NGUYEN VAN DAT

19521347gm.uit.edu.vn

*Prepared for*

Ahihi Project

**Version 0.7.0**

**SOFTWARE REQUIREMENTS SPECIFICATION**

Prepare for Ahihi project

**Revision and Sign Off Sheet**

**Change Record**

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Version** | **Change reference** | **Date** |
| Nguyễn Văn Đạt | 0.5.0 | Complete requirement direction |  |
|  |  |  |  |

**Reviewers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Company** | **Version** | **Position** | **Date** |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of Contents**

Contents

[1. Introduction 4](#_Toc90873158)

[1.1. Purpose 4](#_Toc90873159)

[1.2. Scope 4](#_Toc90873160)

[1.3. Intended Audiences and Document Organization 4](#_Toc90873161)

[1.4. References 5](#_Toc90873162)

[2. Functional Requirements 5](#_Toc90873163)

[2.1. Use Case Description 5](#_Toc90873164)

[UC1: Login 5](#_Toc90873165)

[UC2: Login by APIs 6](#_Toc90873166)

[UC3: Forgot Password 7](#_Toc90873167)

[UC4: Register 9](#_Toc90873168)

[UC5: Search 10](#_Toc90873169)

[UC6: Write Post 11](#_Toc90873170)

[UC7: Write Comment 12](#_Toc90873171)

[UC8: Like 14](#_Toc90873172)

[UC9: Dislike 15](#_Toc90873173)

[UC10: Report 16](#_Toc90873174)

[UC11: Chat 17](#_Toc90873175)

[UC12: Add Services 19](#_Toc90873176)

[UC13: Book Services 20](#_Toc90873177)

[UC14: Watch News 21](#_Toc90873178)

[UC15: Block News 22](#_Toc90873179)

[UC16: Add Friend 24](#_Toc90873180)

[UC17: Delete Friend 25](#_Toc90873181)

[UC18: View Friend List 26](#_Toc90873182)

[UC19: Change Password 27](#_Toc90873183)

[UC20: Lock Account 28](#_Toc90873184)

[UC21: Change Account Information 29](#_Toc90873185)

[UC22: Help 31](#_Toc90873186)

[UC23: Create Page 32](#_Toc90873187)

[UC24: Create Group 33](#_Toc90873188)

[UC25: Switch Page/Group 34](#_Toc90873189)

[UC26: Add Admin 36](#_Toc90873190)

[UC27: Delete Admin 37](#_Toc90873191)

[UC28: Get Admin List 38](#_Toc90873192)

[UC29: Lock Page/Group 38](#_Toc90873193)

[UC30: Open Page/Group 40](#_Toc90873194)

[UC31: Delete Page/Group 41](#_Toc90873195)

[UC32: Edit Page/Group 42](#_Toc90873196)

[UC33: Ranking 43](#_Toc90873197)

[UC34: View Member 44](#_Toc90873198)

[UC35: Delete Member 45](#_Toc90873199)

[UC36: Add Member 46](#_Toc90873200)

[UC37: Register ADS 47](#_Toc90873201)

[UC38: View ADS 49](#_Toc90873202)

[UC39: Cancel Ads 49](#_Toc90873203)

[2.2. List Description 51](#_Toc90873204)

[2.3. View Description 51](#_Toc90873205)

[3. Non-functional Requirements 51](#_Toc90873206)

[User Access and Security 51](#_Toc90873207)

[Performance Requirements 52](#_Toc90873208)

[4. Other Requirements 53](#_Toc90873209)

[Archive Function 53](#_Toc90873210)

[5. System Requirements 53](#_Toc90873211)

[Ahihi Sites 53](#_Toc90873212)

[Ahihi Lists 53](#_Toc90873213)

[Ahihi Groups 53](#_Toc90873214)

[Technical Concern 54](#_Toc90873215)

[Wireframe 54](#_Toc90873216)

[6. Appendixes 54](#_Toc90873217)

[Glossary 54](#_Toc90873218)

[Mapping to Notes Application 54](#_Toc90873219)

[Messages 54](#_Toc90873220)

[Issues List 55](#_Toc90873221)

# Introduction

# Purpose

This Software Requirements Specification and Design document contains the software requirements to migrate the application from Lotus Notes/Domino to a new target platform e.g. Ahihi, Java or PHP and detailed design for migrated application on target platform. Firstly, this document along with the Notes database(s) and other reference documents are complete requirements to perform a migration from Domino environment to target platform. Secondly, it defines, technically, how applications will operate. Developers will base on this document to conduct a development plan, task assignment and implementation of the new application.

# Scope

This document is prepared for the application, in scope of the project Ahihi.

# Intended Audiences and Document Organization

This document is intended for:

* Development team: Responsible to develop detailed design, implement and perform unit test, integration test and system test for the migrated application
* Data Migration team: Responsible to create data migration scripts and perform data migration for the application.
* Documentation Team: Responsible to writing User Guide for the application.
* UAT team: Responsible to conduct user acceptance test sessions with end users.

