ASHESI UNIVERSITY COLLEGE

EXPERT FINDING SOCIAL NETWORK FOR INFORMAL AND SEMI-FORMAL EXPERTS

APPLIED PROJECT

B.Sc. Computer Science

David Haq Thorndollawende Inusah

2017

ASHESI UNIVERSITY COLLEGE

Expert Finding Social Network for Informal and Semi-Formal Experts

APPLIED PROJECT

Applied project submitted to the Department of Computer Science, Ashesi University College in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science

David Haq Thorndollawende Inusah

2017

DECLARATION

I hereby declare that this applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

.....……

Candidate's Name:

.....…….....

Date:

…… ……

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor's Signature:

……...

Supervisor's Name:

……..……

Date:

……

Acknowledgement

I would like to thank God for the grace and presence of mind to complete this project. All the glory goes to Him. I would also like thank my supervisor Mr Stephane Nwolley for his patience and gentle direction in the progression of this project. He has been a great source of encouragement for me. I would also like to thank my parents, Mr and Mrs Inusah for their support in prayers and various efforts to encourage me to complete this project. I would also like to thank Antoinette Doku for taking an interest in my project when it seemed that I was progressing slowly in developing it. I would also like to thank Mr Aelaf Dafla for encouraging me at some point in this project. Finally I would like to thank Dr Korsah, the Head of the Computer Science Department for all her efforts in making sure that all of us students undertaking Capstone projects this year kept on track during the development of our projects. God bless you all.

Abstract

Patatte is a simple social network that helps people to find and collaborate with informal and semi-formal experts. This project's aim is to solve the 'expert finding+ problem' which is the problem of finding an unknown expert (for any purpose including hiring, working with or questioning the expert) by searching through real world or online social networks with skill or interest keywords.

The system is designed to be simple and intuitive to use. This is because some of the targeted users (informal and semi-formal experts) are basically educated and may have limited experience with the usage of web applications and social networks. The system is also accessible by the most basic internet user requiring a computer, internet access and a basic knowledge of how to use social networks.

Users can share projects they have worked on, follow other users, view those users' projects, like some of those projects, search for users with specific skills and collaborate with them in an integrated media sharing and chat application.

Table of Contents

List of Abbreviations

Abbreviation	Full meaning
API	Application Program Interface
SQL	Structured Query Language
HTML	Hyper Text Mark-up Language
PHP	Personal Home Page
CSS	Cascading Style Sheets

Chapter 1: Introduction

1.1 Background

Collaboration has always existed in man's history, being demonstrated in various ways. There were barter traders that exchanged goods for services they could otherwise not provide themselves. Even today, with currency, we still need to collaborate similarly. There are fishermen that need carpenters to make them boats and carpenters that need fishermen to sell them fish.

In recent times, collaboration is no longer localized. It is occurring on a global scale, with intercontinental collaborations being common place thanks to the internet. The resulting data generated as a result of this global connection is growing at an overwhelming rate. The users that create this data are virtually represented as interconnecting nodes in a vast network. This network mimics the real world network of people and how this real world network functions.

Though this advent has been great at connecting the world irrespective of boundaries, time differences and geographical barriers, some of the real world network problems still persist in this virtual network. One real world problem that has persisted and even become enlarged in this virtual network is the "Expert Finding Problem".

The "Expert Finding Problem" is the problem people face in finding an expert they do not know but whose skills they desire. People often have to find such unknown experts with desired skills from a large collection of people/users in a real world or virtual network. They often do this by asking people they know if they in turn know any such experts. This problem has become more complex or enlarged in the virtual network because a person's virtual network has the potential to be larger and is more interconnecting (because it overcomes

geographical challenges to connect users) than his/her real world network. Based on this sheer size difference, it is more difficult to find an unknown expert on a given topic in the large mass of interconnected users across the entire planet on the virtual network than it is for a person to find a relevant expert to his problem by physically asking people he knows (searching his real world network). That said, the searching user is more likely to find a strongly qualified expert to collaborate with when he/she searches the virtual global network because he/she is exposed to a larger sample size.

1.2 Project Solution

Imagine looking for a rare expert, say a griot, from among the network of people you personally know or do not know, but physically have access to. You would have a large variety of related and unrelated experts to search from in this network you are looking through. Let us call this an unordered network. Searching for an expert you can only identify by their skills from an unordered network is a very hectic search even with people's recommendations and help.

