Bee Lazi

Version 2.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 04/12/2016 | 1.0 | First Draft | Gia-An To |
| 11/12/2016 | 2.0 | Update Use-case and Deployment View | Gia-An To |
| 4/1/2017 | 3.0 | Final Check | Gia-An To |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose: 4

1.2 Scope: 4

1.3 References: 4

2. Architectural Goals and Constraints 4

3. Use-Case Model 5

3.1 Create Account 5

3.2 User Authentication 5

3.3 Log In 5

3.4 Log out 5

3.5 View sugesstion 5

3.6 Update Personal’s Schedule 5

3.7 View detailed-information 5

3.8 Vote/ Rate suggestion 5

3.9 Guiding suggestion 5

3.10 Build up characteristic 6

3.11 Manage User’s Account 6

3.12 Manage Application Data 6

4. Logical View 6

4.1 Application Specific Layer: 6

4.2 System Processing Layer: 6

4.3 System Management Layer: 7

4.4 Middleware Layer: 7

5. Deployment 7

6. Implementation View 7

# Introduction

## Purpose:

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system**.**

## Scope:

This Software Architecture Document provides an architectural overview of the Smart Suggesting System. The Smart Suggesting System is being developed by APCS\_01 to support activies suggestion for officers, travelers or alumni. The majority of the sections have been extracted from the Software Architecture Document template.

## References:

<https://github.com/skyfishjy/android-ripple-background>

<https://github.com/Diolor/Swipecards>

<https://github.com/florent37/MaterialTextField>

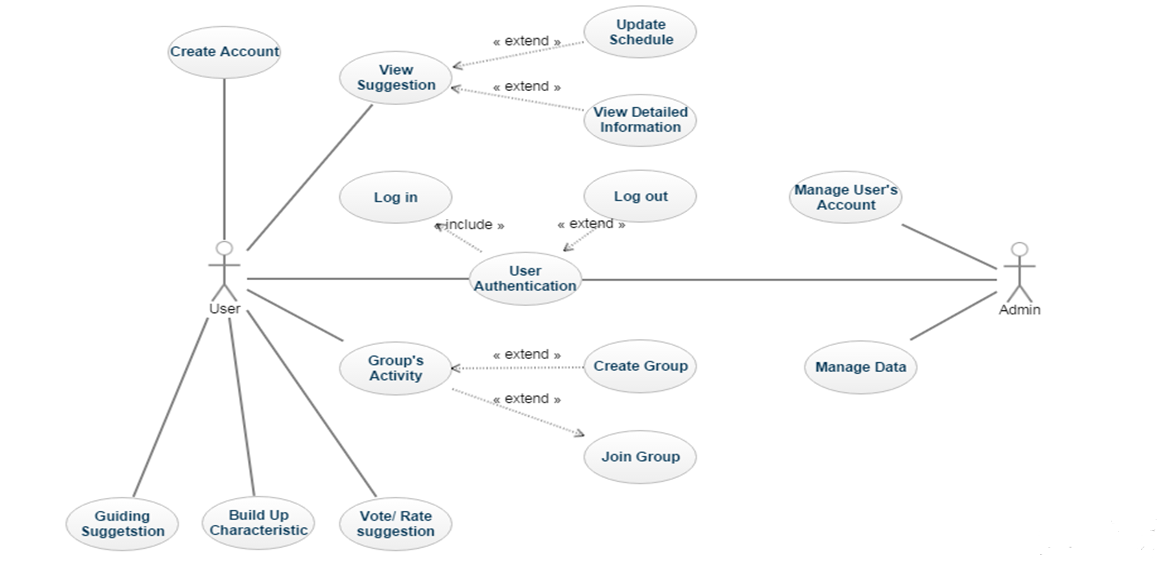
<https://github.com/dmytrodanylyk/android-process-button>

<https://github.com/wrapp/floatlabelededittext>

# Architectural Goals and Constraints

Breaking down the project into easy-to-handle components which the communications between them are comprehensive is our goal here. Retaining the reuse of components, especially the learning model and the design components, is prior. So two progress performing parallel are well-known, coding the learning model and the design which the left one uses code Python, meanwhile, the other develops in Android Studio. There is a server component which goes between these two big components solving the task of connecting them cause of two different supporting environments, Python and Android Studio. Therefore, now the team structure is forming three flows, one is aiming for learning model, two is for designing, and three is for constructing server.

