audience and Reading Suggestions

This Software Requirements document is intended for:

- Developers who can review project's capabilities and more easily understand where their efforts should be targeted to improve or add more features to it (design and code of the application. It sets the guidelines for future development).
- Project testers can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. This way testing becomes more methodically organized.
- End users of this application who wish to read about what this project can do.

## 1.4 Product Scope

Institutes all over the country use a firewall to allow only institute members to use the campus network. This Login Manager has two internal scopes, one dealing with the details of the user account and second dealing with the Institute Firewall.

### 1.4.1 Objectives

- Automate the login and logout process for connecting to the campus network.
- Reduce delay and number of steps.
- Store User Account details.
- Reduce Keep-alive errors.

### 1.4.2 Expectations

- Only one time entry of the user Account details required.
- Use only a single button to toggle between login and logout states.

### 1.4.3 Constraints

- The user upon leaving the network before logging out, is then locked from another login for 40 minutes or until he is connected to the same network.
- When the network breaks down the application will stop functioning.

## 1.5 References

- IEEE SRS template:
  `https://www.overleaf.com/latex/templates/requirements-specification-layout/vbrqbjpzcmfy`
- GitHub Project Link: `https://github.com/Shahraaz/LoginManager`
- Wikipedia: `https://www.wikipedia.org/`
- Use Case Diagram: `https://www.draw.io/`

# Overall Description

## 2.1 Product Perspective

FLMI serves as an automation tool for connecting to the campus network. It strives to eliminate the need to enter the user details, which is the username and password of a student or a member of the institute manually. FLMI also reduces the multiple steps required to authenticate and login to the campus firewall network. The application aims to remove user-generated issues such as the loss of the Keepalive window, forget passwords. The application helps secure connectivity and worry-free access to the campus network. FLMI works on all mobile devices which run on the Android operating system.

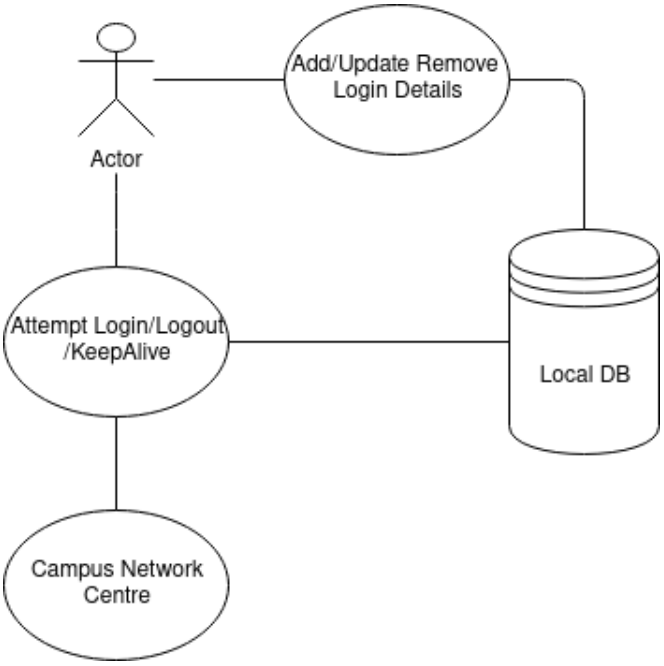
## 2.2 Product Features

FLMI provides users the following features:

