Bright Light Systems
System Design Document
v. 1.0

by Michael Gulenko & Dallas Winger

INTRODUCTION:

Bright Light Systems is an app for Android OS and iOS that allows users to take full control over the Philips Hue light bulbs by communicating user's requests to a Hue Bridge which on its end will adjust the lights according to request.

The main set of features consist from following:

- Color/Brightness Change
- Color/Brightness Transitions Over Time
- Theme Organizer
- Scheduling (Alarm or Timer based)
- Location Based Actions (Auto On/Off)
- Message Notification (Text, Email, Facebook, etc)
- Music Mode
- Homescreen Widget Control
- Smart Suggestions

This document will describe high level system design specifications of this features as well as components' interaction within the system.

GOALS AND CONCERNS

There are several factors that dictate certain criteria for the design. The first factor is a marketing strategy. Bright Light Systems will be developed and released as a commercial product. The system provides a collection of different services, where individual price will be applied per different service. Before purchasing, users will have an opportunity to try a specific service for fixed period of time, and then will have to purchase it for further use.

Another factor is the development process and the future support of the product. The first release may or may not include certain services which can be developed and added later on. Thus, the design has to be flexible and clean, allowing developers to add new features and services in future while the system itself has to function independently from available services.

Security
The Philips Hue system has built in measures for security that this app will have to comply with to function properly. Upon pairing to a Hue Bridge, the system will require the user to prove physical access to the system by pressing a button on the Hue Bridge; much like WPS technology implemented on many of today's wireless routers. The main security goal for this project is to not allow unintended access to a Bridge or set of lights. Additionally any information this system stores (in the Database component) considered to be unique to a user, or that could be used to allow access when unintended, should be obfuscated (in the form of a hash or encryption algorithm).

Battery life and Performance
Being an application for a mobile platform, this app should perform as efficiently as possible. On an "average" device this app shouldn't exceed 15% of the battery usage. This is an important metric to look at because the feature set involved will require background services to listen for events. This will be the responsibility of the Service Manager component.

SYSTEM COMPONENTS

The following components describe internal structure of the system.

- Activity Stack
- Database
- Service Manager
- Gateway
- Trait Analyzer

Activity Stack
The Activity Stack will be responsible for all of the UI and its logic. An Activity in Android is considered to be: "a single, focused thing that the user can do." An activity consists of a Java class and XML layout. The XML layout file is responsible for controlling the UI layout to be displayed to the user, while the Java class is responsible for all of the listeners and logic behind the Activity/UI. As Bright Light Systems will have more than one UI screen, it will require multiple activities; thankfully the Activity Stack itself, is managed by Android.

Database
The database component is responsible for storing and retrieving information about light bulb settings, themes, events, and services. The data will be stored on the device, and access to the database will be provided through SQL interpreter that will be issuing SQLite queries for the Android environment or MySql for the iOS.

Service Manager
The service manager will handle tasks related to services such as start, abort and lock or unlock. The service manager will be responsible for retrieving data from the Android Service(s) running in the background; such services would include Geofencing, notification listeners, etc. It will be the job of the Service Manager to control whether or not that service is in use.

Gateway
The gateway component will be completely responsible for Philips' Java Multi-Platform and Android SDK. To make the desired changes to lights connected to the Hue Bridge locally, Phillips has given developers a Hue Bridge API. This is a RESTful API over HTTP. This means that every controllable parameter has its own local URL to make the request to. All Hue Bridge responses return in JSON blobs (JavaScript Object Notation) with UTF8 encoding for easy parsing. This API requires direct access to your bridge so you'll only be able to access it when your app and bridge are on the same local network. To access the bridge from outside of the network the user must interact with the portal (a web based control panel which connects your home to the internet). For this, a web view will frame the control panel to allow for this access.

Trait Analyzer
The Trait Analyzer component will be responsible for any and all AI algorithms used to intelligently order items in the activity stack/UI. This will include ordering themes by popularity, displaying most often used bridges/bulbs at the top of any listing, etc.

INTERNAL COMMUNICATIONS

The diagram 1 describes communication and logical data flow between components. The single headed grey arrows indicate request messages, whereas double headed grey arrow indicates request-response type of messages. A thick blue double headed arrow indicates that communication is happening over the Wi-Fi using Phillips API.

