****

**ADDIS ABABA UNIVERISTY**

**ADDIS ABABA INSTITUTE OF TECHNOLOGY**

**Center of Information Technology and Scientific Computing**

*Department of Software Engineering*

**Software engineering II**

Software Requirement Specification

**Prepared By:**

1. **Aman Bereket ATR/9348/08**
2. **Biya Girma ATR/7547/08**
3. **Estifanos sisay NSR/9401/08**
4. **Hena Fufa ATR/3750/08**
5. **Hermella Frew ATR/1689/08**
6. **Mihret Tamene ATR/3534/08**
7. **Oromia Godanna ATR/6053/08**
8. **Yohannes Fassil ATR/4122/08**

**Submitted to: Mr.** Natnael Argaw

Date: March 23/2018

Addis Ababa

Ethiopia

**Software Requirement Specifications**

(SRS)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| March 23 | Version 1 | Team members | Draft |
| April 8 | Version 2 | Team members | First revision |
| April 23 | Version 3 | Team members | Final revision |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Natnael Argaw | Instructor, ITSE |  |

Table of Contents

Revision History…………………………………………………………………………………………………… i

Document Approval……………………………………………………………………………………………... i

List of Tables……………………………………………………………………………………………………….. ii

List of Figures……………………………………………………………………………………………………. ii

Definitions, Acronyms, and Abbreviations……………………………………………………………. iii

Declaration………………………………………………………………………………………………………….. iv

1. **Introduction………………………………………………………………………….. 1**
   1. Purpose………………………………………………………………………………………………… 1
   2. Scope……………………………………………………………………………………………………. 1
   3. Overview……………………………………………………………………………………………….. 2
2. **General Description………………………………………………………………. 3**
   1. Product perspective………………………………………………………………………………. 3
   2. Product functions………………………………………………………………………………….. 3
   3. users characteristics……………………………………………………………………………… 3
   4. general constraints………………………………………………………………………………... 4
   5. Assumptions and dependencies…………………………………………………………….. 4
3. **Specific Requirements…………………………………………………………… 5**
   1. External Interface Requirements…………………………………………………………… 5
      1. User Interfaces…………………………………………………………………………… 5
      2. User Interface Prototype…………………………………………………………….. 8
      3. Hardware Interfaces…………………………………………………………………… 12
      4. Software Interfaces…………………………………………………………………….. 12
      5. Communication Interfaces…………………………………………………………... 12
   2. Functional Requirements……………………………………………………………………….. 13
      1. FR-01 Registration……………………………………………………………………….. 13
      2. FR-02 Login………………………………………………………………………………… 13
      3. FR-03 Customize Profile……………………………………………………………… 14
      4. FR-04 Choose mood…………………………………………………………………...... 15
      5. FR-05 Suggestion…………………………………………………………………………. 15
      6. FR-06 Rate……………………………………………………………………………………. 16
      7. FR-07 Set Profile Picture………………………………………………………………... 16
      8. FR-08 Reset password……………………………………………………………………. 17
      9. FR-09 Add/Remove photo……………………………………………………………… 17
      10. FR-10 Change Password………………………………………………………………… 18
      11. FR-11 View Place profile………………………………………………………………… 19
      12. FR-12 View on map………………………………………………………………………… 19
      13. FR-13 Logout………………………………………………………………………………….. 20
   3. Uses Cases………………………………………………………………………………………………….. 21
      1. UC-01 Customize profile…………………………………………………………………. 22
      2. UC-02 Add Profile Picture……………………………………………………………….. 22
      3. UC-03 Specify mood………………………………………………………………………... 23
      4. UC-04 request suggestion………………………………………………………………… 24
      5. UC-05 Rate……………………………………………………………………………………… 25
      6. UC-06 Remove/add photo………………………………………………………………. 25
      7. UC-07 Sign up…………………………………………………………………………………. 26
      8. UC-08 Sign in…………………………………………………………………………………… 27
      9. UC-09 Change Password………………………………………………………………….. 27
      10. UC-10 Reset Password…………………………………………………………………….. 28
      11. UC-11 Logout …………………………………………………………………………………. 29
   4. NON-FUNCTIONAL requirements……………………………………………………………….. 29
      1. Performance……………………………………………………………………………………. 29
      2. Reliability………………………………………………………………………………………… 30
      3. Availability………………………………………………………………………………………. 30
      4. Security………………………………………………………………………………………….. 30
      5. Maintainability……………………………………………………………………………….. 30
      6. Portability………………………………………………………………………………………. 30
   5. Inverse Requirements………………………………………………………………………….......... 31
   6. Design constraint………………………………………………………………………………………. 31
   7. Other Requirements………………………………………………………………………………….. 31
      1. Training-related Requirements……………………………………………………….. 31
4. **Change Management Process…………………………………………………………………….. 32**

**References…………………………………………………………………………………………………. 33**

# List of Tables

None

# List of Figures

Figure 3.1.1.1 Logical UI flow diagram

Figure 3.1.1.2.1 Login

Figure 3.1.1.2.2 Sign up (Individual)

Figure 3.1.1.2.3 Sign up (Place)

Figure 3.1.1.2.4 Home Screen (Individual)

Figure 3.1.1.2.5 Home Screen (Place)

Figure 3.1.1.2.6 Suggestion Screen

Figure 3.1.1.2.7 View Map

Figure 3.1.1.2.8 Customize Profile

Figure 3.1.1.2.9 View Profile

Figure 3.3 Use case diagrams

# 

# Definitions, Acronyms, and Abbreviations

**Mood:** A state or quality of feeling at a particular time.

**Place vibe**: A distinctive emotional atmosphere offered by the place.

**Individual:** A type of user which wants to visit places providing his current mood.

**Place**: Another type of user which wants to advertise his/her place by specifying a certain place vibe.

**SRS**: A description of a software system to be developed.

**Google Maps:** An application which is developed by Google and offers a map services.

**IOS:** An operating system developed by Apple Inc.

**Android:** Open source operating system made for mobile phones.

**GUI**: Graphical user interface.

**TCP/IP:** Transmission Control Protocol/ Internet Protocol.

**ER**: Entity relationship diagram, used for showing logical characteristics of a database.

# Declaration

We declare that this written submission represents our ideas in our own words and where others’ ideas or words have been included we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

# 1. Introduction

## 1.1 purpose

The purpose of this SRS is to provide a general overview and also a very detailed examination of what the system (Vibe) should offer its users and the constrains under which it operates.

The document first begins by giving a general overview of the application which should be suitable for anyone without technical knowledge of the system to read and understand it. The document then continues examining the requirements in detail giving those with some technical knowledge to be able to read it and implement or test or maintain it.

This SRS documentation is intended is to be proposed to any potential user for their approval and to give a brief description of the application as a reference for developing the first version of the system for the development team, to identify what it must do and the basic requirements it must fulfill.

## 1.2 scope

By the end of development a web based application, Vibe, will be produced with the following characteristics:

I. The application will allow a certain registered place to advertise itself in its own words which it thinks best describes the place’s ‘Vibe’ or feeling.

II. The application will allow any registered user to be able to find places based on their input of current mood or feeling.

III. The application will allow any registered place or user to upload picture including profile pictures.

IV. The application (if available) will display the places location in a map.

V. The application will allow any registered user to rate the places they visited.

VI. The application will enable the user to search for any registered palaces to view their profile, the service they provide and their location.

VII. In the contrast the application will not allow any type of communication between any two users, i.e. direct communication among individual users and between individuals and the registered places.

**Benefits of the system:**

* Provides a means for places to advertise themselves as products, by giving a list of services they provide and associating the emotional response they expect from their customers.
* Enables users to find new places based on their current mood.

The primary goal of the system is to create an easy way for users to navigate what’s around them based on their current mood and places to attract customers based on what they offer and think represents the places vibe.

## 1.3 Overview

With the above simple understanding of the application it is possible to continue to define more of the requirements of the system. In the next section, General Description, the requirements of the application from a more general and easy perspective, it tries to make the specific requirements in section 3 easier to understand and familiarizes readers with the system informally.

To achieve the above goal section 2 contains subsections such as **product perspective**, to familiarize the reader with the system by comparing this application with similar existing systems, **product functions,** to provide an overall summary of the functions this system provides, **user characteristics,** this subsection highlights the characteristics of expected end users of the application, and finally the **general constraints** subsection provides insights to the constraints put on the developer whilst making the application.

Section 3, Specific Requirements, then explores those requirements outlined in section 2 briefly.

# 2. General description

## 2.1 Product perspective

The application provided below tends to have few similarities:

I. **Google maps**: This system provides navigation of countries, cities, searching of place and more, which makes it similar with this system even though Google maps offers a very large scope of exploration.

II**. YELP**-Yelp [mobile app](https://en.wikipedia.org/wiki/Mobile_app), which publish [crowd-sourced](https://en.wikipedia.org/wiki/Crowd-sourced) reviews about local businesses, as well as the online reservation service Yelp Reservations. The company also trains small businesses in how to respond to reviews, hosts social events for reviewers, and provides data about businesses, including [health inspection](https://en.wikipedia.org/wiki/Health_inspection) scores.

## 2.2 Product functions

* Suggests a place to visit in based on the feeling they provide
* The application will let a registered user rate a place.
* The application will allow places to promote their services to appeal variety of customers through pictures and list of moods they elevate to meet customers state of conscious.

## 2.3 users characteristics

The users of the application can roughly be divided into two: the **Individual user** and **place user**.

* Both end users need to register in order to get or provide services.
* The individual user has general characteristics that resemble most social media users; where registration is required to find places he/she might enjoy.
* The place user is generally assumed to have organizational characteristics. The page being their canvas where they can paint a picture of their place through marketing and advertisement skills.

Structure of users resembles client-server design where individual user requests for a suggestion (as a client) and the place user is suggestion based on signed up mood.

## 2.4 general constraints

* Since the software interacts with users’ personal it’s critical it doesn’t leak or expose that sensitive information to unauthorized person. Thus developers should be aware of security threat.
* To improve User experience, developers should be cautious about performance and response time of the application.
* Connection with maps to provide exact location for user.

## 2.5 Assumptions and dependencies

* Assumes the user can express their current feeling through an emoji or an adjective.
* Assumes user has had exposure on how to use the internet or different social Medias.
* Assumes signed up place can be accessed through maps
* Since the system shall retrieve the current location of the user, it assumes the user has consistent internet connection.
* It is generally assumed that the Users computer is able to give information about its current location.

# 3. Specific Requirements

In the above section, we have tried to give the general overview of our entire system including what requirements it has. Now we will give the specific and detailed requirements which will be the guidelines to the design, implementation and testing phase. These requirements are specifically written to aid the developer in developing the system and the testers to test the system.

