|  |
| --- |
|  |
| **KnowlegeBase****Final Report** |
|  |
|  |
| **Steve Cho****John Hoang****Oliver Jang****Jeffery Wasty****Boning Yang** |
|  |

## 

Table of Contents

[1. Document Version](#_jzc5vrhdqvg6) 3

[2. Contact Information](#_yjullqpi8sp3) 3

[3. Description](#_ezmkkmhu4inf) 4

[4.1. Non-Technical Assumptions](#_gquhjvx3rw22) 4

[4.2. Technical Assumptions](#_qojw7m26x0o5) 4

[5. High-Level Overview](#_t4rt0j1174n5) 4

[5.1. Routes](#_87pk4cz37gf6) 5

[5.2. Controllers](#_hzvm5rkt3u16) 6

[5.3. Views](#_6v4r4u2tzkfj) 7

[5.4. Models](#_caqkfd7srj6e) 8

[5.5 Database Design](#_14xuzhd4ns9c) 9

[6. High-Level Schedule](#_x3tjeiumht4r) 9

[7. Risks and Issues](#_22u3i37lyq5i) 10

[8. Learnings](#_3c7jvi2a0gsh) 10

[9. Appendix](#_kb988idzhgna) 11

9[.1. ER Diagram](#_8l79bpnp6ss1) 10

9[.2. Sequence Diagrams](#_8340jbv0cu68) 11

9[.2.1 Signup, login, logout](#_nmtgkmwmj5e) 11

9[.2.2 Create post, search posts, filter posts](#_so0tg2bd1418) 11

9[.2.3 Enable likes, like user](#_1a8sx5loliv7) 12

9[.2.4 Message user](#_f9p7xy4myy2u) 12

## **1.** **Document Version**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Description** | **Date** | **Author(s)** |
| **1** | Initial Version | 14/11/2019 | Steve Cho  Oliver Jang  John Hoang  Jeffery Wasty  Boning Yang |
| **2** | Final Version | 5/12/2019 | Steve Cho  Oliver Jang  John Hoang  Jeffery Wasty  Boning Yang |

## **2.** **Contact Information**

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Email*** | ***Role*** |
| Steve Cho | swcho0182@gmail.com | Fullstack Developer |
| Oliver Jang | jangoliver@gmail.com | Fullstack Developer |
| John Hoang | johnquochoang@gmail.com | Fullstack Developer |
| Jeffery Wasty | jeff.wasty@gmail.com | Fullstack Developer |
| Boning Yang | sylvartore@gmail.com | Fullstack Developer |

## **3.** **Description**

KnowledgeBase is a social media platform that allows users to share their expertise around a topic, search for expert help regarding a topic, and match with other users interested in certain topics.  
  
The application will be constructed using a SQL database, holding user, message, and post data. The frontend will present this information through a user feed, profile page, and instant messaging client.

**4.** **Assumptions**

## **4.1.** **Non-Technical Assumptions**

* Project scope is not subject to change.
* Project deadline is not subject to change.

## **4.2.** **Technical Assumptions**

* Node dependencies will not break by update.
* Website will not need to be scaled, i.e. no need to be mobile friendly.
* Website will be tested on Google Chrome.

**5.**  **High-Level Overview**

## **5.1.** **Routes**

|  |  |
| --- | --- |
| **Name** | **Description** |
| / | View: signUp.  Controller: authController |
| POST  /signup | Controller: authController  Redirect: /register?id={userID}  Post route to sign up users, and redirect to /register for further information input that user’s ID |
| GET  /register?id={userID} | View: register  Controller: authController  Use ID from query param to append additional info to user details |
| /homePage/:id | The main page including user information, includes partials for questions and discussions.  Controller: profileController, discussionController |
| /search/:topic | Gets any discussions that match the search query  Controller: discussionController |
| /user/profile/:id/allPosts | A list of all posts made by the user |
| /message | View: messageStart  Controller: messageController to send messages |
| /conversation | View: message  Controller: messageController |

## **5.2.** **Controllers**

|  |  |
| --- | --- |
| **Name** | **Methods** |
| authController | signup   * Post user registration info from ‘/’ and user ID to DB * On success, render to /signup for second part of signup * Input: form data   includeDetail   * The second part of the signup process where extra details are added to DB using user ID * Input: form data   login   * Post login info and reroute to get user/profile/:id upon success * Input: user id/password |
| profileController | serve   * Serve the profile to the user. Calls loadPosts. * Input: user id   loadPosts   * Loads user posts from database   update   * Update the information fields on user profile. * Input: name, description, location (all default null)   increaseLikes   * Increase the number of likes on page. |
| discussionController | addPost   * Add a new post to feed * Input: subject, explanation, topic |
| messageController | getConversationList   * Loads message sessions for the user   getConversation   * Loads historical messages belongs to selected message session   Input: message session id  sendMessage   * Sends message and start a new conversation session if not exist * Input: receiver id |

## 

## **5.3.** **Views**

|  |  |
| --- | --- |
| **Name** | **Description** |
| main | The search by subject, and logout header and kb logo, common to all pages besides sign up and registration |
| profile | User profile page: includes info, userPostsPartial, likes |
| message | The real-time chat functionality |
| signUp | The sign up page, has login and first portion of user creation |
| registration | Additional form input for user |
| search | Search page: includes userPostsPartial |
| messageStart | Creation of a message |
| questionPartial | What is your question today? |
| discussionPartial | Condensed post/discussion before expansion with search by topic, includes the All posts in the profile view, the result of search by subject, result of search by topic, all posts by X user |
| discussionsAndPosts | After clicking replies the expansion |
| userProfileSummaryPartial | The sidebar with the user image, likes, posts, msgs etc. |

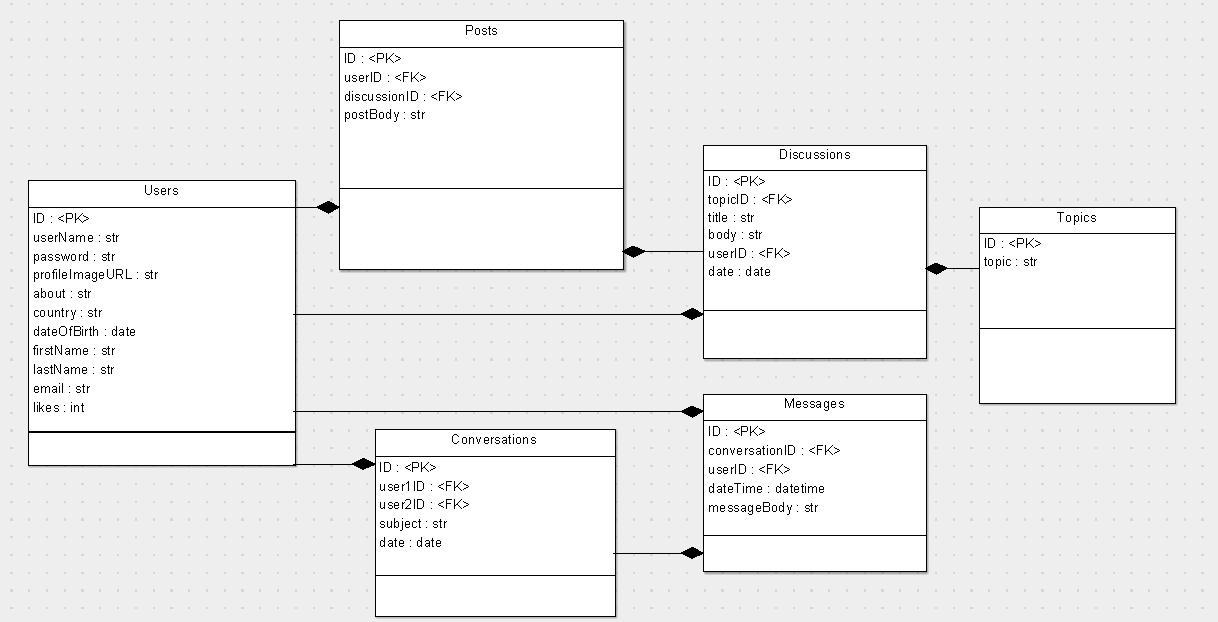
## 

## 

## **5.4.** **Models**

|  |  |
| --- | --- |
| **Name** | **Methods** |
| profileModel | validateUserLogin   * Check if user + password combination exist in database and return private info if they exist.   getPrivateInfo   * Returns all user info for a specified ID.   getPublicInfo   * Returns public user info (info other users are allowed to see) for a specified ID.   modifyUser   * Update various fields for a user.   createUser   * Add a new user to the database. |
| discussionModel | getUserDiscussions   * Return all discussions created by a specified user ID.   getAllDiscussions   * Returns all discussions ordered by descending date.   getDiscussionsBySearch   * Returns all discussions matching search parameters.   getPostsForDiscussions   * Returns all posts (replies) to a specified discussion ID.   createDiscussion   * Add a new discussion to the database.   createPost   * Add a new post to the database attached to a specified discussion. |
| conversationModel | getUserConversations   * Return all conversations involving a specified user ID.   getMessagesForConversation   * Returns all messages ordered by ascending date.   createConversation   * Add a new conversation to the database.   createMessage   * add an associated message to the database. |

