rijksuniversiteit groningen

Hestia for iOS

Requirements Document

Client: Lars Holdijk

iOS Team:

Andrei Mădălin Oancă

Daan Groot

Geanne Barkmeijer

Juanjo José Méndez Torrero

Jur Kroeze

Marc Fleurke

Tom den Boon

Zino Holwerda

TA: Feiko Ritsema

Version: 0.1.0

2018

Introduction

With the introduction and advancements of technology in more and more aspects of our lives, a new area of interest has emerged, home automation. Because of the increasing interest in this area, many platforms have been created that promised to ease the control of the hardware in a home. Many manufacturers have their own platform for their products, an approach that is not desired for the end users. Just imagine the hassle of needing to switch systems each time you want to turn on the lights in the bathroom and kitchen, just because the smart light bulbs are not created by the same manufacturer. There are problems even on platforms which promise to function with a wide variety of peripherals.[1] The problem lies in the fact that the developer needs to manually update the platform in order to work with new hardware. This is a problem which hinders the regular user from using such platforms.

With the Hestia system, we want to change the market and create a platform which is independent from the hardware used in home automation. The main idea is that the system is extendable and the user can add new peripherals without anyone needing to change the internal structure of the platform. The user can create their own plugins for the peripherals that they possess. Using these plugins, the system Hestia will be able to control everything and the user will never be required to change systems just because of different hardware.

The most important goal of this system is to create a stable platform which is extendable and makes it easy to control every smart device in a user's home. Any such device should be able to connect to our system. The list of possible peripherals that the system will support are the following, but these are not the only ones and many more should be included in the future: light bulbs, smart locks, refrigerators and heating systems.

Our team's main task is to develop an iOS application extending the existing Hestia system that the home user can use in order to control the activity of the peripherals he owns.

The requirement document will now continue with a short system overview and then our team will present you the functional and non-functional requirements. The requirements are categorized by their importance in the final project.

System Overview

The system that we are planning to implement already consists of the following components: the server side and the Android application. With our main goal in mind, which is to create a really versatile system to work with, we will also implement an application for iOS devices.

Any peripherals that will be added to the Hestia system must have the ability to be controlled by means of the Android application, iOS application and the Web Interface. Using the iOS application, the user should be greeted with a well polished design, that follows the Apple's Human Interface Guidelines.[2] The application should also be really easy to operate,

[1] http://fortune.com/2015/08/19/smart-home-stupid/
[2] https://developer.apple.com/ios/human-interface-guidelines/overview/themes/

without being too cluttered with information. With the iOS application, the user should be able to control, install and remove peripherals.

All the information regarding installation, removal and control of the peripherals, which will be done from the iOS application, will be sent to and managed by the server. The server will apply or save the changes.

Critical Functional Requirements

- The application must connect to the server and display all available peripherals to the user.
- The application must also connect to the peripherals through the server such that the peripherals are not managed by the application itself.
- The user must be able to select various actions for different peripherals such that they can interact with the peripherals within the house.
- The user should be able to add peripheral plugins through the application for interaction with said peripheral. By doing so adding the peripheral in the process.
- Every peripheral should have its own menu with specific controls.

Important Functional Requirements

- User should be able to use voice control to interact with the application.
- A system for logging in safely with username and password to make sure there is no unauthorized access to the peripherals on the local server.
- The peripherals should be easily identifiable and distinguishable and open to change even after first declaration. (e.g. icon, name change)
- The user should be able to see a status for the peripherals connected.

Useful Functional Requirements

- Creating new user profiles directly from the application.
- Different controllers for the peripherals besides switches and sliders.
- Connecting to the server through a web server for better security and creating a secure way of connecting to the server from outside of the local network.
- Scheduling actions such that they can be performed at a later time by the system. (e.g. switching lights off after midnight).
- Ability to set timers on certains actions, such that they only take a set amount of time. (e.g. keeping the light on for specific amount hours.
- Expanding on personal security; authorization services supplied by third party providers would also be an option. (e.g. Facebook, Google)
- Location based control of peripherals. (e.g. lights go out if the person leaves the house)

Won’t do

- Implementing different methods of connecting to the server, besides the internet protocols. (e.g. Bluetooth)

Critical Non-Functional Requirements

- The traditional iOS look and feel should be best maintained while also keeping similarities to the Android and web application.
    - Many iPhone applications have menu items in the bottom of the screen, this could become a design option if so many functionalities are needed.
    - Settings and edit are mostly in specific corners, this could also be used as a design template.

Important Non-Functional Requirements

- Clean code.
- Concise, visual yet clear documentation.

Useful Non-Functional Requirements

- Secure transmission of data between the application and server. (e.g. through encryption.)
- The code is tested with TravisCI (continuous integration testing) and has high code coverage (CodeCov)

Customer Meeting Log

When	**What**
February 23 2018	· Customer wants voice control capability of the application · Customer wants a nice iOS look and feel design of the application. · Functionality is not important first, but installing peripherals is primary

Change Log

Who	**When**	**Section**	**What**
M. Fleurke	February 23 2018	Document	Created document + headings.
A. M. Oanca	February 24 2018	Document	Added extra headings.

			Finished paragraph Introduction. and System Overview. Modified design of the cover.
Z. Holwerda	February 25, 2018	Document	Merged and updated requirements. Adjustments of style all across. Added headings. Updated the introduction & overview.
D. Groot	February 25, 2018	Document	Improved formulation and fixed some grammatical errors.
M. Fleurke	February 26, 2018	Document	Added location based control requirement. Started glossary. Small grammar fixes.
A. M. Oanca, M. Fleurke, Z. Holwerda	February 26, 2018	Document	All issues resolved, requirements merged even more. Fixed consistency.
A. M. Oanca	February 27, 2018	Document	Some retouches for Introduction. Updated the logo.
M. Fleurke	February 27, 208	Introduction	Fixed/added the footnotes.

Glossary

Peripheral
Device that is controlled by the application (via the server), for example a light bulb or a smart lock.

Server
The local server on which information of peripherals is stored. The application interacts with peripherals through the server.