To solve this, I would first order the network so that finding a person within it would depend on the category of people they fall into. This way, whole sections of people will be cut off from the search and things would be made easier. To order the above narrated network, we could create the following subcategories of networks: work, friends and market. In order to search for a griot, it would be better to search the market.

Within the market, there should also be an ordering that allows a searching user to further narrow down their search area. In addition, there could also be a market enquiry center that provides searching users with individual profiles of people in the expert field that the user is

searching for. This cuts the chase down tremendously. The user can now select any griot he/she wants from a known collection of available griots by any criteria he/she desires to use.

This is the solution my project proposes. Patatte is a specialized sub network of the entire unordered internet. It is specialized in the sense that, it is meant only for interest/talent/skill based project enthusiasts. Therefore people searching for friends with certain social skills would not waste their time searching through Patatte. They would rather search Facebook or Twitter.

Within Patatte, all users will be within categories to further order the network. Beyond this, there will be a search feature that easily allows a cup cake maker to find the best web designers the network has to offer at the click of a button. This has the potential to take away a lot of the hustle, questioning around and unsatisfied workmanship that comes with a person searching his/her entire real world network or even bigger virtual network for an expert. This is the reason why vast general social interaction networks like Facebook and Twitter are not ideal for solving the expert finding problem.

This project sets out to focus specifically on connecting people with informal experts (including painters, carpenters, bakers etc) and semi-formal experts (including photographers, graphic designers, web developers etc).

1.3 Related Work

1.3.1 Academic Papers

A research paper titled 'Discovering Experts Across Multiple Domains' by Aditya Pal, an IBM researcher describes an expertise finding framework that analyses documents on a variety of aspects including language, questions, topics and an indicator of expertise. The aim

is to extract key features that can help determine the expertise of an author through the documents they share.

It is different from some of the other expert finding systems I have encountered in that it searches for experts across domains as opposed to searching for experts within the same domains like single social networks.

The paper presents an algorithm that accepts a user's work (documents shared etc) and computes an expertise score for the author. It also indicates that one of the rare related computations for expertise across domains like the one it has explained is the Lucene based model to index multi- domain documents and combined document relevance, and popularity to compute user expertise.

This work is relevant to my project in that even though I am trying to find relevant experts for users within one social network, it would be very helpful in creating a ranking for user expert searches if my system could compute an objective expertise level of all the system's users (Pal, 2015).

Another related academic work to my project is a paper titled 'A Model for Expert Finding in Social Networks' by Elena Smirnova. This paper develops a Bayesian hierarchical model that also makes attempts to account for a social layer in expert finding where the experts found will be linked to the searching user by one or more of the people in his social circle.

The model in the paper is built closely on the AT model that extends the latent Dirichlet allocation model by adding authorship information. This addition helps create an expert finding system that not only finds relevant experts to a user's search based on their

work shared on the network, it also makes sure that there is a high chance that the two users can work together because they commonly know some other users.

This approach was built heavily on the assumption that people within a user's current social circle on the network have shared topical interests. Though that leads to a useful expert finding solution, it is also an assumption that is not very true as some people add other people to their social circles simply because they know them and may not have any similar topical interests. In such cases, the results of the search using this system could be compromised returning irrelevant experts to the user (Smirnova, 2011).

Another related academic work to my project is a paper titled 'QuME: a mechanism to support expertise finding in online help-seeking communities' by Jun Zang, Mark Ackerman, Lada Adamic and Kevin Nam. This paper describes a method of computing a user's expert level for a system called the QuME (question matching engine) by analyzing the information they create on a social network. It considered a case in point; the Java Forum where behaviors of users with high expert levels, intermediate expert levels and low expert levels were observed. It was seen that questions have to be matched to specific users based on their expert level because otherwise there is usually an imbalance in which experts answer questions more quickly but get their questions answered much less quickly because experts are likely to ask hard questions. Also, low expert users also get their questions answered more quickly because they most likely ask basic questions. On the other hand, they may not understand the answers given by high expert users because the answers may be complex. This imbalance has required that questions are distributed more equitably on the Java Forum. Other considerations that were also made in the paper were the motivations for which people helps strangers on such social networks. From their research it was identified that some of these motivations include;

altruism, incentives to support one's community, reputation-enhancement, expected reciprocity, and direct learning.