**Here is the list of things covered in this section:**

1. Requirements concerning External interfaces (i.e. UI, hardware and software Interfaces).
2. The functional and non-functional requirements of the application will be briefly dealt with.
3. Use cases will be illustrated.
4. We will briefly discuss our design constraints.
5. The requirements for the database needed will be discussed.

## 3.1 EXTERNAL Interface Requirements

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

### 3.1.1 User Interfaces

Link to the signup/register page should be provided for a first time users of the application. The application shall offer a sign up screen to its users. The signup page screens shall provide the necessary tools to successfully register a *place* and *individual*s respectively on the system. After successfully registered a user should be directed to the login page. The registration step (Screen) shall provide a way to accept the following inputs:

**From *Individuals*:**

1. Username , Password and email

**From *Places:***

1. Username and Password
2. Current location
3. Contact information (Phone number and email)
4. Associated Place vibe

If the user is not a first time user, he/she should be able to login. The login screen shall offer its users to insert their credentials (Username and Password) to login. After successful login, the system shall present the user with the appropriate home screen based on the type of user he/she is.

There shall be a home screen for both users after successful login. The home page is responsible for displaying a picture gallery, the total rank (given by people) and the profile and contact information for the place. The home page for the individual user should include their profile information, a list of previously suggested places, and an input to insert the emotion they are experiencing to request place suggestion.

The users shall be able to customize their profile by:

1. Providing their preferred profile picture.
2. Change their current password.
3. *Individuals*, by providing their current mood.
4. *Places*, by providing their picture gallery content.
5. *Places, by removing pictures from the gallery.*
6. *Places*, by providing their current place vibe.
7. *Places,* by modifying their current location

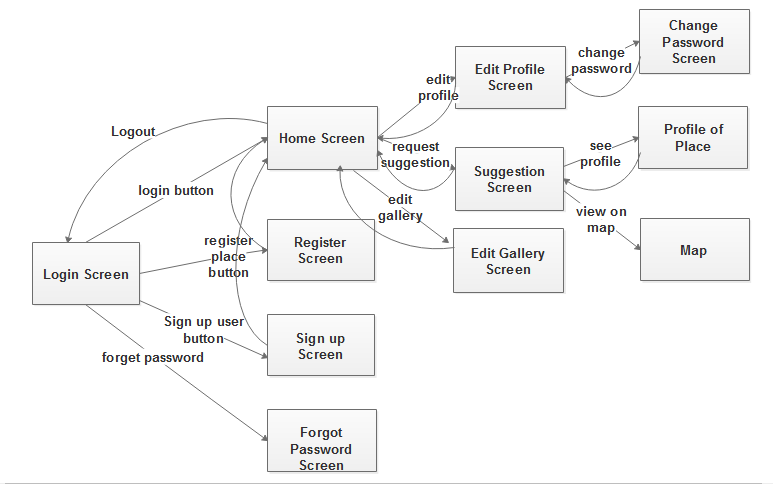
The GUI shall be able to display the suggested places in a map. There shall be an interface for the user to rate and the system **may** allow users to give comments about the *place* he was suggested after visiting it. An *Individual* shall be able to see the profile of the suggested places, view their picture gallery, view the user ratings and **might** see the comments associated with the place. The system **may** allow users to retrieve forgotten passwords through a forgotten password feature. Users who previously have experience using Google maps and any of the social media apps like what’s app, Messenger, or Telegram (on android or IOS) will not find it hard to use.

Each registered place should have a profile page which should be available for individual’s users when the place got suggested. It should include the whole profile of the place with the name, location, services and location.

There **may** be a screen for when the user forgot their passwords. It should enable the users to re-set a new password. After the password is successfully changed the user should be able to login with the new password.

#### 

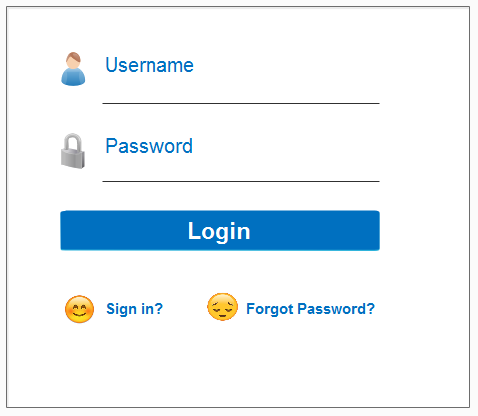
#### 3.1.1.1 Logical UI flow Diagram

****

*Figure 3.1.1.1 logical UI flow diagram*

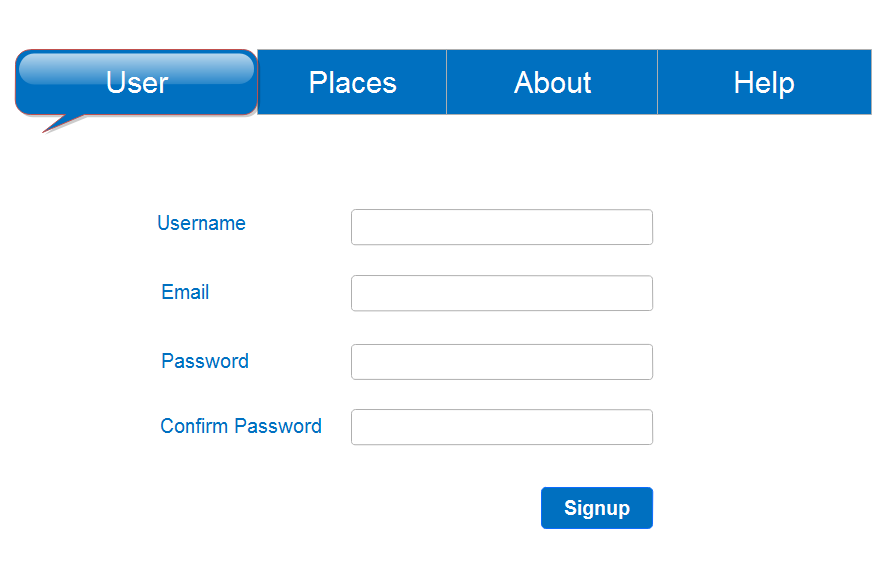
**3.1.1.2 User Interface Prototype**

**3.1.1.2.1 Login screen for users.**

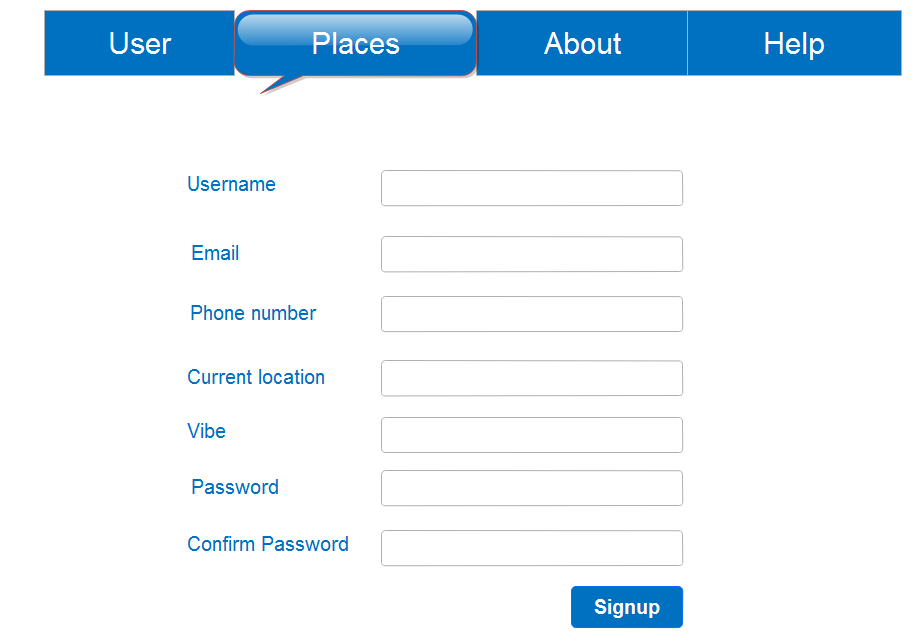


*Figure 3.1.1.2.1 Login screen*

**3.1.1.2.2 Signup screen for Individuals**

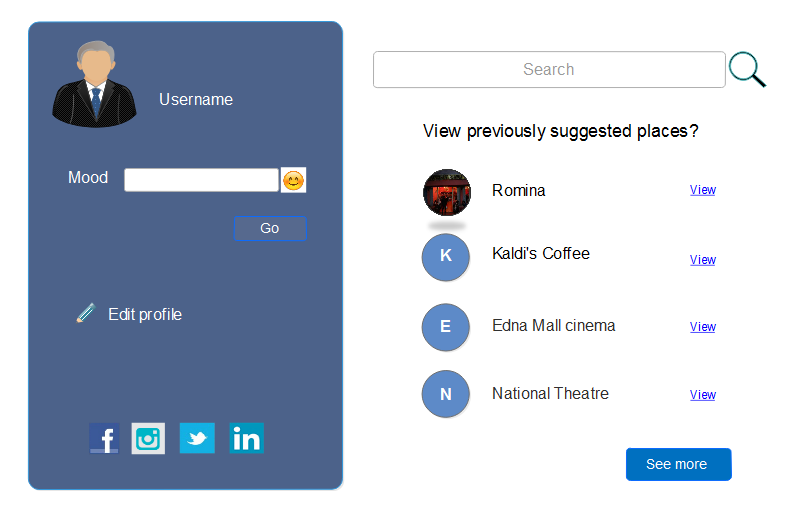
*Figure 3.1.1.2.2 signup (Individual) screen*

**3.1.1.2.3 Signup screen for Places.**

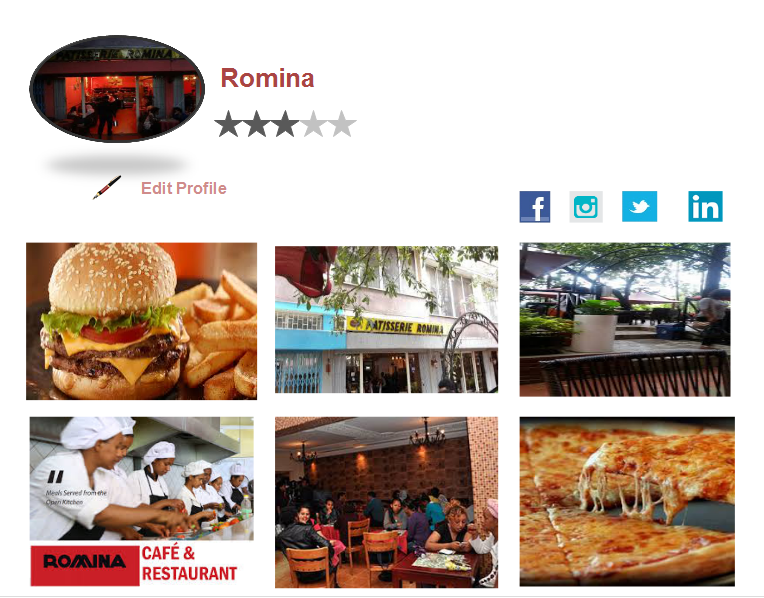


*Figure 3.1.1.2.3 signup (Place) screen*

**3.1.1.2.4 Home screen for Individuals.**

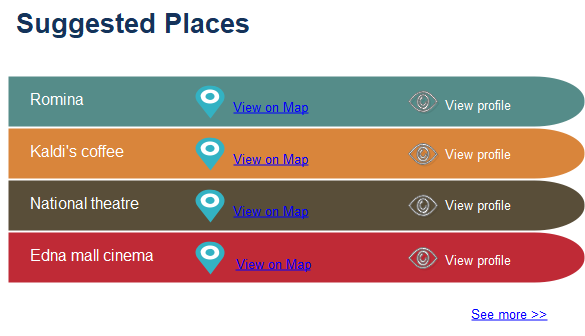
*Figure 3.1.1.2.4 Home screen (Individual)*