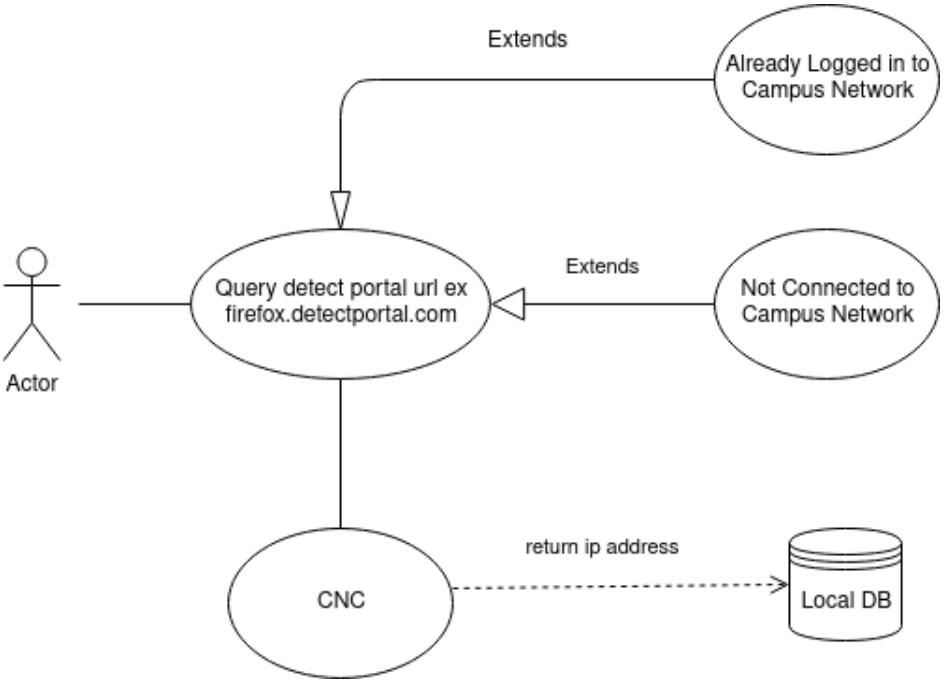
- Detect IP address and magic number: The application detects the IP address of the router of the WiFi network to which the mobile device is connected and receives the webpage containing the firewall portal.
- Login: One tap of a button can log the user into a connected network.
- Logout: The user can logout of a already logged in network using a button.
- Keepalive: The login of a user is maintained continuously by re-sending login requests after Keepalive time has expired.
- Add, Delete or Edit multiple user accounts: Multiple usernames and passwords for connecting to the firewall can be stored in the database of the application.
- Set credentials preferences: Most recently used Account is stored as next account to be used for login.
- Log Keepalive history: User login history is stored in the database so if a user forgets to logout of a network he can know to which network he was last connected.

