****

**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 13](#_Toc509606794)

[3.1. Class Diagram 13](#_Toc509606795)

[3.2 Sequence Diagram 14](#_Toc509606796)

[3.1. State Chart Diagram 23](#_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

**UI** – User Interface

**DBMS -** Database Management System

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

**SRS**– System Requirement Specification

**API -** Application Programming Language

**DB –** Database

# 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 application being designed is named “vibe”, it is a android 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

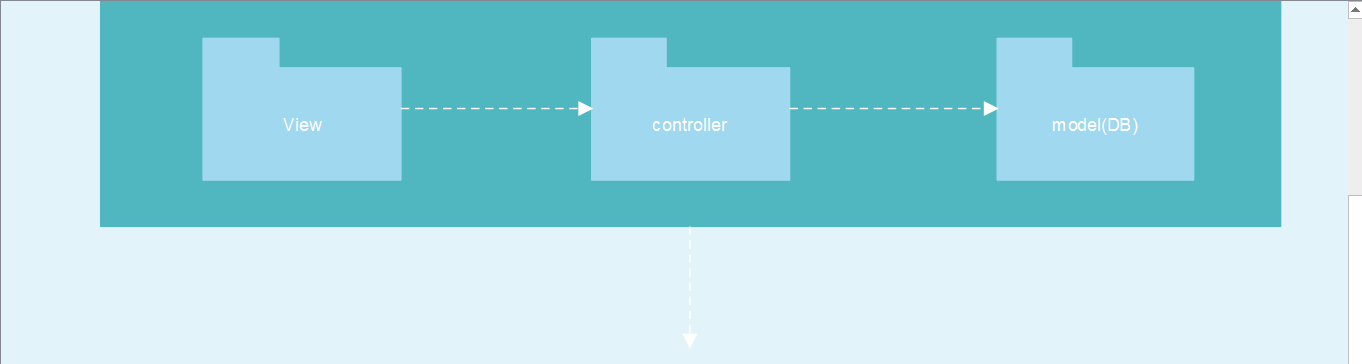
We will be using the staged delivery method because of the time constraint to modify the requirements continuously and our understanding of the system requirement. The implementation will take a Model-View-Controller system architecture separate internal representation from the ways that information 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.