**3.1.1.2.5 Home screen for Places.**



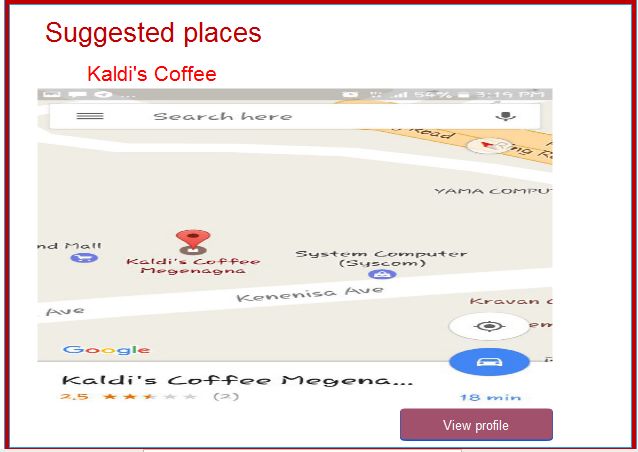
*Figure 3.1.1.2.5 Home screen (Places)*

**3.1.1.2.6. Suggested Place**



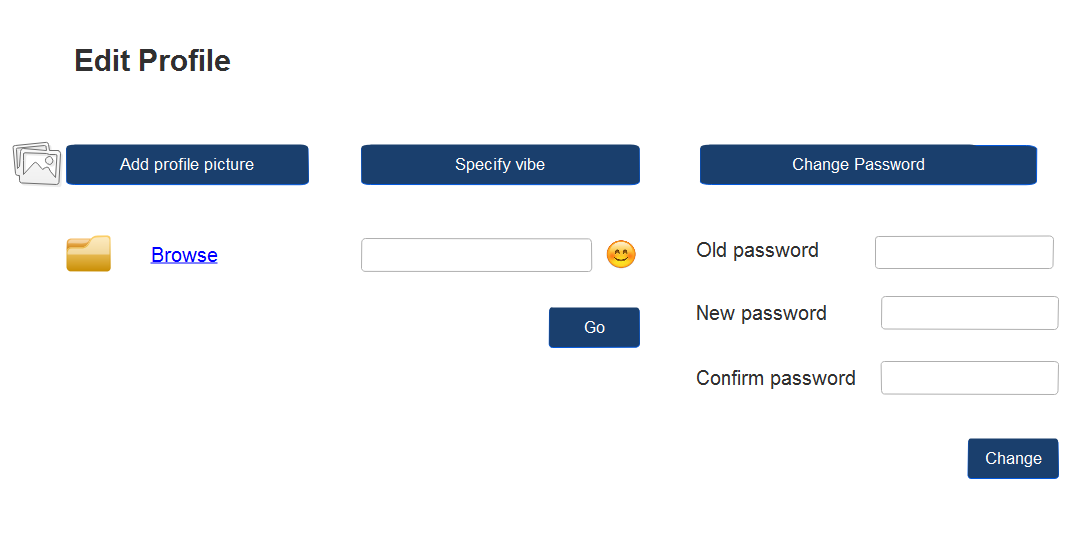
*Figure 3.1.1.2.6 Suggested Places*

**3.1.1.2.7 View on map**



*Figure 3.1.1.2.7 view on map*

**3.1.1.2.8 Customize Profile**



*Figure 3.1.1.2.7 Customize Profile*

**3.1.1.2.9 View Profile**



*Figure 3.1.1.2.8 view Profile*

### 3.1.2 Hardware Interfaces

The system has no specific hardware interface.

### 3.1.3 Software Interfaces

This system has no specific software interface.

### 3.1.4 Communication Interfaces

The system uses the Internet as a means of communication between the client app and the server, thus the system will be relying on the TCP/IP protocol.

## 3.2 Functional Requirements

### 3.2.1 FR-01 Registration

**3.2.1.1 Introduction**

This functional requirement states: the application should allow users to register either as a *place* or *individual,* with the appropriate screen*.*

**3.2.1.2 Input**

Inputs required are: Username, Password and email address from both types of Users, and Phone number, current location and vibe for *Places*.

**3.2.1.3** **Processing**

The user inputs mentioned above are sent to the server using the Internet where they are written to a database record.

**3.2.1.4 Output**

Upon successful registration, the user information will be written to database and notify the user or prompt the user to their home screen.

**3.2.1.5 Error Handling**

The application will notify the user of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 

### 3.2.2 FR-02 Login

**3.2.2.1 Introduction**

The application shall provide Users with a means of authorization before logging in to their account.

**3.2.2.2 Input**

The application will accept Username and password as a form of input.

**3.2.2.3 Processing**

The application will send the accepted inputs and sends it over to the server where verification of the user credentials is assured.

**3.2.2.4 Output**

The application upon successful verification will prompt users to their home screen.

**3.2.2.5 Error Handling**

The application will notify the user of the problem that occurred. Actions he might take to resolve the problem are also suggested

### 3.2.3 FR-03 Customize Profile

**3.2.3.1 Introduction**

The application shall provide users with a profile which they can customize, based on the type of users they are.

**3.2.3.2 Input**

The application accepts a mood/vibe choice or a request to further modify the profile via add profile picture, change password, or add/remove pictures.

**3.2.3.3 Processing**

The application will send the accepted mood/vibe input and send it over to the server, or redirects user to the appropriate screen if further modifications is requested.

**3.2.3.4 Output**

The database will be updated with the new mood/vibe.

The profile screen of the user will be updated accordingly, or will be redirected to the appropriate screen if further modifications are requested.

**3.2.3.5 Error Handling**

The application will notify the user of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 

### 3.2.4 FR-04 Choose mood/vibe

**3.2.4.1 Introduction**

The application shall offer its users multiple types of moods or *place vibes* to choose from.

**3.2.4.2 Inputs**

Request to choose moods or *place vibes.*

**3.2.4.3 Processing**

Determine the type of User it is and prepare a set of moods or *place vibes.*

**3.2.4.4 Output**

A list of moods or place vibes to choose from.

***3.2.4.5 Error Handling***

The application will notify the user of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 3.2.5 FR-05 Suggestion

**3.2.5.1 Introduction**

The application shall suggest a place based on *Individual* mood and *Place* vibe.

**3.2.5.2 Inputs**

Current *Individual* mood and place vibe entered.

**3.2.5.3 Processing**

After the user requests a place to be suggested, the client app sends current mood of the individual and to the server.

The server then tries to find best fit by comparing this data against several place vibes.

**3.2.5.4 Output**

Notifies *individual* of the found places and presents them with a link to the profile of the *Places* suggested.

**3.2.5.5 Error Handling**

The system will notify the user of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 

### 3.2.6 FR-06 Rate

**3.2.6.1 Introduction**

*Individuals* shall be able to give feedback on a suggested *place* by rating them out five*.*

**3.2.6.2 Input**

The input is the amount of score the user gave the place on some defined scale.

**3.2.6.3 Processing**

After an individual has rated a *place,* the data is sent to a server, where the rating is used to calculate the overall rating of the place.

**3.2.6.4 Output**

The database and profile of the *place* will be updated following Rating.

**3.2.6.5 Error handling**

The application will notify the user of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 3.2.7 FR-07 Set Profile picture

**3.2.7.1 Introduction**

The application shall allow the User to specify a profile photo.

**3.2.7.2 Inputs**

The selected photo.

**3.2.7.3 Processing**

Sending of the photo to server and saving the photo in a database.

**3.2.7.4 Output**

Profile photo of the user will be changed.

**3.2.7.5 Error Handling**

The application will notify the user of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 3.2.8 FR-08 Reset password

**3.2.8.1 Introduction**

The application shall allow the User to reset his/her password.

**3.2.8.2 Inputs**

User’s confirmation he still has access to the email address provided. The other is the new password desired.

**3.2.8.3 Processing**

Instructions to reset password are sent to the email address

**3.2.8.4 Output**

Resets the password with the new password provided.

**3.2.8.5 Error Handling**

The application will notify the personnel of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 3.2.9 FR-09 Add/remove photo

**3.2.9.1 Introduction**

The application shall allow the Place to add or remove a photo to/from its gallery.

**3.2.9.2 Inputs**

The photo selected to be removed or added.

**3.2.9.3 Processing**

Send the photo over to server or instruct the server to remove the selected photo based on the user’s request.

**3.2.9.4 Output**

The selected photo is removed or added as instructed.

**3.2.9.5 Error Handling**

The application will notify the personnel of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 

### 3.2.10 FR-10 Change password

**3.2.10.1 Introduction**

The application shall allow the User to change his current password.

**3.2.10.2 Inputs**

The current password and the new password.

**3.2.10.3 Processing**

The current password supplied is compared with the existing password on the server if it matches the current password is replaced with the new password.

**3.2.10.4 Output**

Change the user’s current password and writes changes to the database.

**3.2.10.5 Error Handling**

The application will notify the personnel of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 3.2.11 FR-11 View Place Profile

**3.2.11.1 Introduction**

The application shall allow the User to view the profile of the place he/she got as a suggestion.

**3.2.11.2 Inputs**

The user name of the place he/she want to see the profile of.

**3.2.11.3 Processing**

The user name of the place is sent to server where its profile information is retrieved from the database.

**3.2.11.4 Output**

Uses the information provided from the server to display the desired place profile.

**3.2.11.5 Error Handling**

The application will notify the personnel of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 3.2.12FR-12 View on map

**3.2.12.1 Introduction**

The application shall allow the User to view the location of the suggested place on a map.

**3.2.12.2 Inputs**

The user name of the place we need to retrieve the location of.

**3.2.12.3 Processing**

The location of the place is retrieved by sending the provided username and used to identify the location on the map

**3.2.12.4 Output**

The place location will be displayed on a map.

**3.2.12.5 Error Handling**

The application will notify the personnel of the problem that occurred. Actions he might take to resolve the problem are also suggested.