### 2.2.1 Use Case Diagrams

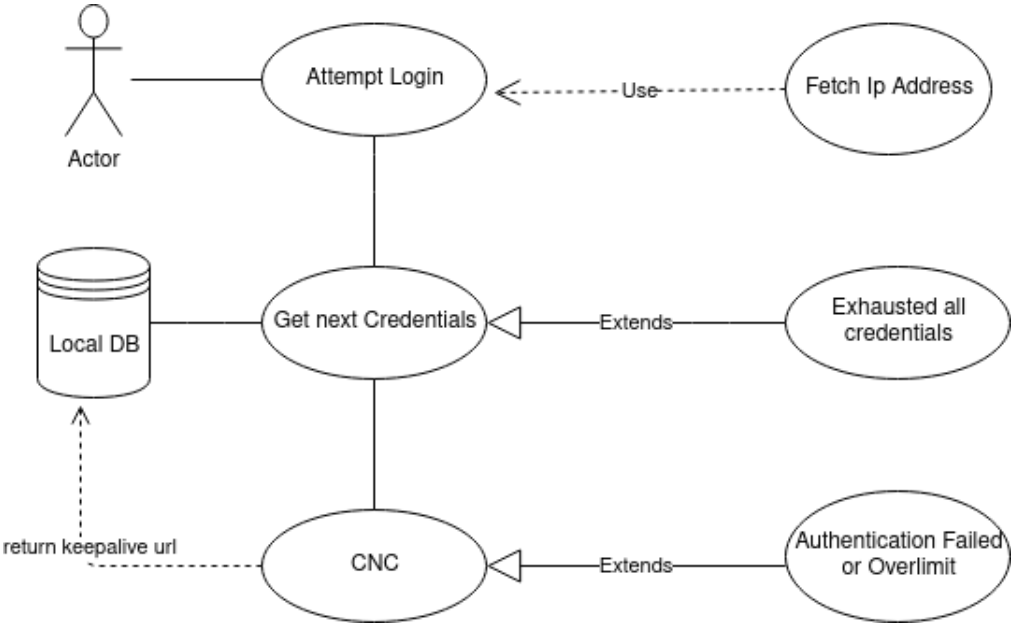
**Overall Systems**



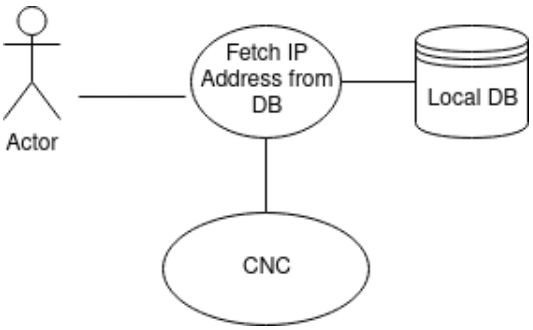

**Fetch IP Address and Magic Number**



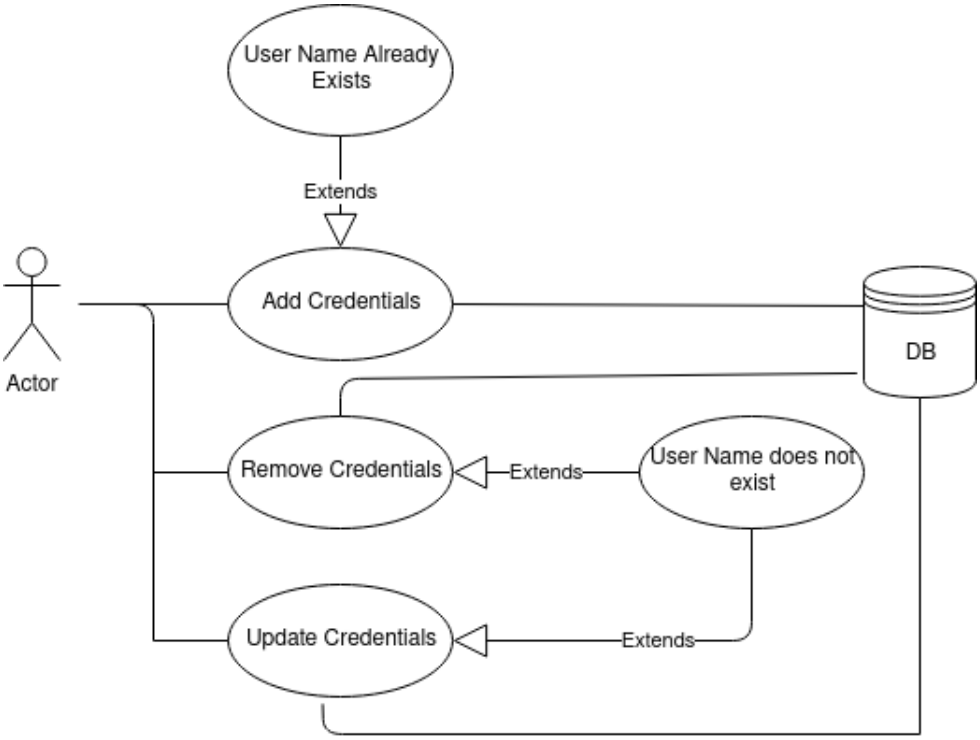
**Login**



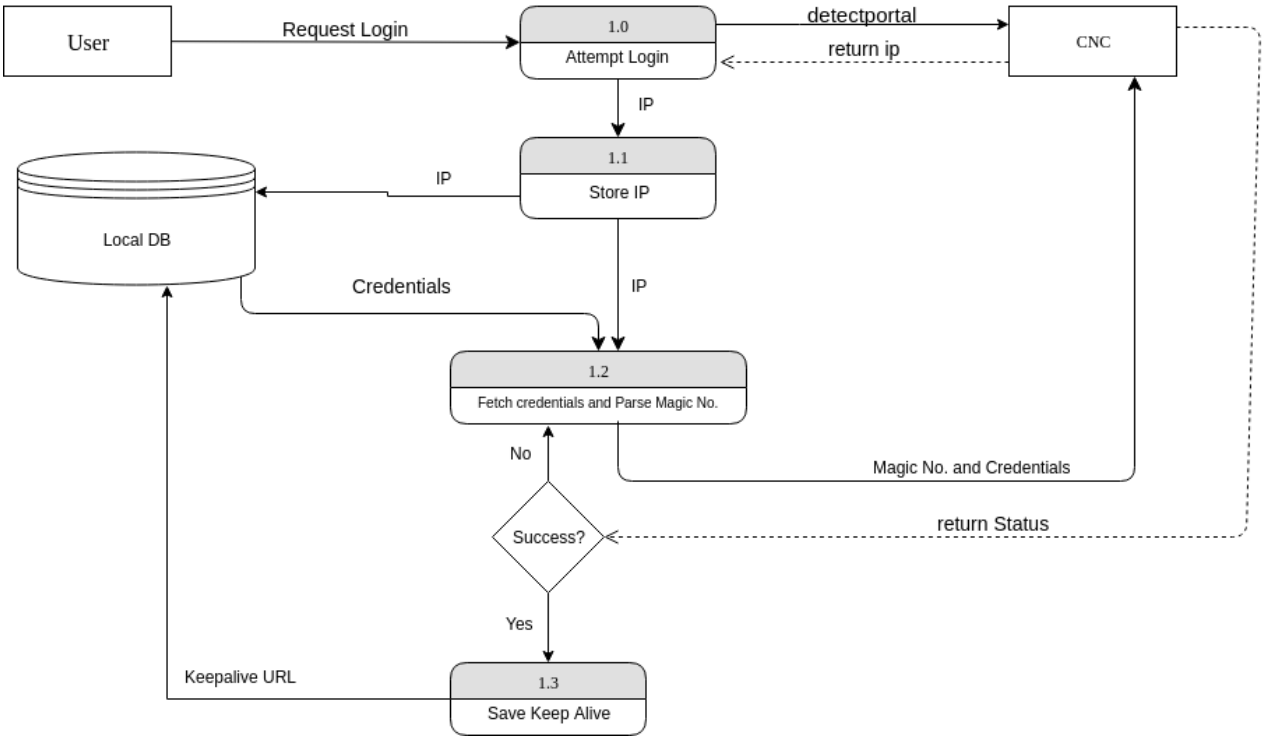

**Logout**



**Add or Remove Users**




### 2.2.2 Data Flow Diagram for login



## 2.3 User Classes and Characteristics

- All kind of users: FLMI is a very simple to use application, all users with basic experience with technology are able to use it efficiently.
- Open source software developers and contributors:
  - Software developers: People with very good knowledge of the programming language, the project, in order to understand and be able to extend project's source code.
  - Anyone who wants to help the FOSS community. The whole project is based on the conception of Free and Open Source Software, so all people are welcome to contribute any way they can/like.

## 2.4 Operating Environment

FLMI requires an Android OS mobile device. An android device that can support basic dependencies of the application is expected for proper user experience.

## 2.5 Design and Implementation Constraints

FLMI is under the GNU General Public License Version 3.0, 2007. Everyone, that does or is going to develop or use FLMI, should agree and fully accept the terms of this kind of license.

## 2.6 User Documentation

Any issues or questions related to the software can be asked by raising an issue on the github page of FLMI. `https://github.com/Shahraaz/LoginManager`

## 2.7 Assumptions and Dependencies

FLMI is an android application it requires and mobile Android device to run. The Apk file has to be downloaded and installed on the device. Proper permissions should be granted when requested to the user to ensure the full functionality of the application.