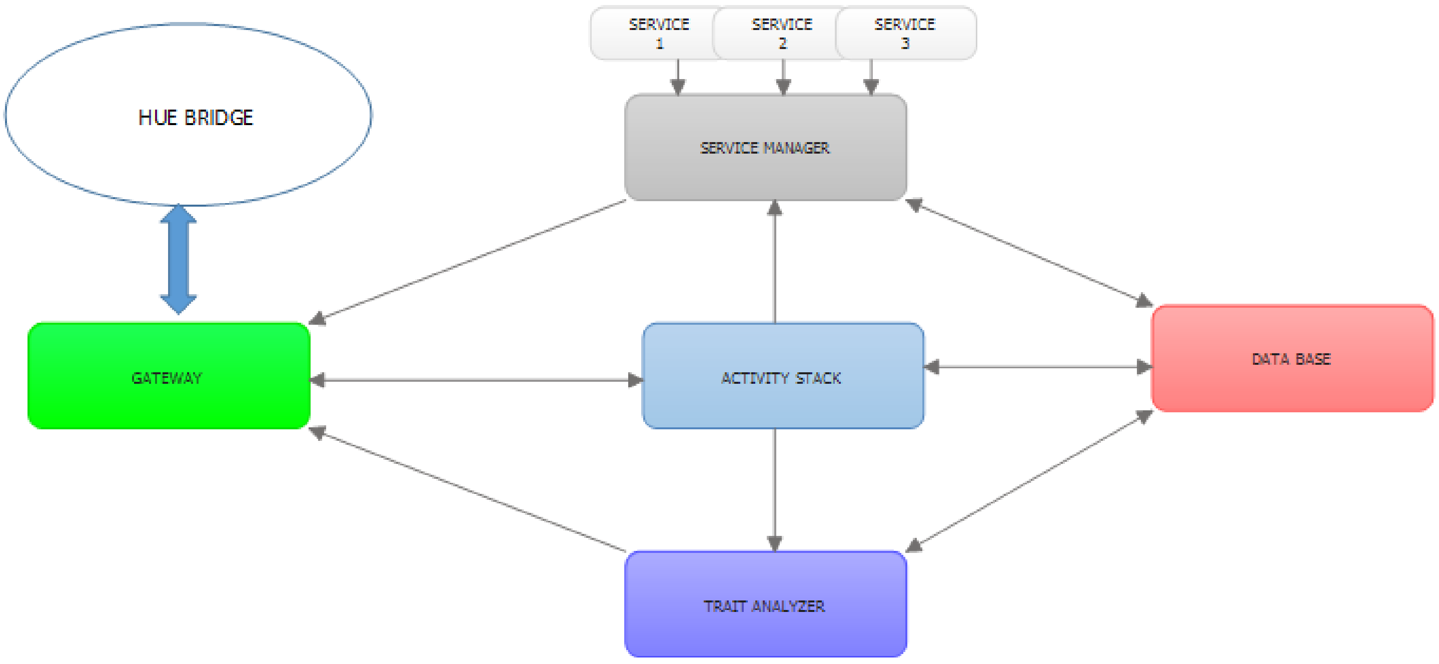

Diagram 1.

Upon calling an activity, the Activity Stack will send a request to the Database component to obtain necessary for that activity information or will tell it to change already existing data. The Database component depending on the request will execute the appropriate SQL query(s) and then will send requested data back to the activity or performs appropriate data manipulations within the database.

Along with supporting communication with the database, the Activity Stack notifies about user's changes and/or requests to the appropriate components including the Service Manager, Gateway, and Trait Analyzer. These components also perform additional notifications to the other components as needed. Example of such communication between these components is presented below on the Diagram 2.

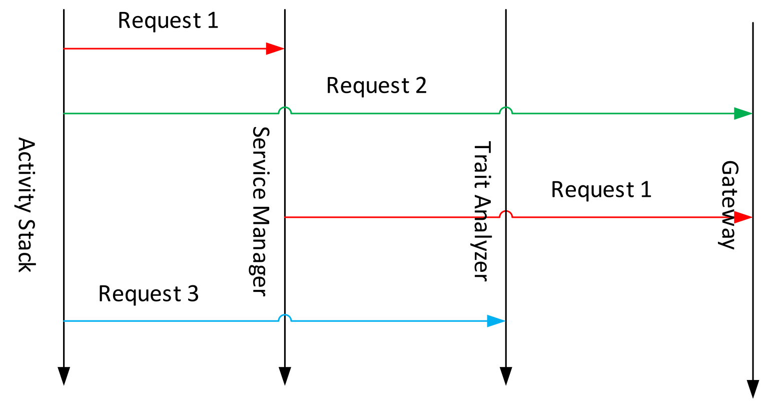

Diagram 2.

The Server Manager upon request from the Activity Stack will issue an appropriate service. That service will listen for a right condition(s) and then will trigger an appropriate ***Bright Light Event*** (see Box 1.). That event will be sent to the Gateway which will interpret it into a request or series of requests and will attempt to send them via WI-FI to the Hue Bridge.

If the Gateway fails to send a request or never receives an acknowledgment from the Hue Bridge, it will attempt to send the request again defined "*n*" amount of times until reporting an error to the user.

Bright Light Event defines a change of a single light or group of lights; caused either directly by the user or by a user-defined Trigger

Box 1.

The Trait Analyzer will communicate with the Database component in order to intelligently sort data for the Activity Stack to display to the user and/or trigger certain events based on a pattern of user's previous choices.