### 3.2.13FR-13 Logout

**3.2.13.1 Introduction**

The application shall allow the User to log out of their respected accounts.

**3.2.13.2 Inputs**

None.

**3.2.13.3 Processing**

The application will clear the user information and the token stored on the local storage.

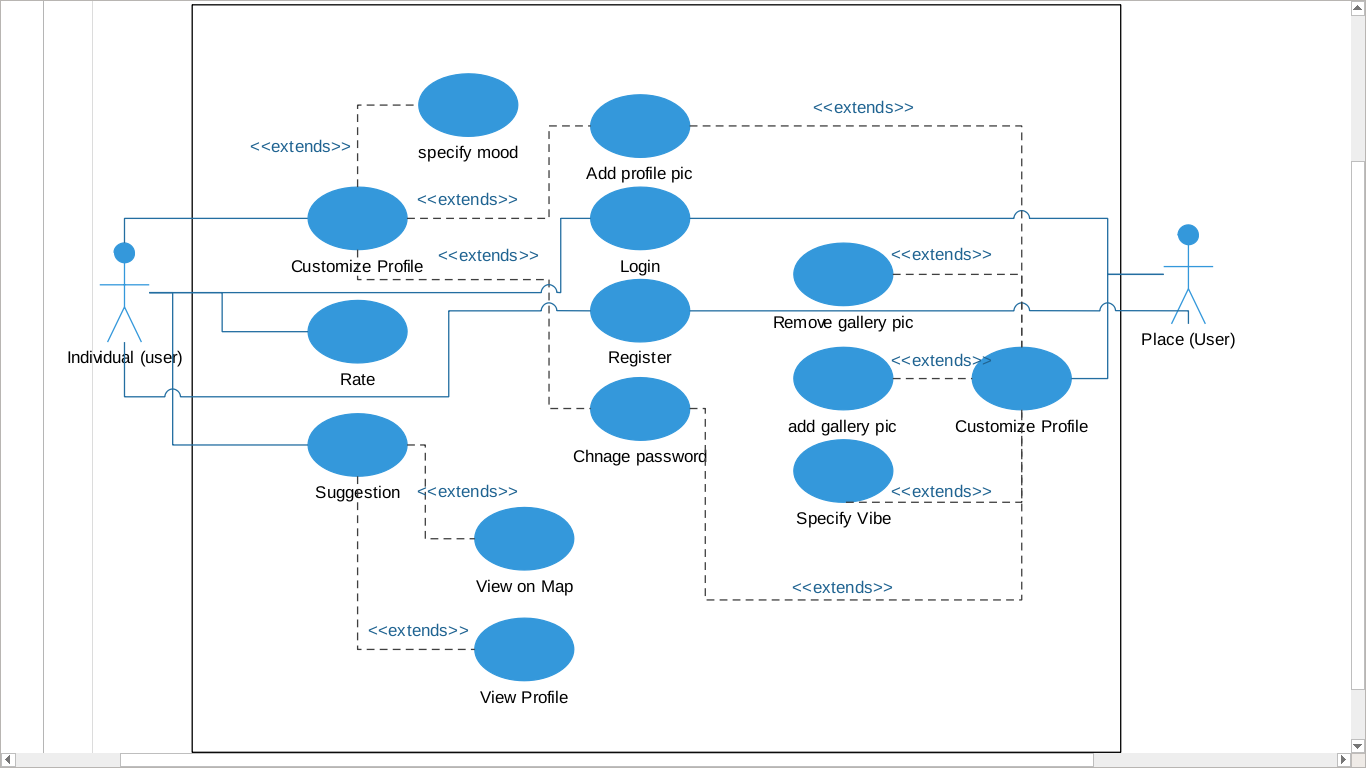
**3.2.13.4 Output**

The user should be logged out successfully and is directed to the login page.

**3.2.13.5 Error Handling**

The application will notify the personnel of the problem that occurred. Actions he might take to resolve the problem are also suggested.

3.3 Uses Cases



*Figure 3.3.1 Use case diagram*

### 3.3.1 UC-01 Customize profile

**Primary actors**: User (*Individual or Place)*

**Goal**: For user to customize their how their profile is.

**Precondition**: User must be logged in.

**Success end**: user customizes his/her profile.

**Failure end**: user can’t customize his/her profile.

**Trigger**: user requests (Through the user interface) to edit their profile.

**Main success scenario:**

1. User inserts username and password at login screen.
2. System authenticates user and logs the user in.
3. System redirects user to home screen.
4. User requests to edit his/her profile.
5. System redirects to a screen where user can edit their profile.
6. User performs any of the following scenarios Add profile picture, Specify mood/vibe, Change password or remove/add picture.
7. System saves the changes made and updates their profile.

**Extensions:**

2a. Authentication fails

2a.1 System displays specific error message to user

### 3.3.2 UC-02 Add profile picture

**Primary actor**: both Individual and place.

**Goal**: to specify a picture to act as the profile photo.

**Precondition**: Users must be logged in, and at the customize profile screen.

**Success end**: Users add a new profile photo, and system displays it.

**Failure end**: No change will be made in the users profile photo.

**Trigger:** User requests to upload a profile photo, through the user interface.

**Main Success Scenario:**

1. User requests to upload a new profile photo.

2. System responds with a mechanism to choose a photo.

3. User selects the photo it wants.

4. System uploads the photo and notifies user of successful upload.

**Extensions:**

3a. User does not select a photo instead cancels.

3a. 1 system notifies user exactly one photo must be chosen.

4a. System fails to upload photo.

4a. 1 System notifies user of failure.

4a. 2 system redirects user to the customize profile screen.

### 3.3.3 UC-03 Specify mood/vibe

**Primary actor**: both Individual and place.

**Goal:** to specify a mood or a vibe to describe the individual or the place.

**Precondition:** Users must be logged in, and at the customize profile screen.

**Success end**: Users add a new mood/vibe and the system displays it.

**Failure end**: No change will be made in the users’ mood/vibe status.

**Trigger**: User requests to specify a new mood/vibe, through the user interface.

**Main Success Scenario:**

1. User requests to specify a new mood/vibe.

2. System lists the option available.

3. User selects the mood/vibe which it thinks fits it.

4. System saves the current mood/vibe status and redirects user to home screen.

**Extensions:**

4a. System fails to save current mood/vibe status.

4a. 1 System notifies user of failure.

4a. 2 system redirects user to the customize profile screen.

### 3.3.4 UC-04 Request Suggestions

**Primary actors**: User (*Individual only*)

**Goal**: User wanted a place suggestion based on his current mood.

**Precondition:** User must be logged in.

**Success end**: System suggests a place to the user.

**Failure end**: No suggestions will be made.

**Trigger**: User requests for a place to be suggested.

**Main Success Scenario:**

1. User requests for a place to be suggested.
2. System reads the current mood and searches for a place.
3. System displays found places.
4. User chooses to view on map.
5. System redirects user to an interactive map

**Extensions:**

2a. User has not supplied his/her current mood.

2a.1 System notifies and redirects user to fill his current mood.

2a.2 User fills mood or returns to home screen

3a. No places found that meets the user’s mood.

3a.1 System notifies then redirects the user to home screen.

**Alternatives:**

4a. User chooses to view profile of the place

4a.1 System redirects user to the profile of the place.

4a.2 User chooses to view on a map or goes back to home screen

### 

### 3.3.5 UC-05 Rate

**Primary actors**: Users (*Individual*)

**Goal**: User wishes to rate a place.

**Precondition**: User must be logged in, the place must have been suggested previously.

**Success end**: Users rating will be visible on the places profile.

**Failure end**: No change will be made to the profile of the place.

**Trigger**: user chooses to rate a previously suggested place.

**Main success Scenario:**

1. User performs Request suggestions scenario.
2. User visits place and tries to rate.
3. System displays the users rating.

### 3.3.6 UC-06 remove/add picture

**Primary actor:** Places.

**Goal**: to add or remove a picture from the gallery of the place

**Precondition: User** must be logged in, and at the customize profile screen.

**Success end**: Users add or remove a picture to their gallery and the system displays it.

**Failure end**: No change will be made in the users’ gallery.

**Trigger**: User requests to add or remove picture, through the user interface.

**Main Success Scenario:**

1. User requests to add picture

2. System responds with a mechanism to select photo to upload

(Add to the gallery).

3. User selects the photo it wants.

4. System saves the new photo in the gallery.

**Extensions:**

4a. System fails to save the new photo.

4a. 1 System notifies user of failure.

4a. 2 system redirects user to the customize profile screen.

**Alternatives:**

1a. User requests to remove a picture.

1a.1 System responds with a mechanism to select photo to remove the photo from the gallery.

1a.2 User selects the photo it wants.

1a.3 System removes the photo from the gallery.

### 3.3.7 UC-07 Sign up

**Primary actor**: Users

**Goal**: to become a registered user of the system.

**Precondition**: No preconditions.

**Success end**: User is registered in the system

**Failure end**: User is not registered

**Trigger**: User requests to be registered

**Main Success scenario:**

1. User requests to sign up as an individual.
2. System redirects user to the appropriate sign up screen
3. User provides necessary information and proceeds
4. System updates database and prompts user to home screen

**Extensions:**

3a. User doesn’t provide sufficient information

3a.1 System notifies user to provide sufficient information

**Alternatives:**

1a. User requests to register a place

1a.1 Continue from step 2

### 3.3.8 UC-08 Sign in

**Primary actor**: Users

**Goal**: To login in to the application

**Precondition**: no precondition.

**Success end**: User logs in to the application

**Failure end**: User doesn’t log in to the application

**Trigger**: User requests to log in.

**Main success scenario:**

1. User inserts his/her credentials and requests to sign in
2. System Authenticates User and logs him in.
3. System presents user with their home screen

**Extensions:**

1a. Users requests to sign in without providing credentials

1a.1 the application notifies user

2a. Authentication fails

2a.1 the application notifies user

### 3.3.9 UC-09 Change Password

**Primary actors**: Users

**Goal**: To change current password

**Precondition**: User is signed in

**Success end**: The password is changed

**Failure end**: Password is not changed

**Trigger**: User requests to change password

**Main Success Scenario:**

1. User Requests to change password.
2. System prompts user to a screen where he can change his/her password.
3. User fills the necessary information and submits the form.
4. System Changes password and updates database.

**Extensions:**

3a. User submits without filling necessary information

3a.1 System notifies user

3b. the entered current password doesn’t match the current Password.

3b.1 System notifies User

### 3.3.10 UC-10 Reset Password

**Primary actor**: User

**Goal**: To reset a forgotten password

**Precondition**: User has access to the app

**Success end**: The password is reset to a new password

**Failure end**: Password is not changed

**Trigger**: User requests to reset password

**Main Success Scenario:**

1. User requests to reset password
2. System redirects user to a screen where he can reset password.
3. System displays option(s) to recover/reset his password.
4. User Chooses an option and proceeds
5. Based on User’s Choice, System notifies user of instructions to follow
6. User Follows the steps and resets password.