The device should be connected to a strong WiFi network with a firewall only then can the login and logout functionality be used. The device, once logged in from one network, cannot be logged out on another network.

# External Interface Requirements

## 3.1 User Interfaces

Using the application is relatively simple and intuitive. A user familiar with the primary android application interface and navigation skill should be able to understand all functionality provided by the application. Nothing additional is required.

## 3.2 Hardware Interfaces

The application should work on mobile devices which are based on the Android operating system. No additional hardware interfaces required.

## 3.3 Software Interfaces

The application should run on Android 5.0 and above. Android below 5.0 (Lolipop) might cause some undefined behavior. The android device should have the application installed and proper permissions granted. It should also have the ability to connect to a WiFi network using the system hardware.

## 3.4 Data Management Interfaces

The application should be able to interact with other components in the android device according to their specifications. Its operating device also limits the application in terms of the maximum number of user accounts the device's database can store at a particular time.

The tables in the database are as follows:

**USER**

| UserID | UserName | Password |
|---|---|---|

**NETWORK**

| NetworkID | Local IP Address | Location |
|---|---|---|

**HISTORY**

| Keepalive URL | Timestamp |
|---|---|

## 3.5 Communications Interfaces

There is a requirement of a working WiFi connection with a firewall setup and a valid user account in that firewall to use all of the functionality of the application. Additionally, an internet connection is required to download the application and update it. The only functionality available offline is the ability of the user to add multiple accounts and change passwords of an account.