Below are main sections of the document:

* **1. Introduction**: This section describes the general introduction of this document.
* **2. Functional Requirements**: This section describesthe functional requirements in detail.
* **3. Non-functional Requirements:** This section describes the non-functional requirements of this application such as user access and security, interfaces, screens and performance.
* **4. Other Requirements:** This section describes other requirements such as archive or security audit function.
* **5.** **Ahihi Application Design:** This section describes the design of Ahihi application.
* **6. Appendixes**: This section describes other requirements for this application and other supporting information for this document**.**
* NOTE:Please refer to section 6.1 for all acronyms and abbreviations you may encounter within this document.

# References

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Title** | **Version** | **File Name / Link** | **Description** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

# Functional Requirements

# Use Case Description

### UC1: Login

|  |  |
| --- | --- |
| **Name** | **Login** |
| **Description** | This use case allows User to login into Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click login button |
| **Pre-condition** | * [User’s Device ID status] == 0; |
| **Post-condition** | * User’s Device ID status updated |

#### Activities Flow



*Figure 1: Activities Login Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Login” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Login”, the system will validate user account in account’s Textbox. If user’s account is invalid, the system will show error message; else, the system will save inputted information, submit the device status to appropriate approver enter Ahihi and update the status as the following:   * If [Username] not exists, show MSG3 * If [Username] or [Password] is empty, show MSG10 * If [Username] == true   + If [Password] == findPasswordByUserName(), Enter Ahihi, Set User’s device status = true  + Else show MSG 3 |

### UC2: Login by APIs

|  |  |
| --- | --- |
| **Name** | **Login by APIs** |
| **Description** | This use case allows User to login into Ahihi app by using other provider method. |
| **Actor** | User |
| **Trigger** | * When User click provider button |
| **Pre-condition** | * [User’s Device ID status] == 0; |
| **Post-condition** | * User’s Device ID status updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 2: Activities Login by APIs Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Login” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on provider button, the system will use provider’s business to help user login. If user’s provider’s account is not exists in database, the system save provider’s account data. Then the system will save inputted information, submit the device status to appropriate approver enter Ahihi and update the status as the following:   * If [userClickOn] == [Provider] do Provider’s bussiness * Try (I/O Exception) catch (return Login Screen, userDeviceStatus = false)   + If [userDeviceStatus] == true, Enter Ahihi. |

### UC3: Forgot Password

|  |  |
| --- | --- |
| **Name** | **Forgot Password** |
| **Description** | This use case allows User to by using email. |
| **Actor** | User |
| **Trigger** | * When User click Forgot Password button |
| **Pre-condition** | * [User’s Login status] == 0; |
| **Post-condition** | * Database update new password |

#### Activities Flow

Diagram

Description automatically generated

*Figure 3: Activities Forgot Password APIs Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * When user clicks on Forgot Password button, the system display “Forgot Password” screen. * When user click on Get Password URL, the system will display “Get Password” Screen when User clicked |
| *(2)* | *BR2* | **Submitting Email Rules:**  When User click on ‘Confirm’ button. The system will check Email Textbox and send Get Password URL to User by mail as the following:   * If [buttonConfirm] == true   + If [Email] != isEmailFormat(), display MSG 5.  + If [Email] is exists, send Get Password URL.   * “Get Password URL” email must contains message:   + “Hello {UserName},\n Click link below to send change Password\n{ Get Password URL}” |
| *(3)* | *BR3* | **Change Password Rules:**  When User click on ‘Confirm’ button. The system will check ‘New Password’ Textbox. System will change and save new password to database that appropriate approver to enter Ahihi as the following:   * If [buttonConfirm] == true && [PasswordFormat] == true, save password to Database, return ‘Login’ Screen   + PasswordFormat require text that not contains special keyword and have minimum length = 8.   * Else, show MSG 8 |

### UC4: Register

|  |  |
| --- | --- |
| **Name** | **Register** |
| **Description** | This use case allows User to register an Ahihi account. |
| **Actor** | User |
| **Trigger** | * When User click register button |
| **Pre-condition** | * [User’s Device ID status] == 0; |
| **Post-condition** | * New account added to Database |

#### Activities Flow

Diagram

Description automatically generated

*Figure 4: Activities Register Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(2)* | *BR1* | **Loading Screen Rules:**   * The system loads “Register” screen. |
| *(4)* | *BR2* | **Submitting Rules:**  When user clicks on register button, the system will use provider’s business to help user register an account. Then the system will save inputted information, submit the device status to appropriate approver enter Ahihi and update the status as the following:   * If [buttonConfirm] == true, validate input data   + If [UserName] == empty(), mark red. Else  + If [Password] == empty(), mark red. Else  + If [Social ID] == empty(), mark red. Else  + If [ConfirmPassword] == empty(), mark red. Else  + If [Email] == empty(), mark red. Else  + If PasswordFormat == false, mark red and display message “Password must contains 8 keyword at least and have no special keyword” below. Else  + If EmailFormat == true, mark red and display message “Email format: abc@xyz” below. Else  + Send confirm email. Confirm email must contains message: “Hello {UserName},\n Click link below to login\n{Confirm URL}”   * If User click on URL   + Try (I/O Exception) Catch (SendAnotherURL)  + If [Confirm URL] == true, run query to insert Account, return ‘Login’ Screen |

### UC5: Search

|  |  |
| --- | --- |
| **Name** | **Search** |
| **Description** | This use case allows User to search data in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click search button |
| **Pre-condition** | * [User’s Device ID status] == 1; |
| **Post-condition** | * System display searched data |