**Extensions:**

3a. User didn’t choose an option

3a. System redirects to step2

### 3.3.11 UC-11 Logout

**Primary actors**: Users

**Goal**: To logout from user account

**Precondition**: User is logged in.

**Success end**: The user logged out successfully.

**Failure end**: User still logged in.

**Trigger**: User requests to log out.

**Main Success Scenario:**

1. User Requests to logout
2. The application clears the data stored on the local storage
3. The application prompts to login screen.

**Extensions:**

3a. the system fails to logout correctly

3a.1 User still logged in.

3a.2 System redirects user to the home page.

## 3. 4 NON-FUNCTIONAL requirements

### 3.4.1 Performance

* **Description** - View loading time

0.1 second is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result.

1.0 second is about the limit for the user’s flow of thought to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 second, but the user does lose the feeling of operating directly on the data.

**Measurements** - The response times will be measured using tools located behind the firewall and in front of the web servers. The timer will measure the time from the request for a page to when the last bit required rendering the page is returned. Backend response times will be measured using the application server log files.

* Database response time should not be greater than 0.5 seconds per query.
* The server should handle multiple client requests with a response time of not more than 1 sec per client.

### 3.4.2 Reliability

* The system should keep its data integrity by reflecting any change made, to data the database.
* The system should backup data in case of failure.
* Any type of failure on the client’s app should not cause loss or damage of client’s data.

### 3.4.3 Availability

* The system shall not be down for more than 30 minutes a day.
* The system must be available all the time, but since that’s not humanly possible we plan on getting to 99% or less of availability.

### 3.4.3 Security

* In signing up the system will check whether there is an existing account before it and will prompt failure.
* The system will check the user’s identity in logging them
* The system shall encrypt passwords when storing them.

### 3.4.5 Maintainability

* The application should be flexible and be capable of any adjustment (to add new features in the future) that would be applied to it.
* The application should also be open to testing its several functions.

### 3.4.6 Portability

* It shouldn’t take more than 2 days for a skilled software engineer to port the system to other platforms (Environments).

## 3.5 Inverse Requirements

The system will not offer any type of private chatting to individuals or places.

## 3.6 Design constraints

As of this writing we are not aware of any design constraints.

3.7 Other Requirements

### Training-related Requirements

There will not be any training provided.

# 4. Change Management Process

Any requests to change the project scope and requirements shall be discussed by all the members of the team. A change will be made only when the majority of the team agrees on the change. In this case, the SRS document shall be updated by the team members in order to reflect the changes, and a date of change shall be noted in the file. If this change request is made by the client or anyone outside of the team, he or she will have to contact the team. If a change request is made by a team member, he or she can raise it during the weekly team meeting or contact other team members via email. During any of these requests, the team will assess the feasibility of the proposed changes considering the time constraints and structural constraints of the implemented modules and develop an implementation strategy. A change plan will be created for the implementation of the change. The team will then continue implementing the new requirements.

# 

# References

* Wikipedia – feasibility engineering
* Ian Somerville – software engineering 9th edition
* The section named “Change management process” taken from samples, specifically “AYATE ETHIOPIAN HOME REMEDIES”.

****

**ADDIS ABABA UNIVERISTY**

**ADDIS ABABA INSTITUTE OF TECHNOLOGY**

**Center of Information Technology and Scientific Computing**

*Department of Software Engineering*

**Software engineering II**

**Software Design Specification**

**Prepared By:**

1. **Aman Bereket ATR/9348/08**
2. **Biya Girma ATR/7547/08**
3. **Estifanos sisay NSR/9401/08**
4. **Hena Fufa ATR/3750/08**
5. **Hermella Frew ATR/1689/08**
6. **Mihret Tamene ATR/3534/08**
7. **Oromia Godanna ATR/6053/08**
8. **Yohannes Fassil ATR/4122/08**

**Submitted to: Mr.** Natnael Argaw

Date: March 23/2018

Addis Ababa

Ethiopia

**Software Design Specification**

**(SDS)**

Table of Contents

[Table of Contents 3](#_Toc509606781)

[Revision History 4](#_Toc509606782)

[Document Approval 4](#_Toc509606783)

[List of tables 5](#_Toc509606784)

[List of figures 5](#_Toc509606785)

[Definitions, Acronyms, Abbreviations 6](#_Toc509606786)

[1.Introduction 7](#_Toc509606787)

[1.1 PURPOSE 7](#_Toc509606788)

[1.2 GENERAL OVERVIEW 7](#_Toc509606789)

[1.3 Development Method 8](#_Toc509606790)

[2.System Architecture 9](#_Toc509606791)

[2.1. Subsystem decomposition 9](#_Toc509606792)

[2.2. Hardware/software mapping 12](#_Toc509606793)

[3.Object Model 14](#_Toc509606794)

[3.1. Class Diagram 14](#_Toc509606795)

[3.2 Sequence Diagram 15](#_Toc509606796)

[3.1. State Chart Diagram 24](#_Toc509606797)

[**4.Detailed Design 24**](#_Toc509606798)

[References 41](#_Toc509606799)

[Books 41](#_Toc509606800)

[Web resources 41](#_Toc509606801)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| March 23 | Version 2 | Team members | First Draft |
| April 12 | Version 3 | Team members | Final revision |
| April 30 | Version 4 | Team members | Modification of SDS |

# Document Approval

The following SDS Documentation has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Natnael Argaw | Instructor, ITSE |  |

# List of tables

Detailed design of all classes--------------------------------------------------------- page 24 – page 40

# List of figures

*Figure 2.1.1* **Layer 1 Component decomposition diagram** ------------------------------------------------ 9

*Figure 2.1.2* **Layer 2 Component decomposition diagram** ---------------------------------------------- 10

*Figure 2.1.3* **Layer 3 Component decomposition diagram** ---------------------------------------------- 11

*Figure 2.2* **UML deployment diagram** ------------------------------------------------------------------------ 12

*Figure 3.1* **Class diagram**-------------------------------------------------------------------------------------------13

*Figure 3.1.1* **Customize profile**-----------------------------------------------------------------------------------14

*Figure 3.1.2* **add profile Picture**---------------------------------------------------------------------------------15

*Figure 3.1.3* **specify mood**-----------------------------------------------------------------------------------------16

*Figure 3.1.4* **request suggestion**---------------------------------------------------------------------------------17

*Figure 3.1.5* **rate**-----------------------------------------------------------------------------------------------------18

*Figure 3.1.6* **remove/add photo**---------------------------------------------------------------------------------19

*Figure 3.1.7* **signup**-------------------------------------------------------------------------------------------------20

*Figure 3.1.8* **Login**---------------------------------------------------------------------------------------------------21

*Figure 3.1.9* **change password------------------------------------------------------------------------------------**22

*Figure* 3.1.10 **Logout**------------------------------------------------------------------------------------------------23

# Definitions, Acronyms, Abbreviations

**Mood:** A state or quality of feeling at a particular time.

**SDS** – Software Design Specification

**SQL** – Structured Query Language

**HTTP** – Hyper Text Transfer Protocol

**CSS** – Cascading Style Sheet

**UI** – User Interface

**DBMS -** Database Management System

**TCP/IP -** Transmission Control Protocol/Internet Protocol

**SRS**– System Requirement Specification

**API -** Application Programming Language

**DB –** Database

**JS -** JavaScript

# Introduction

## 1.1 PURPOSE

This SDS documentation is a detailed description of the system's design method and processes. It will be used as a base for the implementation phase of development. It will also give an overview of the system architecture mentioned in the requirement documentation.

## 1.2 GENERAL OVERVIEW

The software being designed is named “vibe”, it is a web application which gives place suggestions by taking the mood of the user as an input. The list is pulled from a database consisting of registered places with their services, location and vibe. The system will help users liberate themselves from their mood, emotion or attitude by pairing them with places that match with the vibe they needed at a convenient place and time.

By taking into consideration that the users could be in any age group we aim to make the product easy to use &understandable. This application will be developed using the nodesjs platform and will run on our own server. It also contains a Mongo DB database for the registered users and places. It will also use the express *package*, which is a package that provides HTTP capabilities*.*

## 1.3 Development Method

We will be using the staged delivery method because of the time constraint to modify the requirements continuously and our understanding of the system requirement. The implementation will take a Model-View-Controller system architecture separate internal representation from the ways that information is presented to and accepted from the user. The MVC design patter decouples these major components allowing for efficient code reuse and parallel development. **Model** represents the data which resides in some database or file system. **View** represents the visualization of the data model and could be presented in many different ways. The **Controller** controls the data flow into model object and updates the view whenever data changes. The detailed architecture of the system will be discussed more in later chapters of this document.

We will be using TCP/IP interface for communication between the client & Server app.

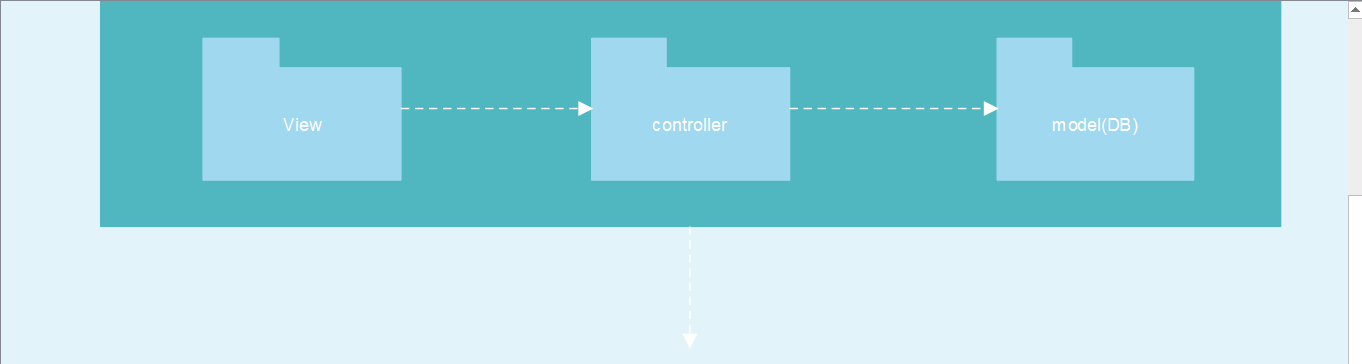
Furthermore the system shall be constrained by the general constrains mentioned by the REST principle. Thus the system shall be a RESTFul web API.

The UI will be developed using HTML, CSS and JavaScript to manipulate the DOM (no templating engine on the server side). We chose this approach because the application logic can be separated from the GUI code which is a better approach for code organization and helps clarify code during maintenance and, furthermore it help in the scalability of the application.