# System Features

The Priority ranges from 1 (most important) to 10 (least important).

## 4.1 Fetch IP Address and Magic Number

### 4.1.1 Description and Priority

The device would request a detect portal request though the Network. The network sends back a firewall login page back to the device requested. Priority = 1.

### 4.1.2 Functional Requirements

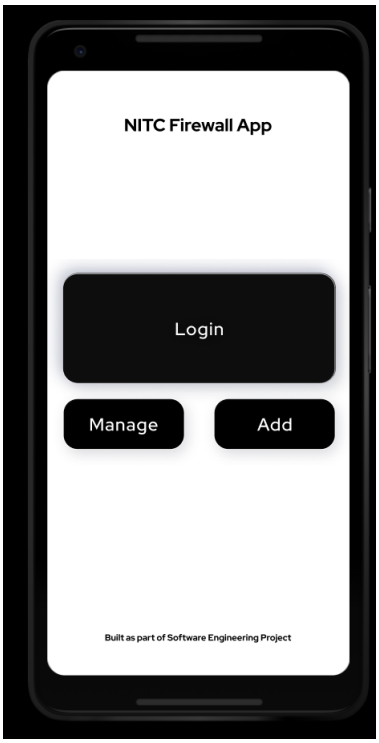
- **Purpose** : The IP address indicates which network the user is connected to and the magic number indicates currently which user is connected to which network.
- **User** : The application sends a request.
- **Input Data**: request to detectportal.firefox.com.
- **Output Data**: Corresponding Page Data.
- **Invariants**: Device is connected to a network.
- **Pre-conditions**: The mobile device is not logged in to the network.
- **Post-conditions**: The page data of the firewall portal is sent to the application
- **Basic Flow**: The application sends a request for the firewall portal from the mobile device the firewall if the device is not already logged in to the network sends back data regarding the firewall portal page.
- **Alternative Flow(s)**: If the device is already logged in to the network the request doesn't return back anything.
- **Business Rules**: This allows the user to access the login portal.

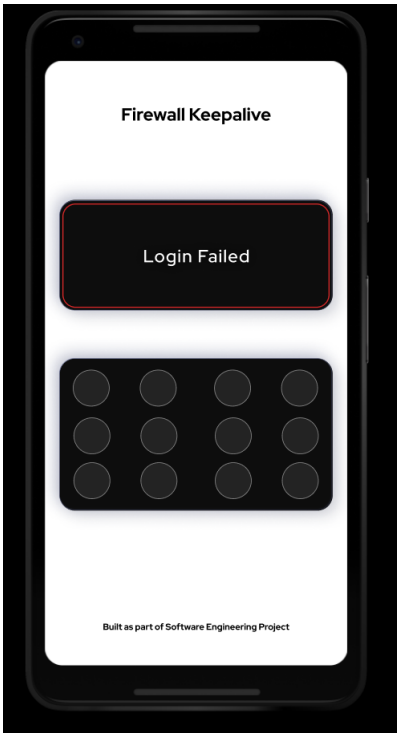
## 4.2 Login

### 4.2.1 Description and Priority

The system sends a login request to the network portal. The network portal, upon receiving this then responds with either a keepalive page (indicating that the user was logged in) or with another login (indicating that the

user wasn't authenticated). Priority = 1.




### 4.2.2 Functional Requirements

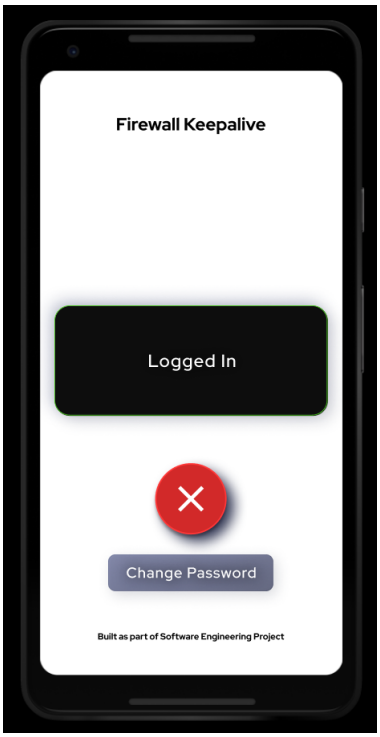
- **Purpose**: Give the user rights to access the internet.
- **Input Data**: Valid user details.