The QuMe and the insights the paper explains are necessary to my project because it is important that people that will be recommended to people looking for other people to collaborate with on some projects must not be too far apart in expertise. It would be useful if the system could recommend experts that have expert levels close to the user's expert level (Zhang, Ackerman, Adamic, Nam, 2007).

A final academic paper that related with my work is a paper titled 'Exploiting Social Context for Expertise Propagation' by Greg Millete, Michael Schneider, Kathy Ryall and Robert Hyland. This paper looks at an interesting idea of selecting only the most relevant experts that are recommended so as not to overburden experts with too many requests for help. The paper explains such functionality within a system called College Search which provides summaries of recommended experts' expertise, and a social context (e.g. job titles and organizational chart location). This allows users self investigate more closely into who they want to work with. They can even view the recommended users' profiles and see their work. This helps narrow the requests each expert searching user sends out hence reducing the number of requests expert users receive.

This is a significant alternative approach I could apply to my project. It is especially much simpler to achieve. (Milette, Schneider, Ryall, Hyland, 2009)

1.3.2 Social Networks

1.3.2.1 LinkedIn

LinkedIn is the world largest professional network recording an active monthly usage of 106 million users as of November 2016 (LinkedIn Facts, 2016).

Advantage:

LinkedIn is a highly intelligent social network with an extensive use of machine learning to predict user profile growth based on certain specific user connections. It is also very broadly solves corporate HR needs like talent searches, C.V uploads, professional profile access, and Job posts.

Disadvantage

LinkedIn is targeted at formal workers. The complexity of LinkedIn profile pages, their heavy demand for user information and the networking skill that is needed to create a vibrant LinkedIn profile may very easily turn off an average carpenter that wants to take advantage of the global reach provided by the internet to market himself/herself. Informal experts prefer simpler content sharing social networks like Fiverr that allows them to easily share their work without so much etiquette.

1.3.2.2 Fiverr

Fiverr is an informal expert finding application that facilitates the sale of users' services to one another.

Advantage

Fiverr very conveniently and simply connects you to the services of semi formal experts like graphic designers.

Disadvantage

Fiverr emphasizes mainly on selling user's services to one another, when some users simply want collaboration partners to work with on a project without having to pay any money.

Chapter 2: Requirements Gathering and Analysis

This chapter introduces in detail the requirements that the system must meet in order to be useful to the "Expert Finding Problem" discussed above. It also describes the processes by which these requirements were gathered.

2.1 User Description

The intended users of this system are expected to be between the ages of 18 and 30. The system is targeted towards college students and young adults that are developing interests and talents they have developed earlier in their lives. Also, the young adults in this age group of users are active in creating projects to help settle/stabilize themselves financially for a better life or skillfully for better job prospects.

2.2 Requirement Gathering Techniques:

2.2.1 Brainstorming: The initial requirements of the system were developed by using the brainstorming requirements gathering approach. By focusing the expert finding problem identified I came up with a basic set of requirements that I wanted the system to fulfill.

2.2.2 Use cases: Another requirements gathering technique that was used was use case analysis. By outlining the flow of use of the system by the identified users, more unseen requirements were identified.

2.2.3 Prototyping: This is the final requirements gathering approach that was used in this project. The requirements gathered this far were used to prototype a mockup of the site. Axure wire framing software was used for this prototyping. I used the prototype to solicit user responses on the project. From their responses and from simply developing the prototype, some requirements were removed and others added in the final iteration that has been developed in this project.

2.3 System Requirements

2.3.1 Functional Requirements:

- Signup for new users and Login for current user.
- Profile details: Users that log in for the first time should be able to input further details about themselves to help setup their profile with. Expected information includes: first name, last name, bio, skills, work shed location (location of their work shop)
- User profile page: There should be a page that profiles the user. Information expected to be provided should be user bio, skills, work shed location, number of people following the user, number of people the user follows, user profile picture and user projects portfolio.
- 'Following' and being 'Followed': A user, say A, should be able to follow another user, say B. 'A' should also be able to be followed by any other user accounts including 'B'. That is, all accounts should be able to follow other accounts and be followed by those accounts as well.
- 'Unfollowing' an account: A user should be able to unfollow the account of another user they are already following. This disconnects their network linkage and so the 'unfollowing' user no longer sees posts from the 'unfollowed' user in his/her News Feed.
- User should be able to upload and share project work in video, audio and image formats.
- Users can delete projects in their portfolios.
- News Feed: A user should be able to see post/project updates of all users they are following in his/her News Feed. Likewise, other following users of that user should see the user's post updates in their News Feeds.
- Users can comment on and like other users' projects/posts.

- Notification system for prompting users about new likes, comments, mentions, follows and collaboration requests.
- Users should be able to 'Mention' other users in their posts. A mention is the ability to mention a user's username (with a link behind it to that user's profile page) in a post, notifying the user of that mention in the process of sending the post.
- Internal search feature that finds and recommends existing users to new users as similar interest users to start off following.
- Search feature for finding other users (experts) in the network by searching with user name keywords.
- Search feature for finding other users (experts) in the network by searching with skill or interest keywords. This feature should return a ranked list of users (experts) based on the distance of their work sheds from the searching user's work shed.
- Sending and receiving collaboration requests: a user should be able to send a request to another user with some desired skills to seek collaboration with them. Similarly, that user should be able to receive collaboration requests from other users seeking collaboration.
- Users should be able to accept or reject collaboration requests sent to them by other users.
- If a user accepts a collaboration request, a chat and media sharing work space should be opened between that user and the requesting user for them to collaborate within.

2.2.3: Non-functional Requirements

- Security: The system should protect user accounts and information from being accessed by attackers or unauthorized users.

- The system should be available for use at all times to every user that has a computer with internet access.
- The system should be intuitively usable to the targeted users.
- The system should be useful in solving expert finding problems of targeted users

2.4 Scope of Requirements for this Project

This project will implement all stated functional requirements excluding

- Creating posts in video and audio formats.
- Returning a ranked search list of experts. Only returns a list of experts
- Username keyword search feature for finding users.

2.5 Project Use Case

The following is a use case rule for a any given system user:

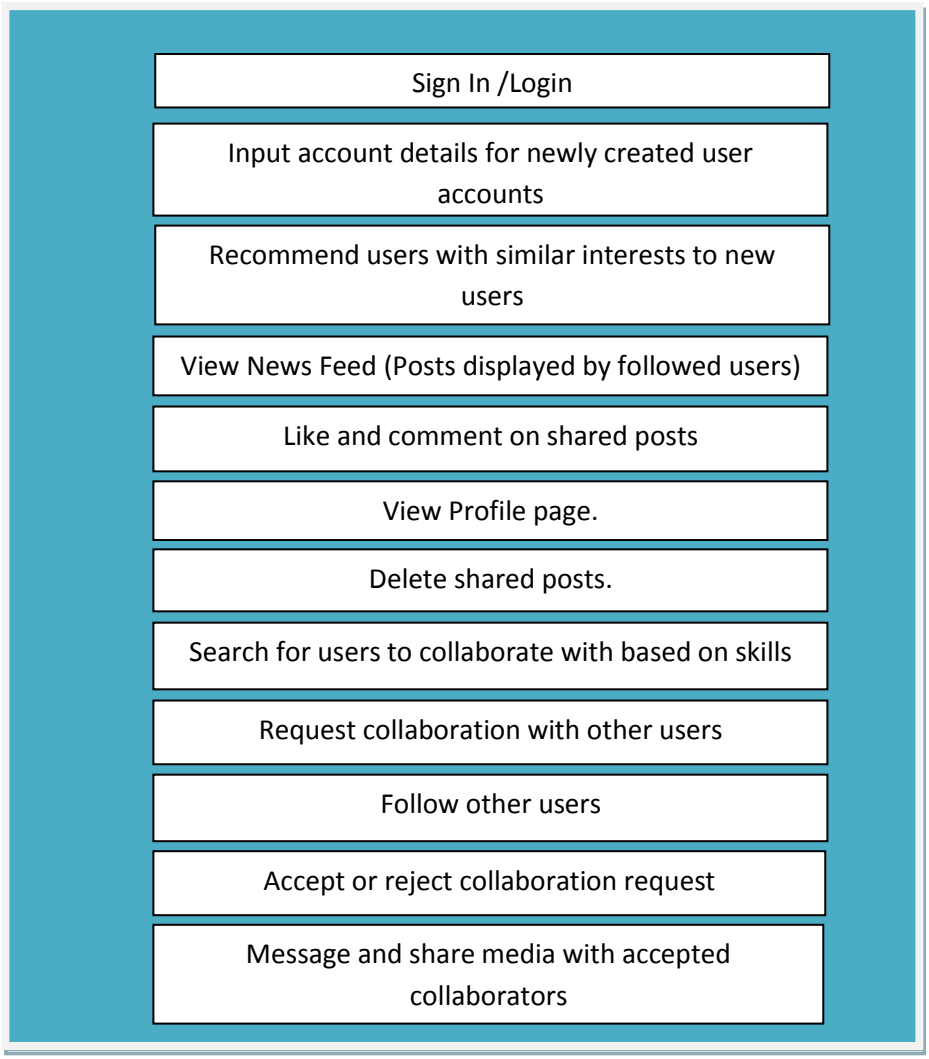

Figure 2.1: Use case rules

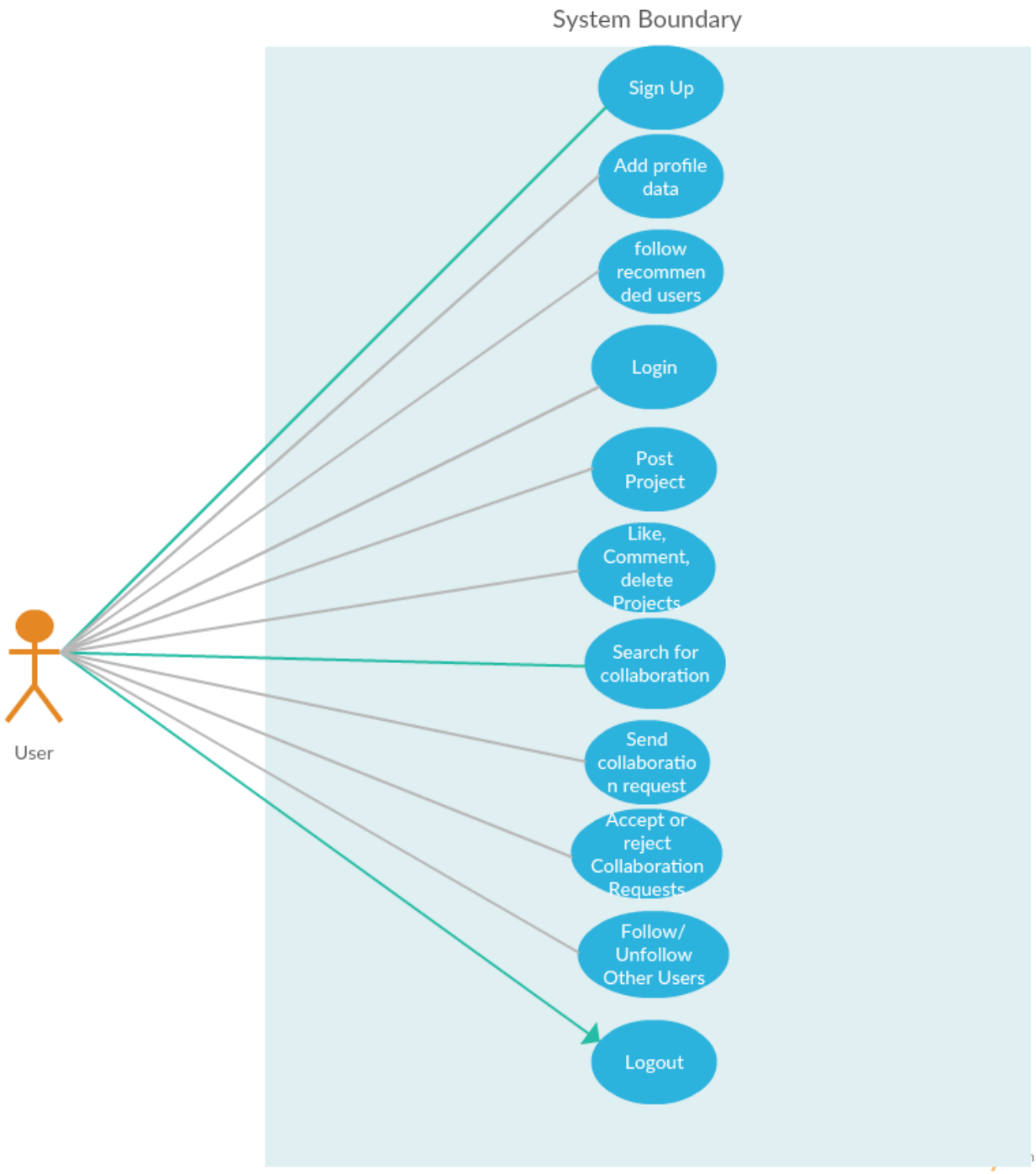

Figure 2.2: Use case diagram

Chapter 3: High Level Architecture and Design

This chapter provides a high level overview of the project, giving more detailed insight into the system's design. It also outlines the arrangement of the various components of the system.

3.1 System Design

The system design is generated at a high level from a series of paper and wireframe prototypes. The design of the flow of the various functionalities and their interaction with one another was developed from these prototypes.

3.1.1 Prototype Screenshots

3.1.1.1 Hand Sketches

The first step I took in the design of the system was to make a hand sketch of what I intended to build. This sketch forms the first iteration of my prototype iterations.

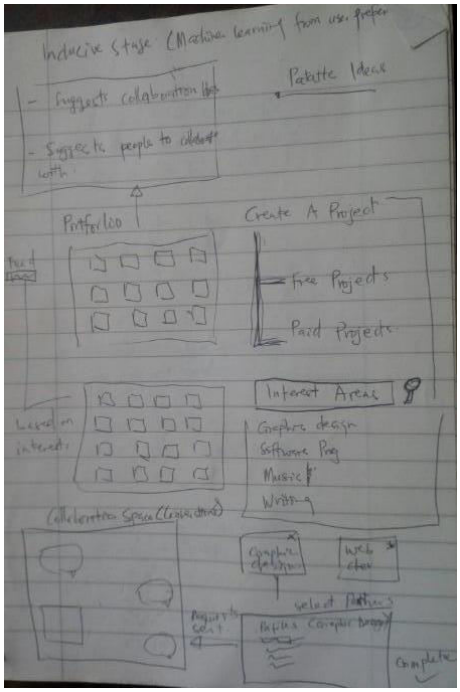

Figure 3.1: Initial hand sketch

3.1.1.2 Wireframe Prototype

These prototypes were generated by using Axure.

