Software Architecture

Smart Door

CSIR
ZEBRA-V
Eduan Bekker (12214834)
Albert Volschenk (12054519)
Zühnja Riekert(12040593)

Publication Date: 23/05/2014

Version 1.0

Change Log

16/05/2014	Version 0.1	Document Created	Albert Volschenk
16/05/2014	Version 0.2	Document Layout	Eduan Bekker
22/05/2014	Version 0.3	Quality Requirements	Eduan Bekker
22/05/2014	Version 0.4	Architecture requirements and constraints	Zühnja Riekert
22/05/2014	Version 0.5	Architectural scope	Zühnja Riekert
23/05/2014	Version 0.6	Architectural pattern	Eduan Bekker
23/05/2014	Version 0.7	Edited overall content	Albert Volschenk
23/05/2014	Version 1.0	First Publication	
07/07/2014	Version 1.1	Updated functional requirement values	Eduan Bekker

Index

- Change Log
- Index
- Architecture requirements
  - Architectural scope
  - Quality requirements
    - Performance
    - Scalability
    - Integrability
    - Security
    - Flexibility
    - Maintainability
    - Auditability
    - Usability
    - Accuracy
  - Integration and access channel requirements
    - Access channel requirements
    - Integration requirements
  - Architectural constraints
- Architectural patterns
- Technologies

Architecture requirements

Architectural scope

Some sort of database will be needed to store faces and voice ID's.

Although it has not yet been specified as a requirement, it may well be needed to implement a system that logs all access attempts and stores the face of any person attempting access that is not on the system (thus not authorized).

Quality requirements

Performance
The user interface should have a response time of less than 2 seconds. Facial recognition should not take longer than 7 seconds. Waking the system by passing users should be instantaneous and should not take longer than 1 second.

Scalability
Each instance of the application should only be able to handle one user at a time.

Integrability
This system should be easily integratable with the current Cmore system and future projects.

Security
No user should be granted any access without authentication. Both face recognition and voice ID identification is used.

Flexibility
The system should always have the option of pin login, if the facial and voice recognition fails.

Maintainability
The system should be easily maintainable by future developers who would like to add functionality and integrate with more systems. A developers guide on how to build the project will be included.

Auditability
Audit logs of all the actions will be recorded.

Usability
The system should have a user friendly interface corresponding with the colour schemes of the CSIR and Cmore.

Accuracy
The system should have a 90% accuracy on facial recognition, i.e. a person should be correctly recognised 90% of the time. Voice recognition should have a 99% accuracy for a valid recognition and it should achieve this accuracy 80% of the time. If the system could not recognise a user then there should be infinite amount of retries.

Integration and access channel requirements

Access channel requirements

The system's services are to be accessed through an Android application on a tablet, mounted to a door. There is also a twitter feed displayed on the home screen. Each Smart Door application should have its own Twitter account.

Integration requirements

Cmore is an existing system that tracks the location of users. As a bonus feature that is not required at this stage, integration to Cmore may be implemented. The integration can be done by making HTTP requests to Cmore's web-based RESTful API.

Although this is not part of the project requirements, the Smart Door app must allow easy expansion and integration with Cmore.

Architectural constraints

The application should not be a HTML5 or cross-platform solution, but be written in Java with the Android SDK. The target version of Android is 4.4, but the application needs to be backwards-compatible to Android version 4.0.3.

Any open source third party library for text-to-speech, speech-to-text, motion detection and the facial recognition may be used.

This user interface of the application should be designed for a 10.1" tablet.

Architectural patterns

A Layering pattern will be used to keep relevant parts of the system apart. This will improve maintainability of the system. If future developers would like to update the UI then only the UI layer needs to be updated and the rest will work as expected. This will simplify integration with existing/future projects. When integrating only the relevant layer needs to be updated or a new layer can be added.

Technologies

- Tablet
- Android operating system
- Java

- Android SDK
- Android API or any 3rd party libraries
- May use SQLite