- **Output Data**: KeepAlive Page data of the firewall.
- **Invariants**: Profile table data and user information.
- **Pre-conditions**: User is not logged into a profile, input profile is valid, user password matches profile.
- **Post-condition**: The mobile device is supplied with page data of the keepalive window for the selected profile.
- **Basic Flow**: The firewall looks up profile data and returns the data for the keepalive page. The application stores the url in the log history.
- **Alternate Flow**: Invalid password, invalid username, or mismatched username and password redirect to error message.
- **Business Rules**: This allows users to log in to their profile.

## 4.3 Logout

### 4.3.1 Description and Priority

The device sends a logout request. This doesn't need the user credentials to do this. The network, upon receiving this, then responds either with a logged out page if the user was logged in, or with a login page if no user was logged in. Priority = 3.




### 4.3.2 Functional Requirements

- **Purpose** : The Logout then logouts from the network and informs the user of the status change.
- **User** : Gets a Logout Page informing the user on successful logout.
- **Input Data** : User taps logout button.

- **Output Data** : 'Successful Logout' response from the network.
- **Invariants** : Logout Status of System in the Network.
- **Pre-Conditions** : The user is at the keepalive page.
- **Post-Conditions** : The user recieves the status upon clicking logout button.
- **Basic Flow** : The system upon recieving the user signal, then requests a logout through the network, informing that the user intends to logout of the network. The network then sends a 'Successful Logout' response informing the same.
- **Alternative Flow** : The network responds with a login page which indicates that the system is not logged in. The system then informs the user of the login change and requests the user to try logging in again.
- **Business Rules** : This feature lets the user login and logout properly.

## 4.4 KeepAlive

### 4.4.1 Description and Priority

The system sends a keepalive request at specific intervals of time to the network. Priority = 2.

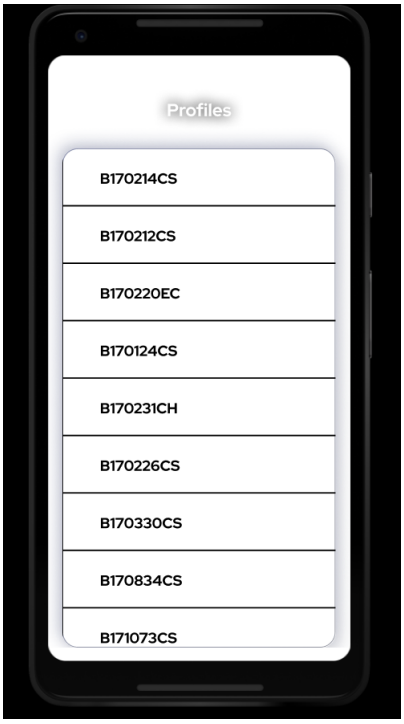
### 4.4.2 Functional Requirements

- **Purpose**: Extend the Login window beyond 40 minutes.
- **User**: The application interacts with the firewall.
- **Input Data**: keepalive URL.
- **Output Data**: Success / Failure.
- **Invariants**: Login State, USER table and Network table.
- **Pre-conditions**: User is logged into a Campus Firewall.
- **Post-condition**: Keepalive table updated.
- **Basic Flow**: Application must request KeepAlive URL to CNC.
- **Alternate Flow**: Report error.
- **Business Rules**: This leads to no timeout after 40 minutes in keepalive.

## 4.5 Add, Delete or Edit Multiple User Accounts

### 4.5.1 Description and Priority

This feature lets the system manage multiple credentials on a single device with operations such as add, remove and manage credentials, so that the system can potentially use multiple accounts to authenticate using the system. Priority = 5.




### 4.5.2 Functional Requirements

- **Purpose** : To allow the user to manage (add, modify or remove) credentials.
- **User** : The user recieves a list of credentials to manage.
- **Input Data** : User clicking 'Add' or 'Remove' or 'Edit' on a certain user.
- **Output Data** : 'Add' Page, 'Edit' Page on a Credential, or list without the user.
- **Invariants** : Human Error (Unintentionally Clicking Edit/Remove)
- **Pre-Conditions** : List of Profiles (accessible by clicking 'Manage' from Home-Page)
- **Post-Conditions** : Page for Add/Edit or List, whichever is relevant.
- **Basic Flow** : The user clicks a button and the system responds properly.
- **Alternate Flow**: Report error.
- **Business Rules** : This allows the user to manage the credentials in a sane manner.