#### Activities Flow

Diagram

Description automatically generated

*Figure 5: Activities Search Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Search” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on search button, the system will display Search screen. System will search information as the following:   * If searchUsername() == true, show Username list (searched) * If searchPage() == true, show Page list (searched) * If searchContent() == true, show Page list (searched) * If searchEmpty == true, show MSG 10   + Sort(Filter()) |

### UC6: Write Post

|  |  |
| --- | --- |
| **Name** | **Write Post** |
| **Description** | This use case allows User to write a Post in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User Post Textbox |
| **Pre-condition** | * Current Screen = Home Screen |
| **Post-condition** | * New Post updated to Database |

#### Activities Flow

Diagram

Description automatically generated

*Figure 6: Activities Write Post Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Post” view. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on Confirm button, the system will help user to post a post as the following:   * If PostSaved() != true, show MSG12. Else * Save post to database, display post |

### UC7: Write Comment

|  |  |
| --- | --- |
| **Name** | **Login** |
| **Description** | This use case allows User to comment at a post in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click comment button |
| **Pre-condition** | * User’s post comment must be user’s friend. |
| **Post-condition** | * Post comment updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 7: Activities Write Comment Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Comment” view. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Comment”, system will save inputted information update data as the following:   * If CommentSaved() != true, show MSG12. Else   Save comment to database, display post |

### UC8: Like

|  |  |
| --- | --- |
| **Name** | **Like** |
| **Description** | This use case allows User to like a post in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click like button |
| **Pre-condition** | * User is on News board; |
| **Post-condition** | * Post liked updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 8: Activities Like Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “News” board. |
| *(2)* | *BR2* | **Confirm Rules:**  When user clicks on “Like”, the system will update the data as the following:   * If ThisUserLikeThisPost() == null   + Like++  + ThisUserLikeThisPost() == User.ID   * Else if ThisUserLikeThisPost() != null   + Like--  + ThisPostLikeList.Delete(ThisUserLikeThisPost()) |

### UC9: Dislike

|  |  |
| --- | --- |
| **Name** | **Dislike** |
| **Description** | This use case allows User like a post in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click dislike button |
| **Pre-condition** | * User is on News board; |
| **Post-condition** | * Post dislike updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 9: Activities Dislike Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “News” screen. |
| *(2)* | *BR2* | **Confirm Rules:**  When user clicks on “Like”, the system will update the data as the following:   * If ThisUserDislikeThisPost() == null   + Dislike++  + ThisUserDislikeThisPost() == User.ID   * Else if ThisUserDislikeThisPost() != null   + Dislike--  + ThisPostDislikeList.Delete(ThisUserDislikeThisPost()) |

### UC10: Report

|  |  |
| --- | --- |
| **Name** | **Report** |
| **Description** | This use case allows User like a post or a page/group in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Report button |
| **Pre-condition** | * User is on News board; |
| **Post-condition** | * Post Report updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 10: Activities Report Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “News” screen. |
| *(2)* | *BR2* | **Confirm Rules:**  When user clicks on “!!!”, the system will update the data as the following:   * If ThisUserReportThisPost() == null   + Report++  + ThisUserReportThisPost () == User.ID   * Else if ThisUserReportThisPost () != null   + Report--  + ThisPostReportList.Delete(ThisUserReportThisPost ()) |

### UC11: Chat

|  |  |
| --- | --- |
| **Name** | **Chat** |
| **Description** | This use case allows User to chat to other’s account in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click chat button |
| **Pre-condition** | * [User’s Device ID status] == 1; |
| **Post-condition** | * User’s Chat data updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 11: Activities Chat Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Chat” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Chat”, the system will validate user account in account’s Textbox. System will decode message following user data and display to user as the following:   * Show ListChatByUserID(SelectedUserID)   + Decode information follow User   * If [onSend] == true   + Push data to database  + If (pushSuccess) == true, Show ListChatByUserID(SelectedUserID). Else show MSG 12 |

### UC12: Add Services

|  |  |
| --- | --- |
| **Name** | **Add services** |
| **Description** | This use case allows User to add services into Page in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Add Services button |
| **Pre-condition** | * [User’s roles] == [PageRole]; |
| **Post-condition** | * Page’s Services updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 12: Activities Add Services Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Services” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Add Services”, update the services list as the following:   * If [Content] == null, Content = ContentByDefault * If [Image] == null, Image = ImageByDefault * If [Value] == null, Value = ValueByDefault * SaveServices()   + If servicesIsSaved() == false, show MSG 12 |
| *(3)* | *BR3* | **Submitting Constraints**   * ContentByDefault = ‘Empty’ * ImageByDefault = ‘EmptyImage’ * ValueByDefault = 0 |

### UC13: Book Services

|  |  |
| --- | --- |
| **Name** | **Book Services** |
| **Description** | This use case allows User to login into Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Book Services button |
| **Pre-condition** | * [UserRoles] == [User]; |
| **Post-condition** | * User communicate with Page |