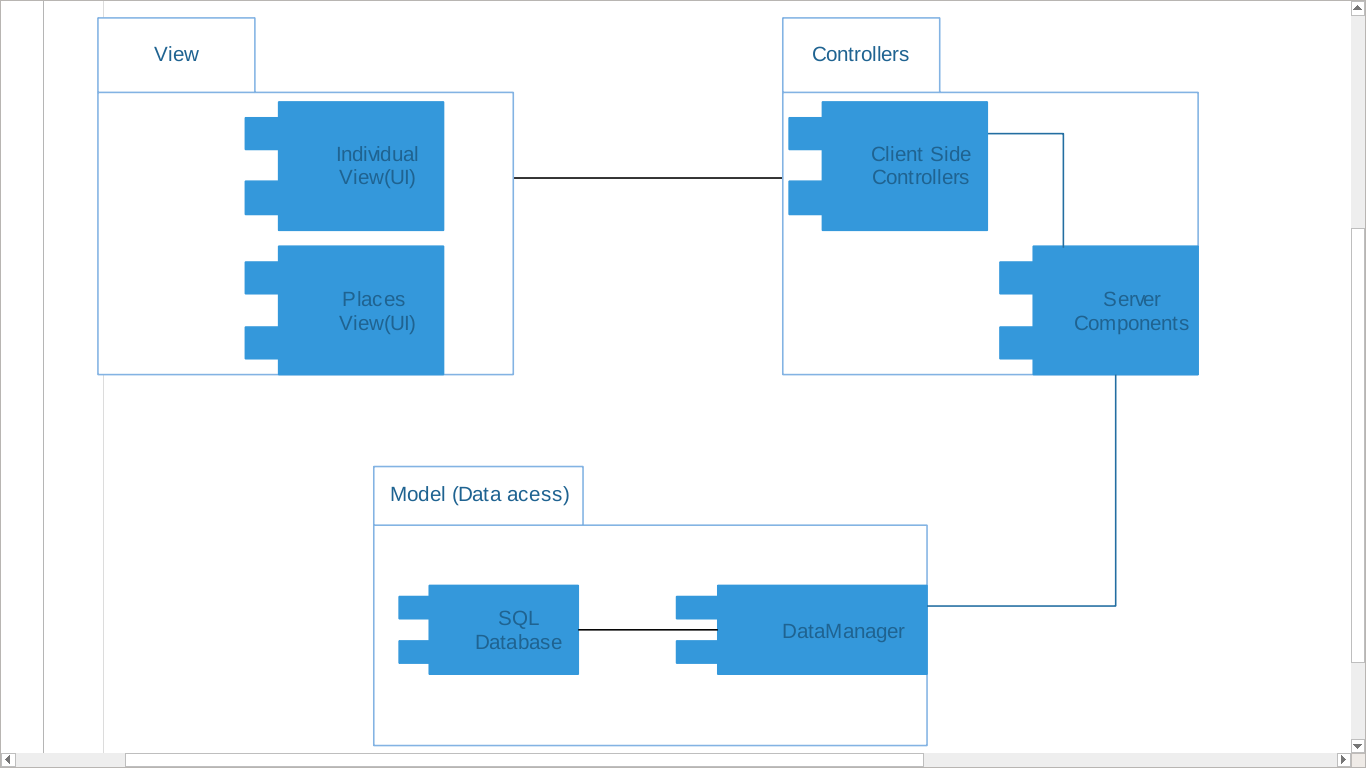
The database will be designed using Mongo DB, a document oriented database program which uses JSON like documents with schemas. We chose this database because it provides high availability with replication sets, can scales horizontal .Mongoose is used to communicate with a database, define schemas and models and make basic queries to perform tasks such as update data on a database or retrieve data from a database. The database will entail two tables. One for registered places and one for signed up users. Each table will have a user name, password which will be encrypted for security purposes. And the place table will extend by adding services and contact information. DBMS will be used for storing, organizing, retrieving and modifying data.