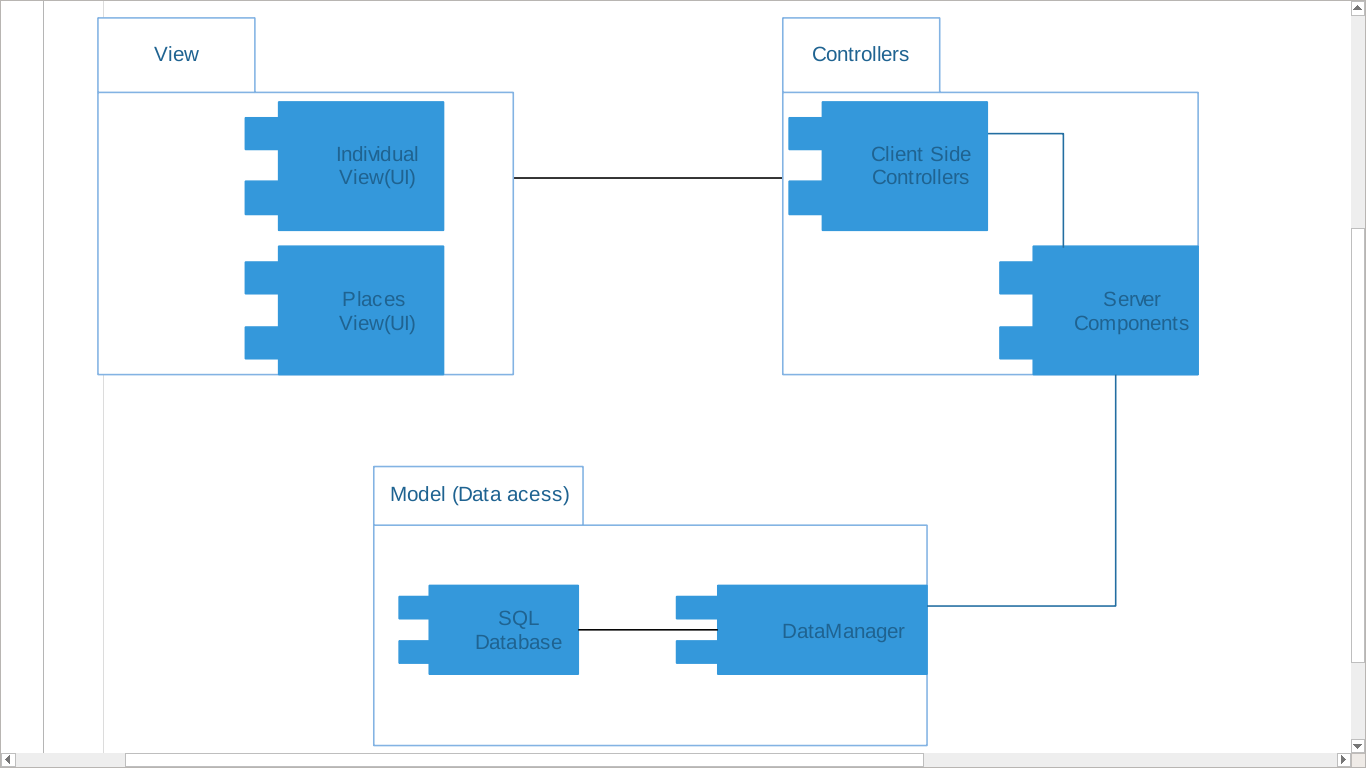
The database will be designed using Mongo DB, a document oriented database program which uses JSON like documents with schemas. We chose this database because it provides high availability with replication sets, can scales horizontal .Mongoose is used to communicate with a database, define schemas and 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