#### Activities Flow

Diagram

Description automatically generated

*Figure 13: Activities Book Services Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading View Rules:**   * The system loads “Services” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Book”, the system will collect service and send to Page as the following:   * CollectServicesWhereQuantityBiggerZero() * Notify.Page(Page.ID).Services(CollectServicesWhereQuantityBiggerZero)   **Notification Rules:**   * Notify message: “{UserName} book {Services}” |
| *(3)* | *BR3* | **Notification Services:**   * Notification services must contain:   + Service name + Quantity + Total Value  + Total service + Total quantity + Total Value |

### UC14: Watch News

|  |  |
| --- | --- |
| **Name** | **Watch News** |
| **Description** | This use case allows User to login into Ahihi app. |
| **Actor** | User |
| **Trigger** | * Current Screen == Main Screen |
| **Pre-condition** | * User login the system; * User view another User profile * User view group profile * User view Page profile |
| **Post-condition** | * User’s Device ID status updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 14: Activities Login Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Main” screen * The system loads “New” View. |
| *(2)* | *BR2* | **Displaying Rules:**  When user is on main screen, the system gets News data from database and display to user |

### UC15: Block News

|  |  |
| --- | --- |
| **Name** | **Block News** |
| **Description** | This use case allows User to Block News in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click ‘X’ button |
| **Pre-condition** | * Current Screen == Main Screen; |
| **Post-condition** | * Database updated following user; |

#### Activities Flow

Diagram

Description automatically generated

*Figure 15: Activities Block News Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading View Rules:**   * The system loads “Main” screen * The system loads “Block” view. Block view include   + Cancel button  + Block button  + Hide Button |
| *(2)* | *BR2* | **Choice Rules:**  When user clicks on “X”, the system will separate into three choice for user as the following:   * If [onClick] == [Cancel], disappear Block view * If [onClick] == [Block]   + Save to Block List  + Block selected News source   * If [onClick] == [Hide]   + Save to Hide List  + Hide selected News |

### UC16: Add Friend

|  |  |
| --- | --- |
| **Name** | **Add Friend** |
| **Description** | This use case allows User to login into Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click add friend button |
| **Pre-condition** | * FriendToUser(This.User) == false; |
| **Post-condition** | * Database updated following user |

#### Activities Flow

Diagram

Description automatically generated

*Figure 16: Activities Add Friend Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Profile” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Add Friend”, the system will notify selected user as the following:   * If AddFiend.isPressed == true   + Save to Database  + Notify selected account  + If [NotifySelectedAccout.Agree] == true.   * Save to database * Notify to selected user   + Else if [NotifySelectedAccout.Agree] == false   * Save to database * Notify to selected user |

### UC17: Delete Friend

|  |  |
| --- | --- |
| **Name** | **Delete Friend** |
| **Description** | This use case allows User to Delete a friend in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click unfriend button |
| **Pre-condition** | * User.IsFriend(SelectAccount) == true; |
| **Post-condition** | * User.IsFriend(SelectAccount) == false |

#### Activities Flow

Text

Description automatically generated

*Figure 17: Activities Delete Friend Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Provide” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Unfriend”, the system will delete friend as the following:   * If [Unfirend.IsPressed] == true, User.IsFriend(SelectAccount) == false * Save to database |

### UC18: View Friend List

|  |  |
| --- | --- |
| **Name** | **View Friend List** |
| **Description** | This use case allows User to watch friend list from Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Friends button |
| **Pre-condition** | * [User’s Device ID status] == 1; |
| **Post-condition** | * None |

#### Activities Flow

Diagram

Description automatically generated

*Figure 18: Activities View Friend List Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Priend” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Profie” -> “Friend”, the system will get data from database and display friend list to User |

### UC19: Change Password

|  |  |
| --- | --- |
| **Name** | **Chang Password** |
| **Description** | This use case allows Users to change their password in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click login button |
| **Pre-condition** | * [User’s Device ID status] == 1; |
| **Post-condition** | * User’s Device ID status updated |

#### Activities Flow

Diagram, schematic

Description automatically generated

*Figure 19: Activities Change Password Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Setting” screen. * The system loads “Security” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Change Password”, change password as the following:   * If [ChangePassword.isPressed] == true   + If [CurrentPassword.IsEmpty] == true, mark Red and show MSG 11 (CurrentPassword). Else  + If [NewPassword.IsEmpty] == true, mark Red and show MSG 11(NewPassword). Else  + If [RightPasswordFormat] == true, show MSG 1. Else  + If [MatchNewPassword] == true, save new password to database |

### UC20: Lock Account

|  |  |
| --- | --- |
| **Name** | **Lock Account** |
| **Description** | This use case allows User to lock their Account in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click login button |
| **Pre-condition** | * [AccountLifeStatus] == 1; |
| **Post-condition** | * [AccountLifeStatus] == 0; |

#### Activities Flow

Diagram, schematic

Description automatically generated

*Figure 20: Activities Lock Account Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Setting” screen. * The system loads “Security” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Lock Account”, the system will validate user account in password Textbox. If user’s account is invalid, the system will show error message; else, the system will save inputted information, submit the account life status to in Ahihi and update the status as the following:   * If [Confirm.IsPressed] == true   + If (Password.IsEmpty) == true, show MSG 9. Else  + If (PasswordValid) == false, show MSG 8. Else  + Lock(Account)  + Save to database |