## 4.6 Set Credential Preferences

### 4.6.1 Description and Priority

This feature lets the user manage the preferences for multiple credentials using a page for editing the account preferences. Priority = 5.

### 4.6.2 Functional Requirements

- **Purpose** : To allow the modification of credential properties.
- **User** : The application sends preferences page to allow modifications.
- **Input Data** : User Proposed Modifications
- **Output Data** : Status on Accepting or Rejecting Proposed Credential Modifications
- **Invariants** : Input from User
- **Pre-Conditions** : The user selects a credential to change preferences.
- **Post-Conditions** : Page exits on accepting changes or notification on rejected changes.
- **Basic Flow** : The application first presents a UI to modify the credential properties, upon clicking 'Submit' the system checks the modifications upon accepting it the system updates the database to reflect the modifications, upon which the application goes back.
- **Alternative Flow** : The application on recieving unintended changes, stops execution following which the system stops execution and informs the user of wrong information.
- **Business Rules** : This allows the user to change the account preferences.

## 4.7 Log Keepalive History

### 4.7.1 Description and Priority

This feature lets the system keep a keepalive history so that the system can potentially manage login and logout process better and also to keep track of any problems the app might act on, which could be used to also debug the software. Priority = 2.

### 4.7.2 Functional Requirements

- **Purpose**: It keeps a keepalive history which indicates which tracks and logs all the status changes of the system.
- **User**: The Application undergoes a change in keepalive state.
- **Input Data**: Signal Log for Keepalive Database.
- **Output Data**: Response on recording the log.
- **Invariants**: Login Status, Credential Preferences, Database Status
- **Pre-Conditions**: The Database for the Authentication has been set.
- **Post-Conditions**: The Database logs the Status Change (Login/Logout) for the system.
- **Basic Flow**: The system recieves the signal for the log change. The system then adds a log in the database logging in detail the change in log status and then returns back to respond with confirmation on return.
- **Alternative Flow**: The system upon recieving the signal, if does not find a valid database, then informs the user by the use of the application's notification system to notify the user that the database was not working properly, and then given a way to contact us for modifications.
- **Business Rules**: This allows the applications to work efficiently and properly.

# Other Nonfunctional Requirements

## 5.1 Performance Requirements

FLMI is a light application that needs very few system resources to work. It is designed not to delay the system from other key processes and the response time of the program is immediate. Performance might be affected when the application has to check through multiple user accounts to find a valid login id.

## 5.2 Safety Requirements

The application is open-source, and its source code is freely available on GitHub. Any bugs or issues can be raised on the repository page. The community fixes those bugs and issues on its next release cycle.

## 5.3 Security Requirements

FLMI requires write access to the android SQLite database to store the user account details. Passwords of the users are saved to a database in the mobile device storage. The user's KeepAlive history is stored in the database of the mobile device.

## 5.4 Software Quality Attributes

FLMI has a well defined and easy to use interface it can be used by both experts and typical users. However, users must already have a basic knowledge of the Android UI before using it.

# Appendices

# Glossary

- **Software Requirements Specification**: A software requirements specification (SRS) is a description of a software system to be developed.The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.
- **GNU General Public License**: The GNU General Public License (GNU GPL or GPL) is a widely-used free software license that guarantees end users the freedom to run, study, share, and modify the software.
- **GitHub**: GitHub is a global company that provides hosting for software development version control using Git. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.
- **Free and open-source software**: Free and open-source software (FOSS) is software that can be classified as both free software and open-source software.[a] That is, anyone is freely licensed to use, copy, study, and change the software in any way, and the source code is openly shared so that people are encouraged to voluntarily improve the design of the software
- **Android**: Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.
- **APK**: Android Package (APK) is the package file format used by the Android operating system for distribution and installation of mobile apps and middleware.
- **network**: A network is a set of nodes connected by communication links. A node can be a computer, printer, or any other device capable of sending or receiving data from the other node through the network
- **WiFi**: Wi-Fi is a family of wireless networking technologies, based on the IEEE 802.11 family of standards, which are commonly used for local area networking of devices and Internet access.
- **firewall**: A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules.A firewall typically establishes a barrier between a trusted internal network and untrusted external network, such as the Internet.
- **database**: Database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques.