High Level Design Document

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 26/09/2015 | 1.0 | Initial Version | Chandrima Chattaraj |

# Overview

This document provides a high level design for initial version implementation of the web application which will let a new user sign up, fill his/her various details and help him to ask queries to people related to same interest and somewhat proficient on the particular topic.

# System Requirements

## 2.1. Expected Behavior

1. User whoever first visits the app, needs to create a profile of his own, thus signs up into the application filling a form with basic details.
2. After creation of the profile, user is redirected to the login page, and logs into the app for the first time with user ID and password.
3. User is redirected to an empty dashboard, and he cannot post any query as his domain of interest is yet to be filled.
4. User needs to go to his personal details section and has to fill details of interested fields, his experience in that particular topic, vote himself/herself according to the his/her proficiency.
5. User can choose as many as sections they are interested in, and raise queries based on any of those sections. After this, the user turns into a member of the domain of his interests.
6. Members of the domains will voted based on the answers or solutions they provide to the queries asked by other members.
7. On navigating to the Dashboard, members can query on any of their selected domains and migrate to the corresponding page on search.
8. After entering the query into the given area, they need to choose whether they need expert advice or submit the query to the defaulted list of interested members of that domain.
9. On selecting expert’s advice, they can choose experts having votes only as same as their personal vote or higher.
10. On submitting the query, a notification gets triggered for all the members who are targeted to answer the query by the submitter.
11. Also a notification will hit a member, if any of the member has responded to the query the former has been tagged to.

## 2.2. Software Requirements

1. **IDE** – Eclipse
2. **Database –** MySQL
3. **GitHub setup**

## 2.3. Languages To be Used

## Front End

1. HTML/CSS (Scripting)
2. Ajax - Combination of XMLHttpRequest object (to exchange data asynchronously with a server) JavaScript/DOM (to display/interact with the information) CSS (to style the data))
3. JavaScript ( For Validation)

## Business Logic

1. Java with *Spring/Struts* Framework (Mainly for Dependency Injection) with *Apache* server

2. Java Server Pages (JSP)

3. Web Services

1. XML

# Data Model

# 1. User [User\_Id, User\_name Password, Email\_address, Phone\_number]

# 2. Member [User\_id, Domain\_id, Member\_id etc.]

**FLOW OVERVIEW**

**MODULES**

1. View Module (Lican)
2. Business module (Phoenix)
3. Database module(BloodLine)
4. Controller modue ()
5. Batch module(Odin)

**Control Flow**

Flow of control starts from the view module which in turn is the UI. The UI will be the first mode of interaction. This View will interact and pass on data to the controller.

The controller decides where and how to pass the data. This data is in turn passed to the business module where the data is processed and returned to the controller to pass on to the view module where the user can get it.

The business module interacts with the database module for retrieving or storing data.

Batch Module is the module which is in charge of all the batch jobs and is totally independent of the other modules. This batch module has jobs scheduled to run at intervals. This batch jobs will do all the housekeeping and maintenance of all the files. This will take care of the scheduled newsletters, notifications, processing of held jobs. This module will be able to interact with the database module in order to gain access regarding user, admin etc.