## **5.5** **Database Design**

**

**6. High-Level Schedule**

|  |  |  |
| --- | --- | --- |
| **Owner** | **Description** | **Completion Date** |
| Oliver | Construction of database and tables | November 20th |
| Steve | Sign up, authentication, log in | November 20th |
| John | Add posts to discussion, search for posts | November 20th |
| Boning | Messaging client and messaging between users | November 20th |
| Jeff | Profile: info, discussions, front end | November 20th |

## 

## **7.** **Risks and Issues**

|  |  |  |
| --- | --- | --- |
| **Risk or Issue Description** | **Potential Impact** | **Mitigation/Resolution Steps** |
| Integration of database with web page. | High | Stay ahead of schedule, finish database design and integration as a high priority. |
| Security issues storing user data | High | Take proper steps to ensure data security. |

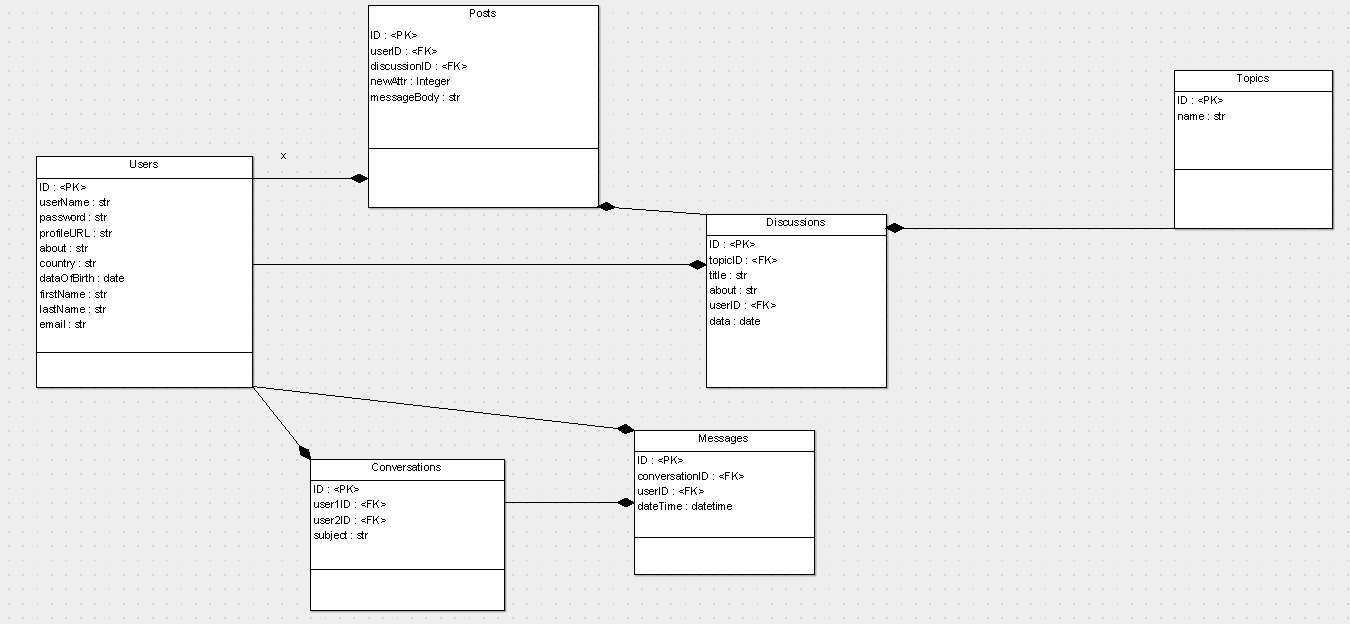
## **8.** **Learnings**

In terms of knowledge, having an in-depth experience into the development of a forum based website revealed the value in employing an MVC structure. Creating partials and views for handlebars allowed for extensive code reuse and minimal work in creating global changes.

Working in a team environment showed the importance of version control in creating a larger scale project, streamlining integration and concurrent code changes. It also showed us the value in investing in the creation and implementation of isolated environments as to not obstruct other developers from their own work.

## **9.** **Appendix**

## **9.1.** **ER Diagram**



## **9.2.** **Sequence Diagrams**

## **9.2.1** **Signup, login, logout**

A screenshot of a social media post

Description automatically generated

## 

## **9.2.2** **Create post, search posts, filter posts**

A screenshot of a cell phone

Description automatically generated

## **9.2.3** **Enable likes, like user**

A screenshot of a cell phone

Description automatically generated

## **9.2.4** **Message user**

A screenshot of a cell phone

Description automatically generated