### UC21: Change Account Information

|  |  |
| --- | --- |
| **Name** | **Change Account Information** |
| **Description** | This use case allows User to change User’s Information in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Account button |
| **Pre-condition** | * [User’s Device ID status] == 1; |
| **Post-condition** | * Account data uppdated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 21: Activities Change Account Information Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Setting” screen. * The system load “Account” screen |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Confirm”, the system will validate user account in Password Textbox. If user’s account is invalid, the system will show error message; else, the system will save inputted information, submit the account detail to in Ahihi and update the data as the following:   * If [Confirm.IsPressed] == true   + If (Password.IsEmpty) == true, show MSG 9. Else  + If (PasswordValid) == false, show MSG 8. Else  + if (ValidTextBox() == true), Save to database  **Validate Rules:**   * ValidTextBox() will validate elements as following:   + Avatar: Image  + Name: String  + Age: int  + Email: String include ‘@’  + Social ID: int  + Address: String |
|  |  |  |

### UC22: Help

|  |  |
| --- | --- |
| **Name** | **Help** |
| **Description** | This use case allows User to access helping from Ahihi team. |
| **Actor** | User |
| **Trigger** | * When User click help button |
| **Pre-condition** | * [User’s Device ID status] == 1; * User have some problem |
| **Post-condition** | * User problem is solved |

#### Activities Flow

Diagram

Description automatically generated

*Figure 22: Activities Help Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Help” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Help”, the system will validate user account and connect to User as the following:   * User.SendHelpRequest * HelpTeam.GetRequest.GetID * Connect(HelpTeam.onFree,User) * Open ‘Chat’ Screen |

### UC23: Create Page

|  |  |
| --- | --- |
| **Name** | **Create Page** |
| **Description** | This use case allows User create a Page in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Create Page button |
| **Pre-condition** | * [User’s Device ID status] == 1; |
| **Post-condition** | * Page data uppdated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 23: Activities Create Page Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Create Page” screen. “Create Page” screen must include   + Page Avatar: Image  + Page Name: String  + Page Address: String  + Page Admin: User |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Confirm”, the system will validate user account in Create Page Form. If user’s account is invalid, the system will show error message; else, the system will save inputted information, submit the device status to appropriate approver enter Ahihi and update the status as the following:   * If [Avatar] == null, mark Red. Else * If [Name] == null, mark Red. Else * If [Address] == null, mark Red. Else * CreatePage(User, InputData) |

### UC24: Create Group

|  |  |
| --- | --- |
| **Name** | **Create Group** |
| **Description** | This use case allows User create a Group in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Create Group button |
| **Pre-condition** | * [User’s Device ID status] == 1; |
| **Post-condition** | * Group data updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 24: Activities Group Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Create Group” screen. “Create Group” screen must include   + Page Avatar: Image  + Page Name: String  + Page Address: String   * + Page Admin: User |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Confirm”, the system will validate user account in Create Group Form. If user’s account is invalid, the system will show error message; else, the system will save inputted information, submit the device status to appropriate approver enter Ahihi and update the status as the following:   * If [Avatar] == null, mark Red. Else * If [Name] == null, mark Red. Else * If [Address] == null, mark Red. Else   Create Group (User, InputData) |

### UC25: Switch Page/Group

|  |  |
| --- | --- |
| **Name** | **Switch Page/Group** |
| **Description** | This use case allows User to switch between Page/Group roles in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Switch button |
| **Pre-condition** | * [User’s Device ID status] == 1; |
| **Post-condition** | * User roles |

#### Activities Flow

Diagram

Description automatically generated

*Figure 25: Activities Switch Page - Group Flow*

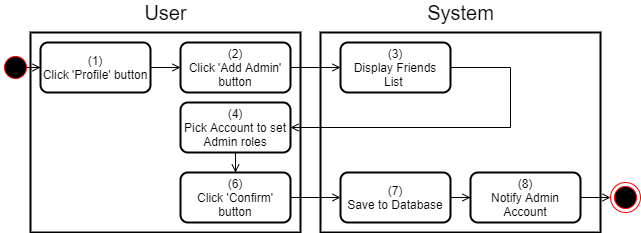
#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Switch Role” screen. |
| *(2)* | *BR2* | **Switch Rules:**  When user clicks on “Switch”, the system will validate user account, show Page/Group that user has specific roles in. Select a Page/Group the system will save inputted information, submit the user status to appropriate approver enter Page/Group roles as the following:   * DisplayList(Page.ListFromUser(User), Group.ListFromUser(User)) * User.SetRoles(SelectedItem) |

### UC26: Add Admin

|  |  |
| --- | --- |
| **Name** | **Add Admin** |
| **Description** | This use case allows User to Add Admin into a Page/Group in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Add Admin button |
| **Pre-condition** | * [UserRole] == [Page] || [UserRole] == [Group]; |
| **Post-condition** | * Page/Group Admin list uppdated |

#### Activities Flow



*Figure 26: Activities Add Admin Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Admin” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Add Admin”, the system will validate user account. Add new Admin as the following:   * Search User -> Display User List * Tick on User * If (Confirm.IsPressed) == true, AddSelectedAdmin() |