# Use-Case Model



## Create Account

Brief Description: This use-case describes how the user create new account to log in application

## User Authentication

Brief description: This use-case describes how user-authentication works.

## Log In

Brief description: This use-case describes choices for user and admin to sign in application. This use-case is a ≪include≫ of use-case User Authentication.

## Log out

Brief description: This use-case describes how user logs out application. This use-case is a ≪extend≫ of use-case User Authentication.

## View sugesstion

Brief description: This use case describes how the app suggest to user.

## Update Personal’s Schedule

Brief description: This use-case describes how user updates personal’s schedule.

## View detailed-information

Brief description: This use case describes how user view detail-information of one suggested action.

## Vote/ Rate suggestion

Brief description: This use case describes how user respond to a suggestion.

## Guiding suggestion

Brief description: This use case gives a suggestion based on a sequence of questions smartly chosen for customer.

## Build up characteristic

Brief description: The system gives the user the suggestion based on the selected characteristic.

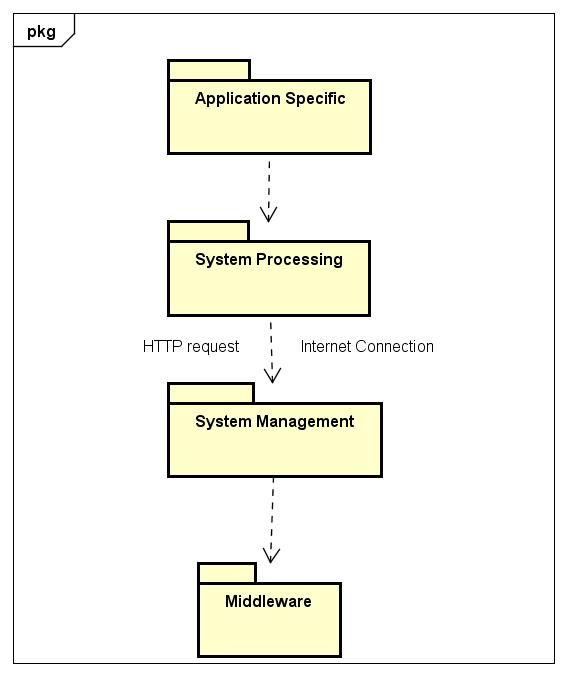
## Manage User’s Account

Brief description: This use-case describes how admin manages user’s account.

## Manage Application Data

Brief description: This use-case describes how admin manages application data.

# Logical View



## Application Specific Layer:

This layer supports application interface. A guide user interface is supported for easy use. Users interact with system through this layer.

## System Processing Layer:

This layer supports the processing of application includes: load request from users and receive response from system, make suggestions based on characteristic, improve suggestions automatically each day based on user’s choice, etc.

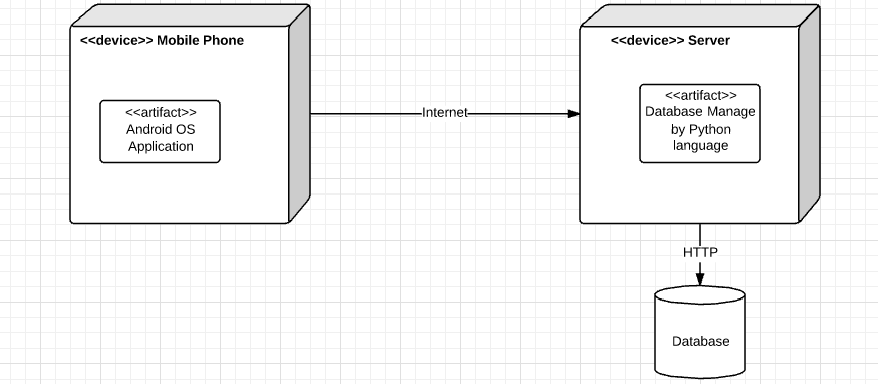
## System Management Layer:

This layer supports administrators to add new users, delete users, modify information of the suggestions made by application, and manage behaviors of users. This layer is also used to receive request from above layers and transfer requests to middleware layer for response.

## Middleware Layer:

This layer supports interaction and management of database.

# Deployment



# Implementation View