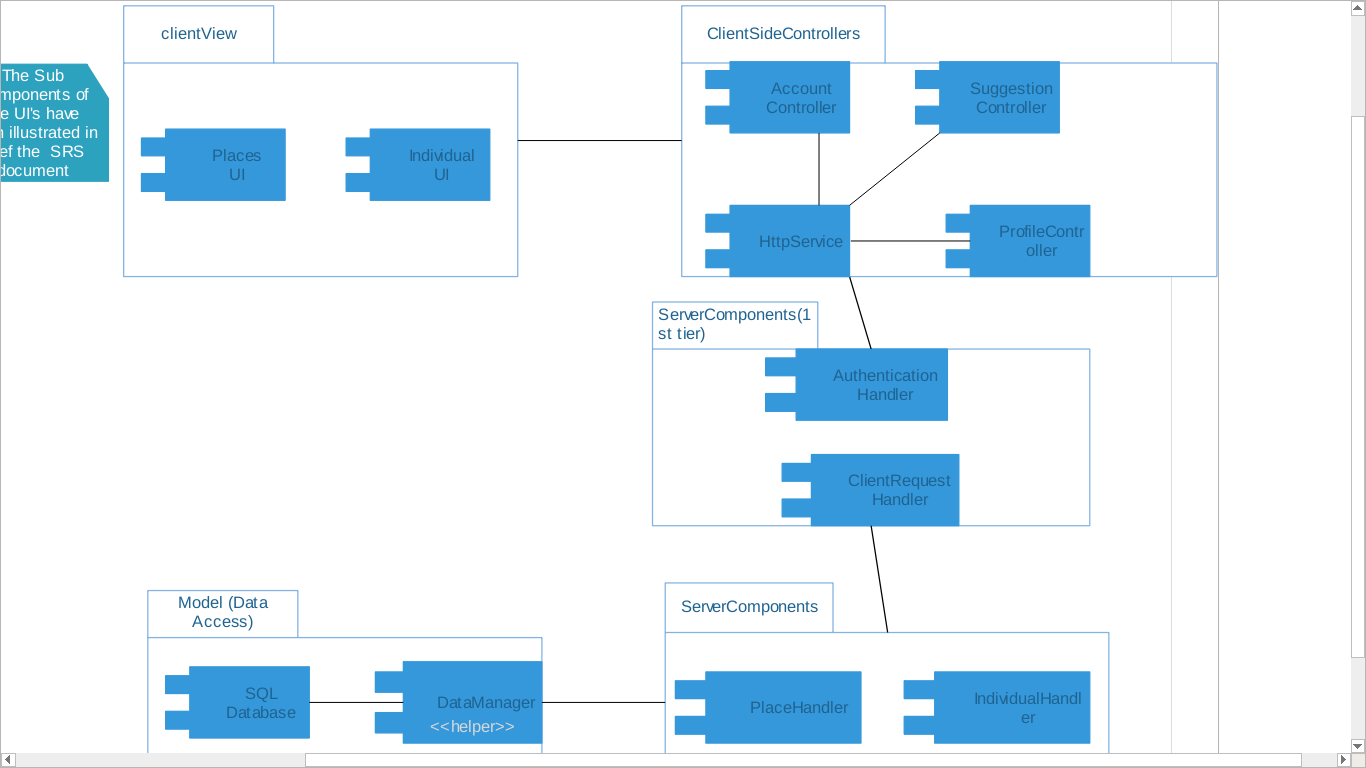


*Figure 2.1.1 Layer 1 component decomposition*



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

**Application**

**Data base**

**Server**

*Fig 2.2 UML Deployment diagram*

# Object Model

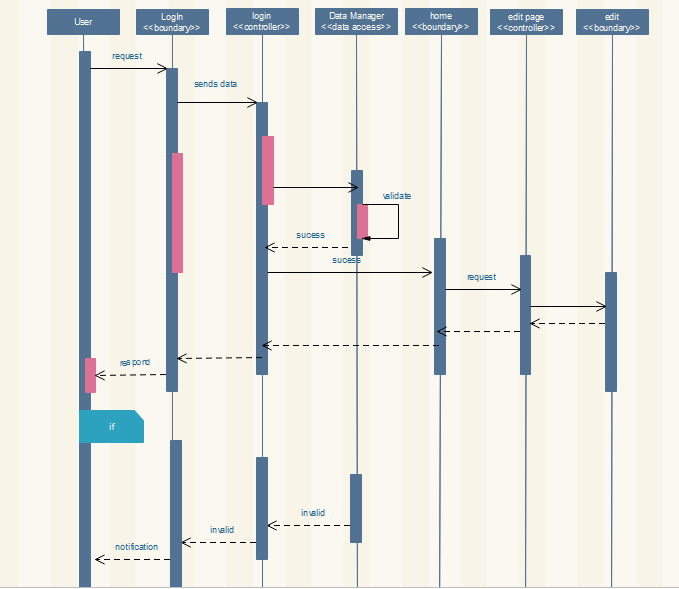
## Class Diagram

## new uml

*Figure 3.1 Class Diagram*

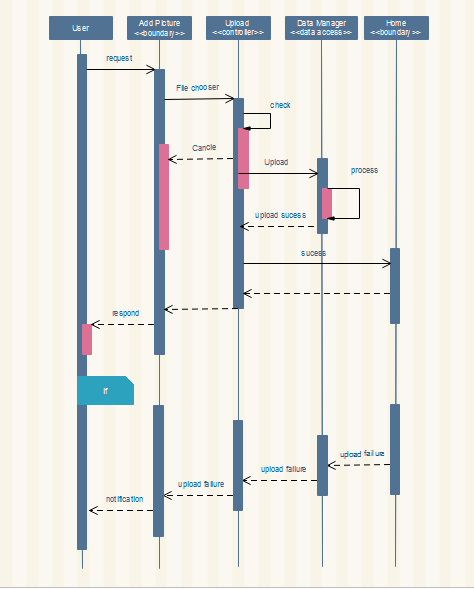
## 3.2 Sequence Diagram

* + 1. Customize profile



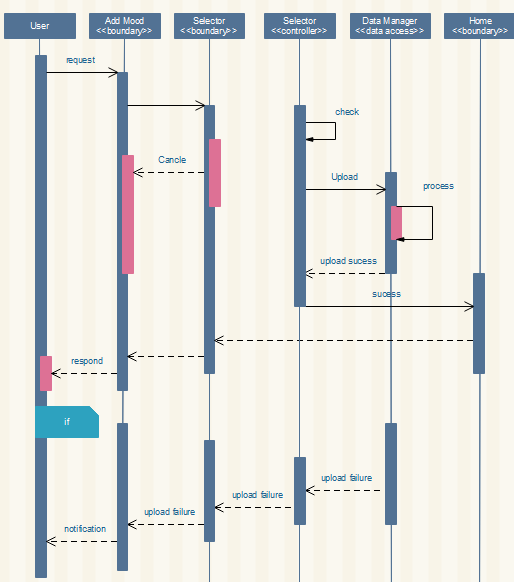
*Figure 3.1.1 Customize profile*

* + 1. Add profile picture



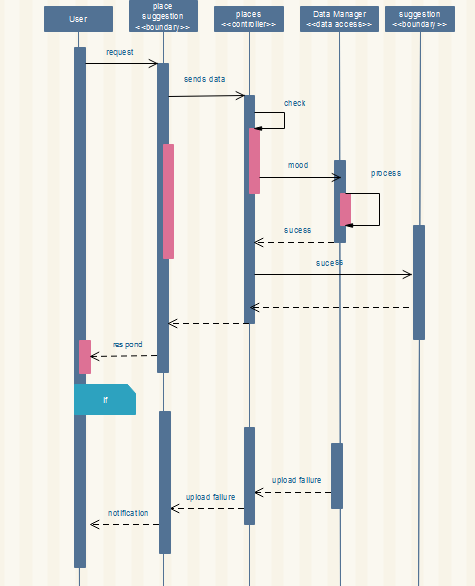
*Figure 3.1.2 add Profile Picture*

* + 1. Specify mood/vibe



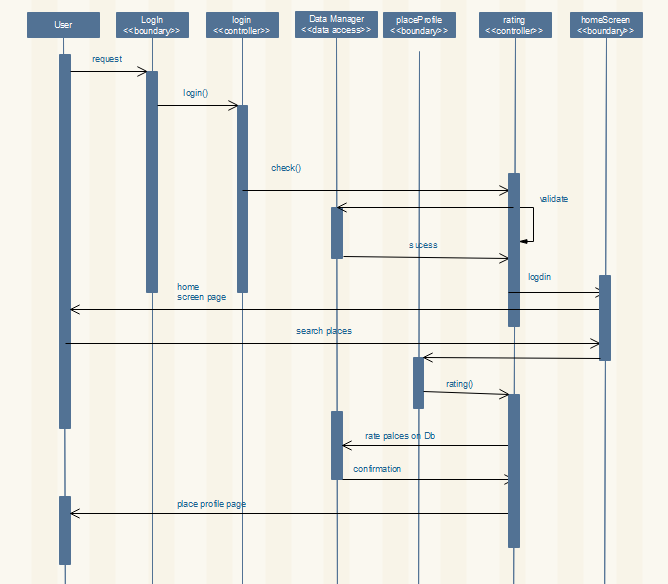
*Figure 3.1.3 specify mood*

* + 1. Request suggestion



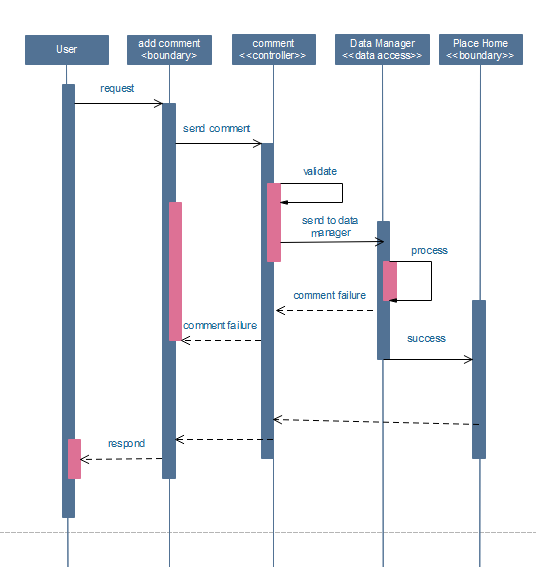
*Figure 3.1.4 request suggestion*

* + 1. Rate



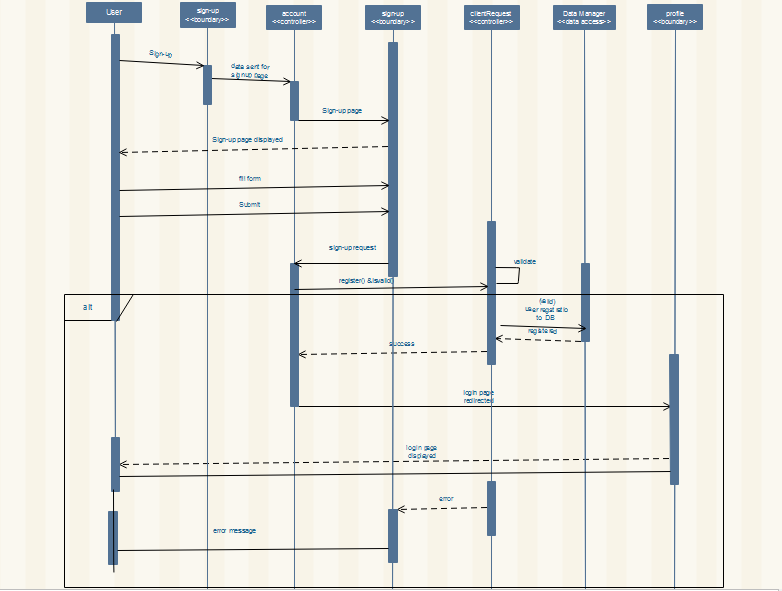