### UC27: Delete Admin

|  |  |
| --- | --- |
| **Name** | **Delete Admin** |
| **Description** | This use case allows User to Add Admin into a Page/Group in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Delete Admin button |
| **Pre-condition** | * [UserRole] == [Page] || [UserRole] == [Group]; |
| **Post-condition** | * Page/Group Admin list uppdated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 27: Activities Delete Admin Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Admin” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Delete Admin”, the system will validate user account. Add new Admin as the following:   * Search User -> Display Admin List * Tick on User   If (Confirm.IsPressed) == true, AddSelectedAdmin() |

### UC28: Get Admin List

|  |  |
| --- | --- |
| **Name** | **Get Admin List** |
| **Description** | This use case allows User to get Admin List in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Delete Admin button |
| **Pre-condition** | * [UserRole] == [Page] || [UserRole] == [Group]; |
| **Post-condition** | * Page/Group Admin list uppdated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 28: Activities Login Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Login” screen. |

### UC29: Lock Page/Group

|  |  |
| --- | --- |
| **Name** | **Lock Page/Group** |
| **Description** | This use case allows User to Lock Page/Group in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Lock Page/Group button |
| **Pre-condition** | * [UserRole] == [UserCreatePage] || [UserCreateUserRole] == [Group]; |
| **Post-condition** | * Page/Group status list updated |

#### Activities Flow

Diagram, schematic

Description automatically generated

*Figure 29: Activities Lock Page/Group Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Lock” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Lock”, the system will validate user account in Password Textbox. If user’s password is invalid, the system will show error message; else, the system will save inputted information, submit data to appropriate approver block Page/Group and update the database as the following:   * If [Password] not valid, show MSG 8. ELSE * LockPage() * Save to database |

### UC30: Open Page/Group

|  |  |
| --- | --- |
| **Name** | **Open Page/Group** |
| **Description** | This use case allows User to Lock Page/Group in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Lock Page/Group button |
| **Pre-condition** | * [UserRole] == [UserCreatePage] || [UserCreateUserRole] == [Group]; * Page/Group status = Locked; |
| **Post-condition** | * Page/Group status updated |

#### Activities Flow

Diagram, schematic

Description automatically generated

*Figure 30: Activities Open Page/Group Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Lock” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Open”, the system will validate user account in Password Textbox. If user’s password is invalid, the system will show error message; else, the system will save inputted information, submit data to appropriate approver block Page/Group and update the database as the following:   * If [Password] not valid, show MSG 8. ELSE * OpenPage() * Save to database |

### UC31: Delete Page/Group

|  |  |
| --- | --- |
| **Name** | **Delete Page/Group** |
| **Description** | This use case allows User to login into Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Lock Page/Group button |
| **Pre-condition** | * [User Role] == [User Create Page] || [User Role] == [User Create Group]; |
| **Post-condition** | * Page/Group Admin list updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 31: Activities Delete Page/Group Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Lock” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Delete”, the system will validate user account in Password Textbox. If user’s password is invalid, the system will show error message; else, the system will save inputted information, submit data to appropriate approver block Page/Group and update the database as the following:   * If [Password] not valid, show MSG 8. ELSE * Delete Page * Return to Main Screen (in user roles)   Save to database |

### UC32: Edit Page/Group

|  |  |
| --- | --- |
| **Name** | **Edit Page/Group** |
| **Description** | This use case allows User to Edit Page/Group in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Edit button |
| **Pre-condition** | * [User Role] == [User Create Page] || [User Role] == [User Create Group]; |
| **Post-condition** | * Page updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 32: Activities Edit Page/Group Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * If [User Roles] == [Group], The system loads “Group Profile” screen. “Group Profile” screen include   + Avatar: Image  + Name: String   * If [User Roles] == [Page], The system loads “Page Profile” screen. “Page Profile” screen include   + Avatar: Image  + Name: String  + Address: String |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Confirm, the system will save inputted information, submit the device status to appropriate approver enter Ahihi and update the status as the following:   * If [Confirm.IsPressed] == true   + If [User.Roles] == [Group], ValidateGroupEditedData()  + If [User.Roles] == [Page], ValidatePageEditedData() |

### UC33: Ranking

|  |  |
| --- | --- |
| **Name** | **Ranking** |
| **Description** | This use case allows system ranking Post in Ahihi app. |
| **Actor** | User |
| **Trigger** | * User click Ranking button |
| **Pre-condition** | * News in Database is not null |
| **Post-condition** | * Data Ranking Displayed |

#### Activities Flow

Diagram

Description automatically generated

*Figure 33: Activities Ranking Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Screen Rules:**  System show ‘Ranking’ Screen as the following:   * Show most reacted Post * Show most reacted Page |

### UC34: View Member

|  |  |
| --- | --- |
| **Name** | **View Member** |
| **Description** | This use case allows User to view Moupember in a Page/Gr Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Member button |
| **Pre-condition** | * [User’s Device ID status] == 1; |
| **Post-condition** | * None |

#### Activities Flow

Diagram

Description automatically generated

*Figure 34: Activities View Member Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Member” screen. |

### UC35: Delete Member

|  |  |
| --- | --- |
| **Name** | **Delete Member** |
| **Description** | This use case allows User to delete members in a Page or Group at Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Delete button |
| **Pre-condition** | * [User’s Roles] == [Admin Page] || [User’s Roles] == [Admin Group]; |
| **Post-condition** | * Page/Group members updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 35: Activities Delete Member Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Member” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Delete”, the system will validate user . Then, save inputted information, submit delete list to appropriate approver Delete Member Ahihi and update data as the following:   * Delete [SelectedMemberList]   + Delete SelectedMemberList on Database   * Try (I/O Exception) Catch (MSG 12, SelectedMemberList = null)   + Delete SelectedMemberUIElements |