# System Architecture

## Subsystem decomposition

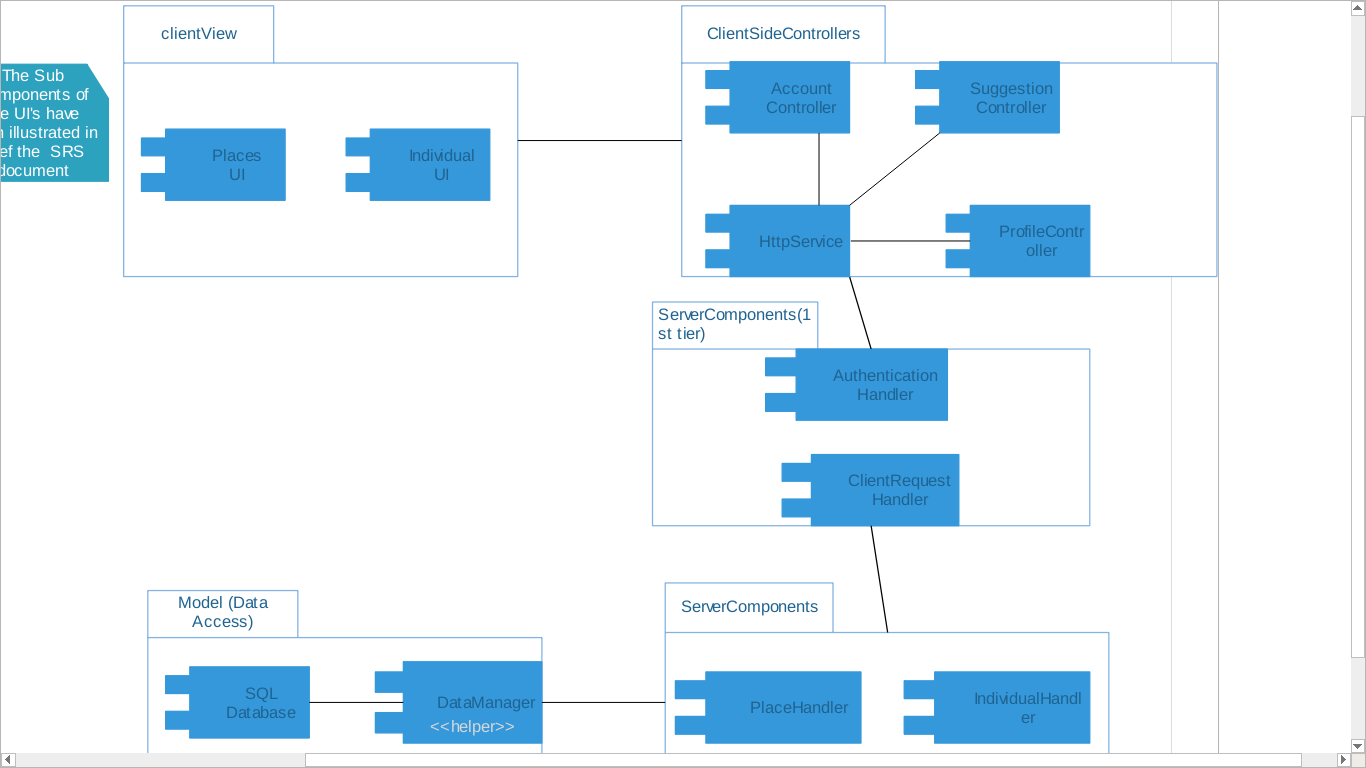


*Figure 2.1.1 Layer 1 component decomposition*



Mongo DB

*Figure 2.1.2 Layer 2 component decomposition diagram*



Mongo DB

Mongo DB

Mongo DB

Database

BB

Mongo DB

*Figure 2.1.3 Layer 3 component decomposition*

## Hardware/software mapping

Mongo server

Node

**User pc**

HTTP Mongoose

**Web Browser**

**Data base**

**Server**

*Fig 2.2 UML Deployment diagram*

# *3.* Object Model

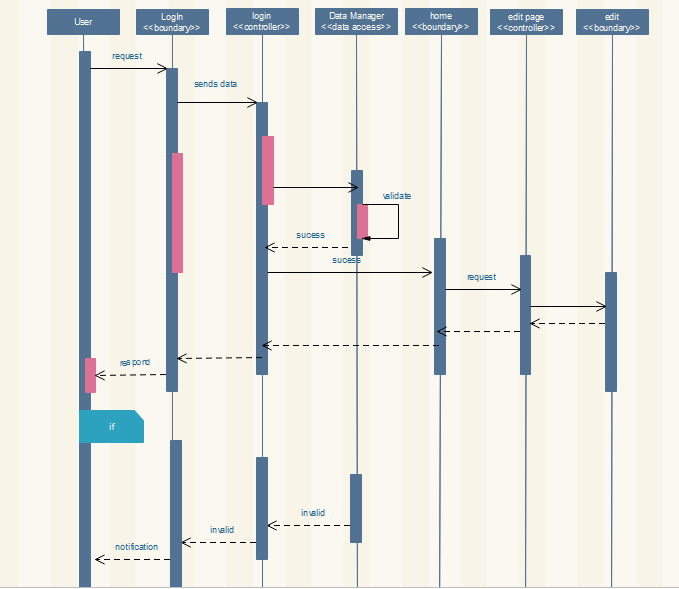
## Class Diagram

## 

*Figure 3.1 Class Diagram*

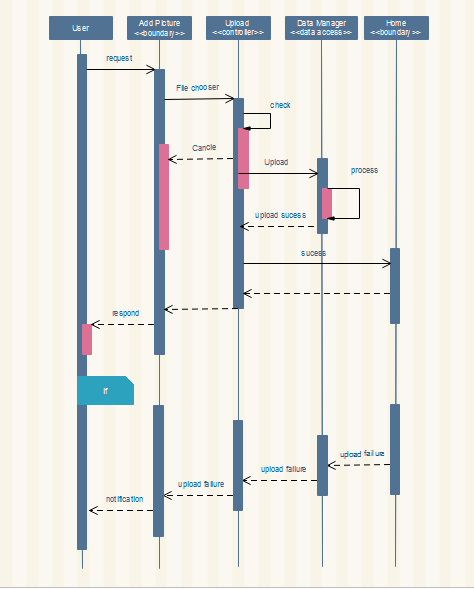
## 3.2 Sequence Diagram

* + 1. Customize profile



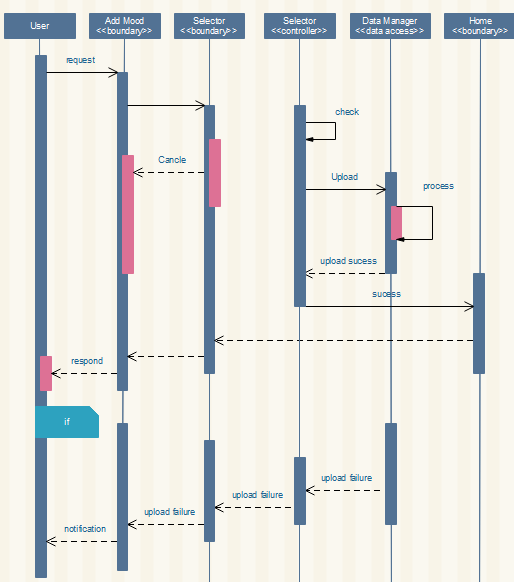
*Figure 3.1.1 Customize profile*

* + 1. Add profile picture



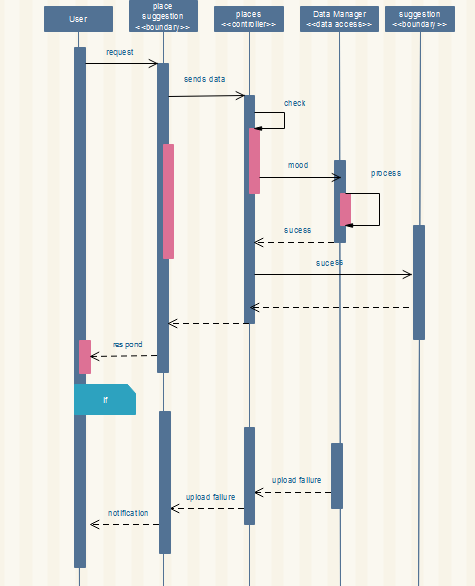
*Figure 3.1.2 add Profile Picture*

* + 1. Specify mood/vibe



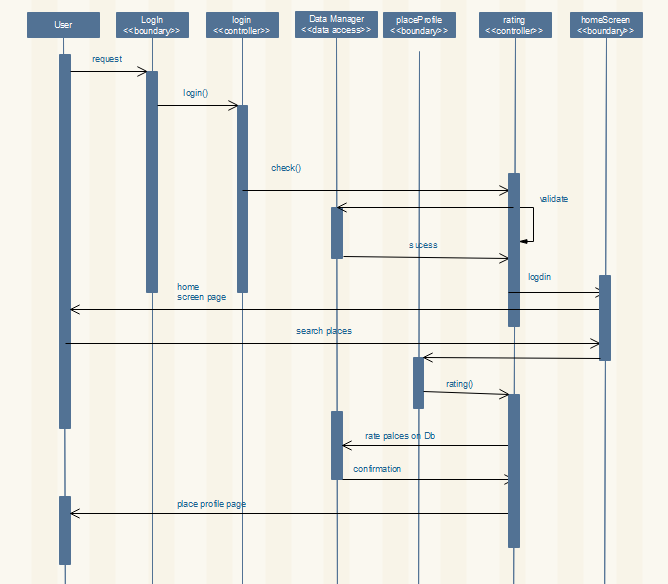
*Figure 3.1.3 specify mood*

* + 1. Request suggestion



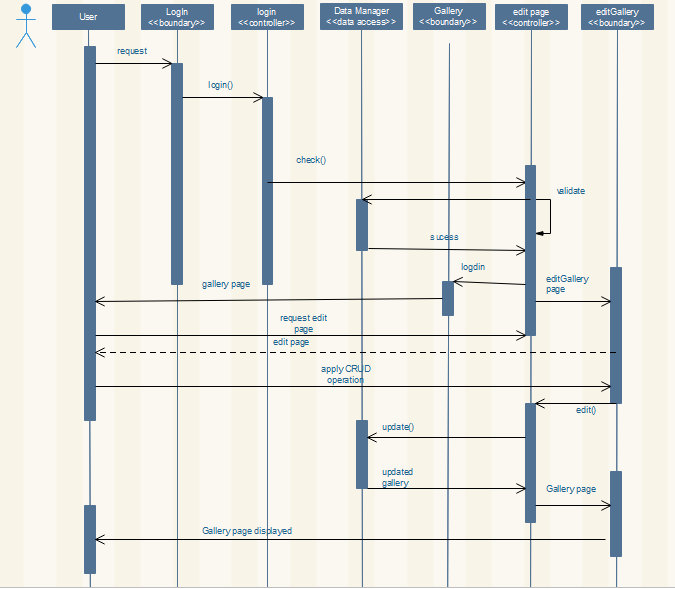
*Figure 3.1.4 request suggestion*

* + 1. Rate



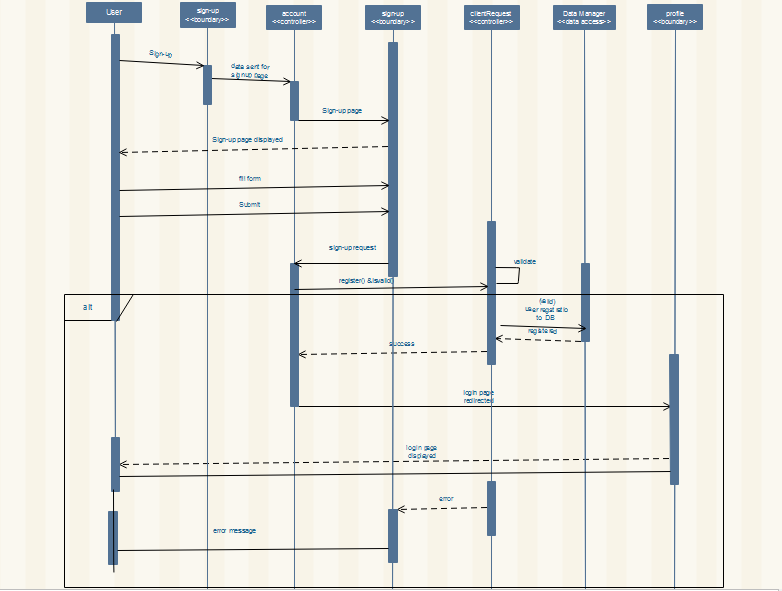
*Figure 3.1.5 Rate*

* + 1. Remove/add picture



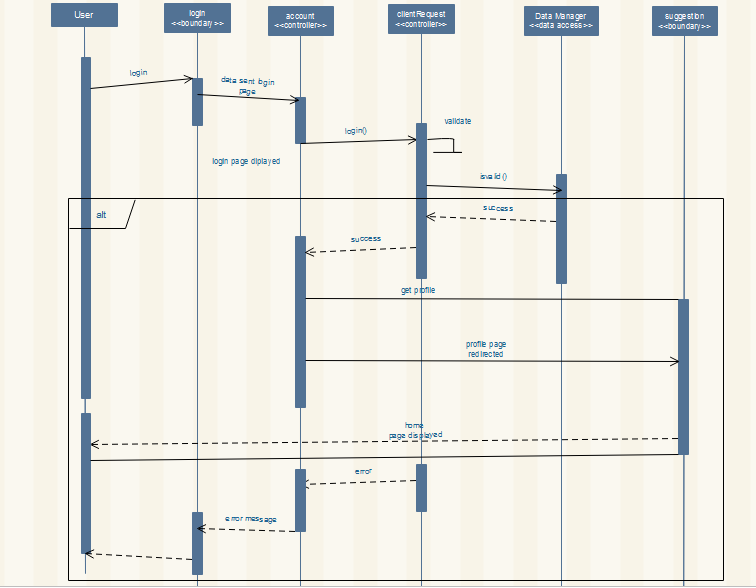
*Figure 3.1.6 Remove/add picture*

* + 1. Sign up



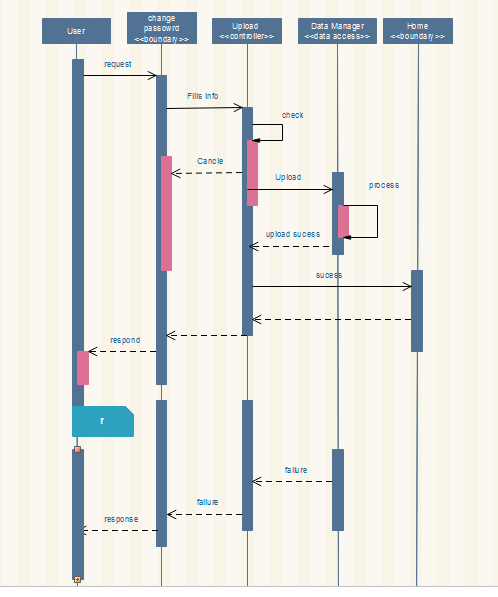
*Figure 3.1.7 sign up*

* + 1. Sign in



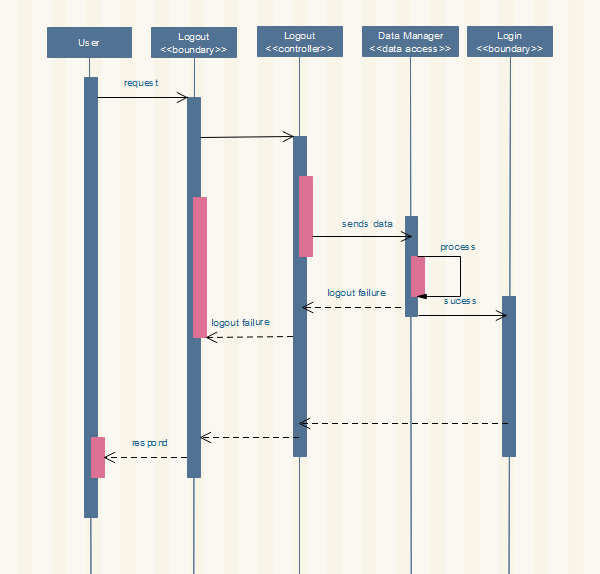
*Figure 3.1.8 sign in*

* + 1. Change Password



*Figure 3.1.9 change password*

* + 1. Logout



*Figure 3.1.10 Logout*

## 3.1. State Chart Diagram

None

## Detailed Design

This section briefly discusses the class along with their respective attributes and methods.