*Figure 3.1.5 Rate*

* + 1. Comment



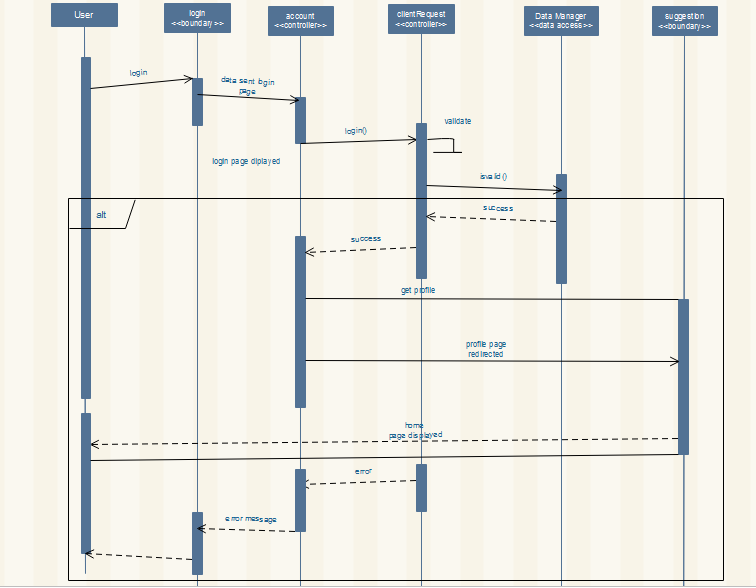
*Figure 3.1.6 Comment*

* + 1. Sign up



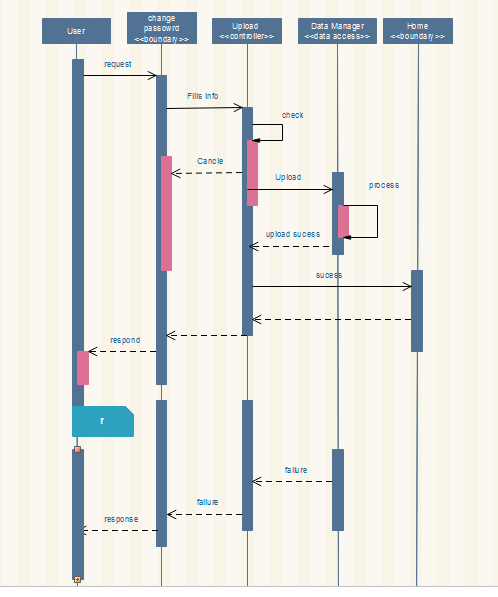
*Figure 3.1.7 sign up*

* + 1. Sign in



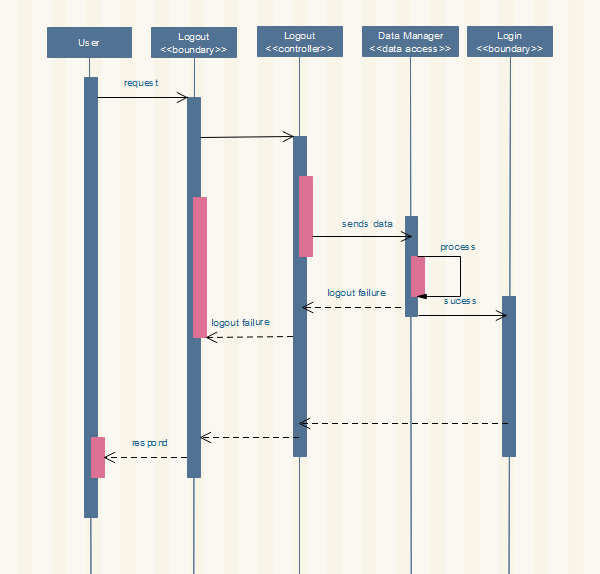
*Figure 3.1.8 sign in*

* + 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: String  #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  ^#userId:int |
| +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 |
| +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 |
| 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 oldPwd, String newPwd, int uid):void  -specifyMood(String type, String mood, int uid):void  +changeProfilePIc(String type , String path,int uid):void  +viewProfile(int pid):void  -rate(int rating , int pid , int uid):void  +getHistory(int uid):void  -Suggest(int uid, String mood):void  -loginRequest(String Uname, String pwd):void  -registerPlace(String userName, String email, String password, Location location, String vibe):void  -registerIndividual(String userName, String email, String password):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  . |

|  |
| --- |
| **Suggestedcontroller** |
| -suggestedPlaces:String[]  -rating:int |
| +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. |
| rating | int | private |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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()  +onReceiveResult(resultCode:int, data:Bundle):void |

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |
| +suggest | public | void | None | User has logged in successfully |  |

|  |
| --- |
| **CustomizeProfile** |
| -currentUser: User |
| +changeProfilePic()  +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. |
| 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  -mongoose:mongoose  -PORT:int |
| +getIndividual(int uid):Individual  +getPlace(int pid):Place  +changePassword(int id, String newPassword, String type):  +addRating(int pid, String path) |

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| Mongoose | mongoose | Private | Must containe a valid database user |
| PORT | int | Private | Not Null |
| 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. |
| AddRating | Public | Void | Int pid, int uid, int rating | The user should exist in the database |  |
| changePassword | public | void | Int id, String type,String newPassword | User has supplied necessary old and new password. | User’s password should be changed |

|  |
| --- |
| **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