### UC36: Add Member

|  |  |
| --- | --- |
| **Name** | **Add Member** |
| **Description** | This use case allows User to add members in a Page or Group at Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Add member button |
| **Pre-condition** | * [User’s Roles] == [Admin Page] || [User’s Roles] == [Admin Group]; |
| **Post-condition** | * Page/Group members updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 36: Activities Add Member Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Member” screen. |
| *(2)* | *BR2* | **Submitting Rules:**  When user clicks on “Delete”, the system will validate user . Then, save inputted information, submit delete list to appropriate approver Delete Member Ahihi and update data as the following:   * ADD [SelectedMemberList]   + ADD SelectedMemberList on Database   * Try (I/O Exception) Catch (MSG 12, SelectedMemberList = null)   + ADD SelectedMemberUIElements |

### UC37: Register ADS

|  |  |
| --- | --- |
| **Name** | **Register ADS** |
| **Description** | This use case allows User to register ADS for their Page Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click ADS button |
| **Pre-condition** | * [User’s Roles] == [Page]; |
| **Post-condition** | * User ADS package updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 37: Activities Register ADS Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “ADS” screen. |
| *(2)* | *BR2* | **Register Rules:**  To register an ADS package. The system must as the following:   * Show ADS\_PackageList * Get SelectPackage * Do Payment follow APIs, save payment data * If [PaymentData] != null, UpdateADSPackage() and save to database. Else Return ADS Screen |

### UC38: View ADS

|  |  |
| --- | --- |
| **Name** | **View ADS** |
| **Description** | This use case allows User to view ADS package in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click ADS button |
| **Pre-condition** | * [User’s Roles] == [Page]; |
| **Post-condition** | * None |

#### Activities Flow

Diagram

Description automatically generated

*Figure 38: Activities View ADS Flow*

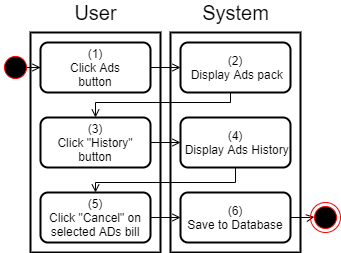
#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “ADS History” screen. “ADS History” screen includes   + Rented ADS package List  + Value of package  + Cancel button |

### UC39: Cancel Ads

|  |  |
| --- | --- |
| **Name** | **Cancel Ads** |
| **Description** | This use case allows User to Cancel current Ads in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click login button |
| **Pre-condition** | * [User’s Roles] == [Page]; |
| **Post-condition** | * Page’s Ads data updated |

#### Activities Flow



*Figure 39: Activities Cancel ADS Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “ADS History” screen. “ADS History” screen includes   + Rented ADS package List  + Value of package   * + Cancel button |
| *(2)* | *BR2* | **Cancel Rules:**  When user clicks on “Cancel”, the system will cancel ADS package as the following:   * User.Current.Ads.Cancel()   + CancelADS().PageID(User.ID) |

### UC40: View Notification List

|  |  |
| --- | --- |
| **Name** | **View Notification List** |
| **Description** | This use case allows User to view all notification in Ahihi app. |
| **Actor** | User |
| **Trigger** | * When User click Notify button |
| **Pre-condition** | * User logged in system |
| **Post-condition** | * Notification data updated |

#### Activities Flow

Diagram

Description automatically generated

*Figure 40: Activities Cancel ADS Flow*

#### Business Rules

|  |  |  |
| --- | --- | --- |
| **Activity** | **BR Code** | **Description** |
| *(1)* | *BR1* | **Loading Screen Rules:**   * The system loads “Notify” screen. “Notify” screen includes   + Name of object Notify  + Content of Notify   * + Time of Notify |
| *(2)* | *BR2* | **Notify Rules:**  When user clicks on a notify, the system will link to notify goal screen |