|  |
| --- |
| **Users** |
| #userName: String  #userId:int  #Email: String  #Phone number: integer  #profilePic: String |
| +getUserName(): String  +getName(): String  +changePassword(String newPassword): void  +changeProfilePic(String newProfilepic): void |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| Email | String | Protected | Email <> NULL   * Must contain @ * Must contain .(dot) * Position @>1 * Position of (dot)> position of @ + 2 * Position of (dot)+3 <= total length of email address and the total character of the Email is at least 5 characters |
| Phone number | String | Protected | Phone <> NULL must not be less than10 digits and must start by +251/09 |
| username | String | Protected | UserName <> NULL contain special characters and integers. |
| UserId | Integer | Protected | Not Null, System generated unique identifier. |
| Password | String | Protected | Password <>NULL, it must be greater than 4 digits and it can contain special characters, integers and characters. |
| profilePic | String | Protected | profilePic <>NULL |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | visibility | Return type | Argument | Pre-condition | Post-condition |
| getUserName | public | String | - | The user’s user name should exist in database. | The submitted user name should be retrieved. |
| changePassword | Public | Void | newPassword: String | User’s previous password should exist in the database and the user has to be logged in . | User’s password should be changed. |
| changeProfilePic | public | Void | newProfilepic: String | The profile pic should be in the database and the user has to be logged in. | User’s profile pic should be changed. |

|  |
| --- |
| **Individual** |
| -firstName: String  -lastName: String  -User Id: Integer  -Mood: String  - History: String[]  ^#userName: String  ^#Email: String  ^#Phone number: integer  ^#profilePic: String |
| +specifyMood(newMood): void  ^+changePassword(String newPassword): void  ^+changeProfilePic(String newProfilepic): void |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| firstName | String | private | firstName<> NULL should not contain special characters. |
| lastName | String | private | lastName<> NULL should not contain special characters. |
| userId | Integer | private | Auto-increment from the database |
| Mood | String | private | Mood <> NULL |
| History | String[] | private | History<>NULL |
| UserName | String | protected | UserName <> NULL contain special characters and integers. |
|  |  |  |  |
| Email | String | protected | Email <> NULL   * Must contain @ * Must contain .(dot) * Position @>1 * Position of (dot)> position of @ + 2 * Position of (dot)+3 <= total length of email address and the total character of the Email is at least 5 characters |
| Phone number | Integer | protected | Phone <> NULL must not be less than10 digits and must start by +251/09 |
| ProfilePic | String | protected | profilePic <>NULL |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return Type | Argument | Pre-Condition | Post-Condition |
| changeMood | public | void | newMood: String | The user’s mood should exist in the database and the user has to be logged in. | The previous mood should be replaced (changed) by the submitted mood. |
| getProfilepic | public | String | - | The user’s profile picture should exist in the database. | User’s profile picture should be returned. |
| changePassword | Public | void | newPassword: String | User’s password should exist in the database | User’s password should be changed. |
|  |  |  |  |  |  |
| changeProfilePic | public | void | newProfilepic: String | The profile picture should be in the database. | The previous profile pic should be replaced by the new profile picture. |

|  |
| --- |
| **Place** |
| -placeName: String  -placeId: Integer  -rating: Integer  - placeVibe: String  -PlaceLocation: String  ^#userName: String  ^#Email: String  ^#Phone number: integer  ^#profilePic: String |
| +insertPic(newPic): void  +removePic(selectedPic): void  +specifyVibe (newPlaceVibe): void  ^+changePassword(String newPassword): void  ^+changeProfilePic(String newProfilepic): void |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| placeName | String | Private | placeName<>Null should not contain special characters and integer |
| placeId | integer | Private | Auto-increment from the database |
| Rating | Integer | protected | Rating<>Null |
| PlaceVibe | String | protected | placeVibe<>Null can contain special characters |
| placeLocation | String | protected | placeLocation<>Null should not contain special characters . |
| userName | String | protected | UserName <> NULL contain special characters and integers. |
| Email | String | protected | Email <> NULL   * Must contain @ * Must contain .(dot) * Position @>1 * Position of (dot)> position of @ + 2 * Position of (dot)+3 <= total length of email address and the total character of the Email is at least 5 characters |
| Phone number | Integer | protected | Phone <> NULL must not be less than10 digits and must start by +251/09 |
| Password | String | protected | Password <>NULL, it must be greater than 4 digits and it can contain special characters, integers and characters. |
| ProfilePic | String | protected | profilePic <>NULL |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-condition | Post-condition |
| insertPic | Public | Void | String |  | Inserts the picture to the gallery |
| removePic | Public | Void | String | The image must exist in the database. | The picture should be deleted from the gallery |
| modifyPlaceVibe | Public | Void | newPlaceVibe: String | The place Vibe must exist in the database. | The vibe of the places must be modified |
| changePassword | Public | void | newPassword: String | User’s password should exist in the database and the user has to be logged in. | User’s password should be changed. |
| changeProfilePic | Public |  | newProfilepic: String | The profile picture should be in the database and the user has to be logged in. | User’s profile pic should be changed. |

|  |
| --- |
| HttpService |
| -SERVER\_URL: String |
| +changePassword(String type,String oldPassword, String newPassword):void  +specifyMood(String type, String mood):void  +changeProfilePIc(String type , File img):void  +viewProfile(int pid):void  +rate(int rating , int pid , int uid):void  +getHistory(int uid):void  +removePicture(int pid, String url):void  +addPicture(int pid, File img):void |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-condition | Post-condition |
| changePassword | Public | Void | String type  String oldPassword  String newPassword | User has supplied necessary old and new password. | The the arguments are sent to the server |
| SpecifyMood | Public | Void | String type -  String mood | Users mood is available | An acknowledgment from the server. |
| ChangeProfilePic | Public | Void | String type  File img | User has selected an image file. | An acknowledgment from the server. |
| viewProfile | public | Void | int pid | Atleast one place must have been suggested | The profile of the place choosen is displayed |
| rate | public | Void | int rating , int pid , int uid | Atleast one place must have been suggested | The rating is displayed and saved in a database. |
| getHistory | public | Void | int uid | None | The history of the individual is returned  . |
| removePicture | public | Void | int pid, String url | A picture must have been selected from the gallery | The picture is removed both from the gallery and from the server/databse. |
| addPicture | public | Void | int pid, File img | A picture must have been selected. | The picture is added to the gallery and saved on the server |

|  |
| --- |
| **Suggestedcontroller** |
| -suggestedPlaces:String[] |
| +ViewOnMap(int pid)  +ViewProfile(int pid)  +rate( |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| suggestedPlace | String[] | Private | String[] elements must be from one of the places registered in the system. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Precondition | Post-condition |
| ViewOnMap | Public | Void | place p | There should be a selected place to be viewed | The map of the selected place will be shown. |
| Viewprofile | Public | Void | place d | The user should have a profile. | The profile should be displayed. |
| rate | public | Void | None - | Place must have been sugested | Acknowledgment from the server.. |

|  |
| --- |
| **Login Controller** |
| -username: String  -password: String |
| +loginHandler()  +signUpHandler() |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-condition | Post-condition |
| loginHandler | public | void | none | none | Username and password sent to server |
| signupHandler | public | void | -none | none | User information sent to server. |

|  |
| --- |
| HomeController |
| user: User |
| +init() |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| user | User | Private |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return Type | Argument | Pre-condition | Post-condition |
| Init | public | Void | None - | User has logged in successfully | The home screen of the user is initialized |

|  |
| --- |
| **CustomizeProfile** |
| -currentUser: User |
| +changeProfilePic()  +removePic()  +addPic()  +changePassword()  +changeVibe() |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| currentUser | User | private | currentUser<> contains user’s name |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-condition | Post- condition |
| changeProfilepic | public | void | NewprofilePic:  String | The user’s previous profile picture should exist in the database. | User’s old profile picture should be replaced by the new profile picture. |
| removepic | public | void | - | The user’s picture should exist in the database. | The pictures should be removed from the database. |
| addPic | public | void | - | The picture shouldn’t exist. | The picture should be added in the database. |
| ChangePassword | public | Void | Noen | User has entered necessary info | New and old password sent to server. |
| Changevibe | public | Void |  | User has entered vibe/mood | Vibe changed |

|  |
| --- |
| **DataManager** |
| -DB\_USER:String  -Con: Connection  -DB\_PASSWORD:String  -DB:String |
| +getIndividual(int uid):Individual  +getPlace(int pid):Place  +addPicture(int pid, String path)  +changePassword(int id, String newPassword, String type):  +addPhoto(int pid, String path);  +removePhoto(int pid, String path);  +addRating(int pid, String path) |
|  |
|  |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| DB\_USER | String | Private | Must containe a valid database user |
| Con | Connection | Private | Not Null |
| DB\_PASSWORD | ResultSet | Private | Must containe a valid database users password |
| DB | User | Private | The database to be accessed must exist. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-condition | Post-condition |
| getPlace | public | Place | Int pid | The user should enter its user name. | Should return true if the user’s name exists |
| getIndividual | public | Individual | Int uid | The user should exist in the database | The user is retrieved. |
| addPidture | Public | Void | Int pid, String path | The database to be updated should exist. | The database should be update |
| RemovePhoto | public | Void | Int pid, String path |  |  |
| AddRating | Public | Void | Int pid, int uid, int rating |  |  |

|  |
| --- |
| **ClientRequestHandler** |
| App:Express  individualHandler:IndividualHandler  placeHandler:PlaceHandler  authenticationHandler:AuthenticationHandler |
| Listen() : void |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| App | Express | private | An express app |
| individualHandler | IndividualHandler | private | Not Null |
| placeHandler | PlaceHandler | private | Not Null |
| authenticationHandler | AuthenticationHandler | private | Not Null |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-condition | Post-condition |
| Listen | public | void | - |  |  |

|  |
| --- |
| **AuthenticationHandler** |
| -router:Express.Router  -dataManager:DataManager  - |
| +login(req,res,next)  +register(req,res,next)  +changePassword(req,res,next)  +newToken(User user):String  +decodeToken(String token):User |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| Router | Express.Router | private | router<> NULL |
| dataManager | DataManager | private | router<>NULL |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-condition | Post-condition |
| Login | public | Void | Express.Request req,  Express.Response  res, |  | Authentication result sent to client |
| Register | public | void | Express.Request req,  Express.Response  res, |  | Registration result sent to client |
| changePassword | public | void | Express.Request req,  Express.Response  res, |  |  |
| newToken | public | String token | User user |  |  |
| decodeToken | public | User | String token |  |  |

|  |
| --- |
| **IndividualHandler** |
| -router:Express.Router  -dataManager:DataManager  authenticationHandler:AuthenticationHandler |
| +suggest(req,res,next)  +history(req,res,next)  +changeProfilePic(req,res,next)  +getProfilePic(req,res,next)  +viewProfile(req,res,next)  +rate(req,res,next) |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| Router | Express.Router | private | router<> NULL |
| dataManager | DataManager | private | router<>NULL |
| authenticationHandler | AuthenticationHandler | private | <> Null |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-condition | Post-condition |
| suggest | public | Void | Express.Request req,  Express.Response  res, |  | Authentication result sent to client |
| history | public | Void | Express.Request req,  Express.Response  res, |  | The history  sent to client |
| changeProfilePIc | public | Void | Express.Request req,  Express.Response  res, |  | Acknowledgment sent to client |
| getProfilePic | public | Void | Express.Request req,  Express.Response  res, |  | Profile picture sent to client |
| viewProfilePic | public | Void | Express.Request req,  Express.Response  res, |  |  |
| rate | public | Void | Express.Request req,  Express.Response  res, |  | Rating added to the database |

# 

# References

## Books

Ian Sommerville (2011). Software Engineering 9. Boston: Pearson Education, Inc.

## Web resources

Tutorials point tutorial on :

* Component diagram
* MVC architecture

<http://www.tutorialspoint.com/>

Wikipedia on:

* Component diagram