# List Description



# View Description



# Non-functional Requirements

## User Access and Security

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Actor** | **User** | | | | **Agency** |
| **Function** | **User Role** | **Page Role** | | **Group Role** |
| **Login** |  | |  |  |  |
| Login | x(\*) | |  |  | x(\*) |
| Login with APIs | x(\*) | |  |  |  |
| Forgot Password | x(\*) | |  |  |  |
| Register | x(\*) | |  |  |  |
| **Search** | x(\*) | | x(\*) | x(\*) |  |
| **Write Post** | x(\*) | | x(\*) | x(\*) |  |
| **Write Comment** | x(\*) | | x(\*) | x(\*) |  |
| **Like** | x(\*) | | x(\*) | x(\*) |  |
| **Dislike** | x(\*) | | x(\*) | x(\*) |  |
| **Report** | x(\*) | | x(\*) |  |  |
| **Chat** | x(\*) | | x(\*) |  |  |
| **Book Services** | x(\*) | |  |  |  |
| **Notify** | x(\*) | | x(\*) | x(\*) | x(\*) |
| **News** |  | |  |  |  |
| Watch News | x(\*) | | x(\*) | x(\*) |  |
| Block News | x(\*) | | x(\*) |  |  |
| **Friend** | x(\*) | |  |  |  |
| Add Friend | x(\*) | |  |  |  |
| Delete Friend | x(\*) | |  |  |  |
| View Friend | x(\*) | |  |  |  |
| **Setting** |  | |  |  |  |
| Lock | x(\*) | | x(\*) | x(\*) |  |
| Open | x(\*) | | x(\*) | x(\*) |  |
| Add services |  | | x(\*) |  |  |
| Delete services |  | | x(\*) |  |  |
| Change Account | x(\*) | | x(\*) | x(\*) |  |
| Help | x(\*) | | x(\*) | x(\*) |  |
| Create Page | x(\*) | |  |  |  |
| Create Group | x(\*) | |  |  |  |
| Switch Group/Page | x(\*) | | x(\*) | x(\*) |  |
| Add Admin |  | | x(\*) | x(\*) |  |
| Add services |  | | x(\*) | x(\*) |  |
| Delete Admin |  | | x(\*) |  |  |
| Delete Page/Group | x(\*) | |  |  |  |
| Edit | x(\*) | | x(\*) | x(\*) |  |
| View Member | x(\*) | | x(\*) | x(\*) |  |
| Delete Member |  | | x(\*) | x(\*) |  |
| **ADS** |  | |  |  |  |
| Register |  | | x(\*) |  | x(\*) |
| View History |  | | x(\*) |  |  |
| Cancel |  | | x(\*) |  |  |

X: User has full permission to do the action.

X(\*): User has permission to do the action on his own items.

X(\*\*): User has permission to do the action on items sent to him only.

X (1): reading permission is specified for each Error Form item regarding to Location of the item. For example, items, which have Location = “London”, are only read by employees of the location.

## Performance Requirements

**Number of users**

* Number of concurrent users: 1000
* Number of business user: 10000

**Data volume**

* Number of documents: 100
* Data growth rate: 100

Level of availability

Usage frequency

Cloud service: Google Cloud Platform

Location: China

Read-only Duration: 5 – 7 Hours

Read-only Timeframe: 3 AM

Maintenance Window: Mounty

Overall conversion timeline: 1 st

Other plans and activities: N/A

# Other Requirements

## Archive Function

|  |  |  |
| --- | --- | --- |
| **List** | **Actor** | **Condition** |
| *List name* | *Actor name* | *Actor* is able to archive item in “*list name*” list by created date. |

# System Requirements

## Ahihi Sites

|  |  |  |
| --- | --- | --- |
| **#** | **Site Name** | **Description** |
| *1* | **Ahihi.com.vn** | Domain of Ahihi |

## Ahihi Lists

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **List Code** | **List Name** | **Description** |
| *1* |  |  |  |
| *2* |  |  |  |
| *3* |  |  |  |

## Ahihi Groups

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Group Name** | **Permission Level** | **Description** |
| *1* | *User* |  | Normal user |
| *2* | *Page* |  | Page controlled by user have admin roles |
| *3* | *Group* |  | Group controlled by user have admin roles |

## 

## Technical Concern

* Growth Rate is low => less risk in performance
* Huge amount of data => saving/loading issue
* Too much content is in a single page
* Integrate to another systems

## Wireframe



# Appendixes

## Glossary

The list below contains all the necessary terms to interpret the document, including acronyms and abbreviations.

|  |  |
| --- | --- |
| **Term** | **Description** |
| *BR* | **B**usiness **R**ule |
| *CBR* | **C**ommon **B**usiness **R**ule |
| *DB* | Notes **D**ata**b**ase |
| *MSG* | **M**es**s**a**g**e |
| *UC* | **U**se **C**ase |
| *N/A* | **N**ot **A**vailable or **N**ot **A**pplicable, used to indicate when information in a certain section could not be provided because it does not apply to this application. |
| *UI* | **U**ser **I**nterface |
| *SRS* | **S**oftware **R**equirements **S**pecification |
| *TBD* | **T**o **b**e **d**etermined or **t**o **b**e **d**efined |

## Mapping to Notes Application

N/A

## Messages

This section describes the details of messages used in business rules e.g. error messages, confirmation messages, etc.

|  |  |  |
| --- | --- | --- |
| **Message Code** | **Message Content** | **Button** |
| MSG 1 | "You must specify a value for this required field" |  |
| MSG 2 | “This Account already exists.” |  |
| MSG 3 | “Username or Password wrong” |  |
| MSG 4 | "{String} must not contain special keyword." |  |
| MSG 5 | “Email wrong format.” |  |
| MSG 6 | "Are you sure about {String}, click ‘Confirm’ to Confirm." | Confirm/ Cancel |
| MSG 7 | “Error, please login again." |  |
| MSG 8 | "Wrong Password" |  |
| MSG 9 | “Require Password” |  |
| MSG 10 | “Cannot find any related data. Please try another’s." |  |
| MSG 11 | “{String} must not empty." |  |
| MSG 12 | “There seems to be a problem with the connection. Please connect to your network.” |  |
| MSG 13 | “Error payment, please try again!” |  |
| MSG 14 | “Name contains a-z and A-Z." |  |
| MSG 15 | “Social ID keyword accepted format: ‘1234567890’." |  |

## Issues List

N/A