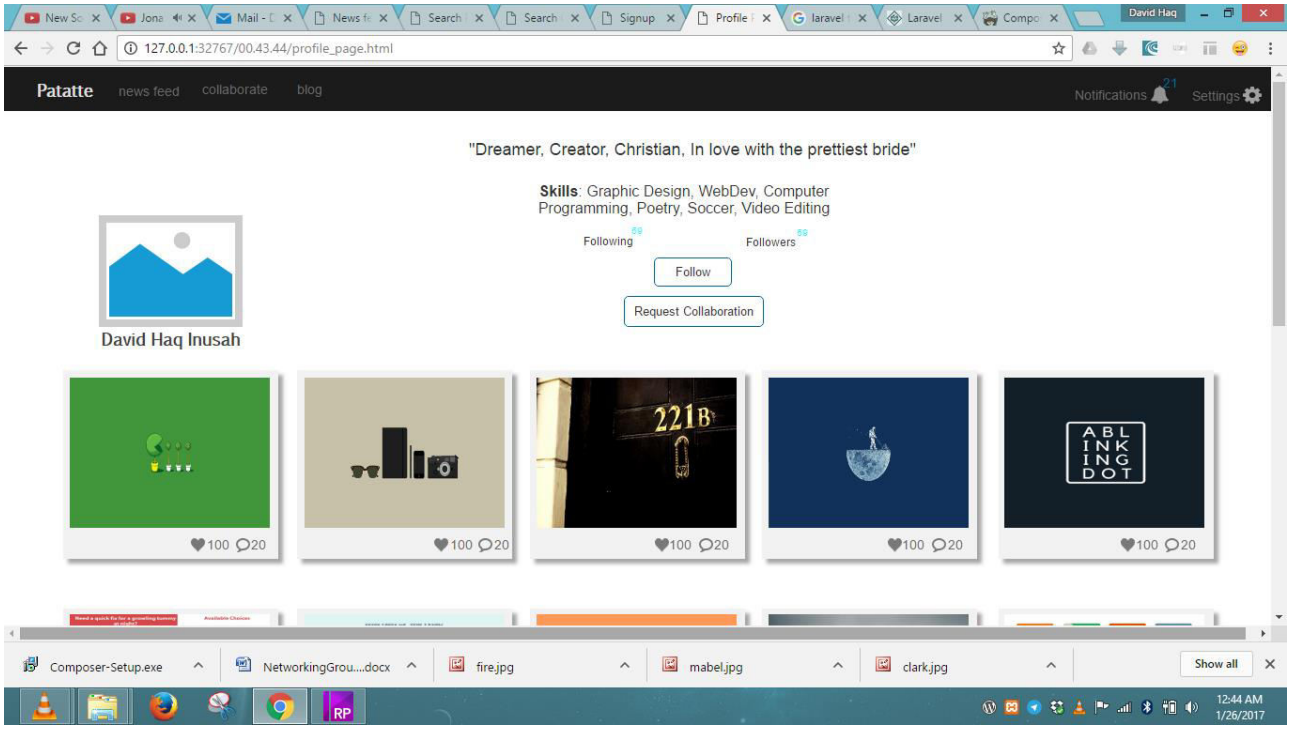

Figure 3.2: Profile Page

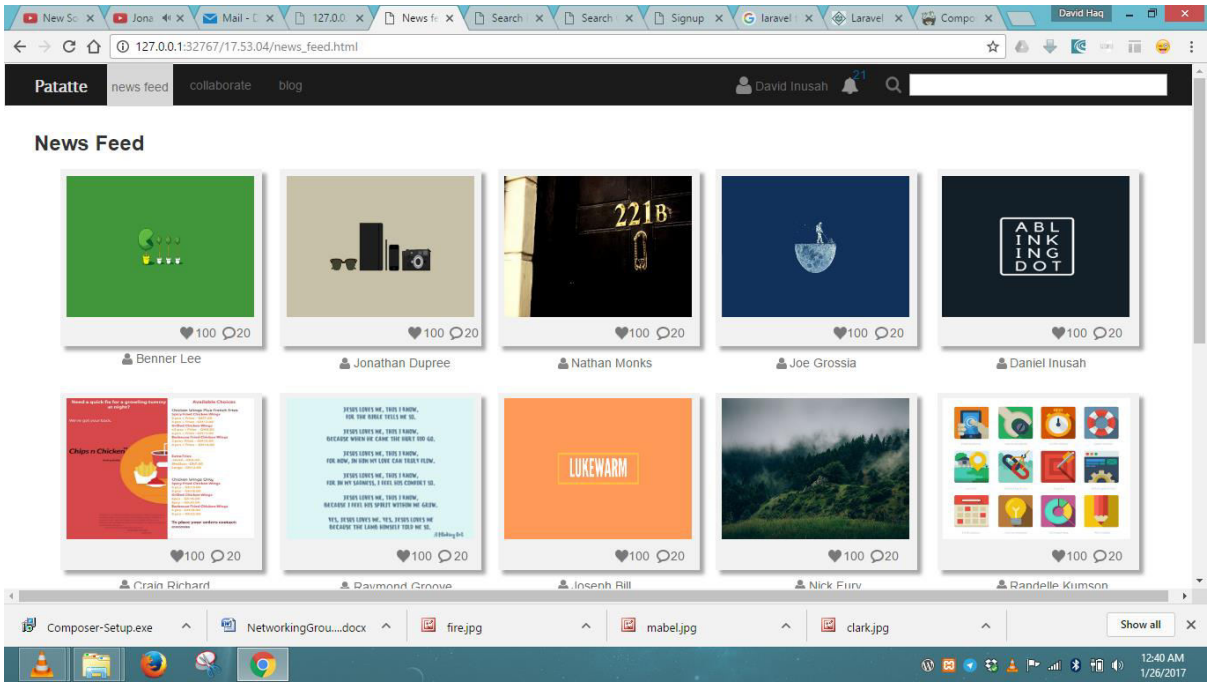

Figure 3.3: News Feed

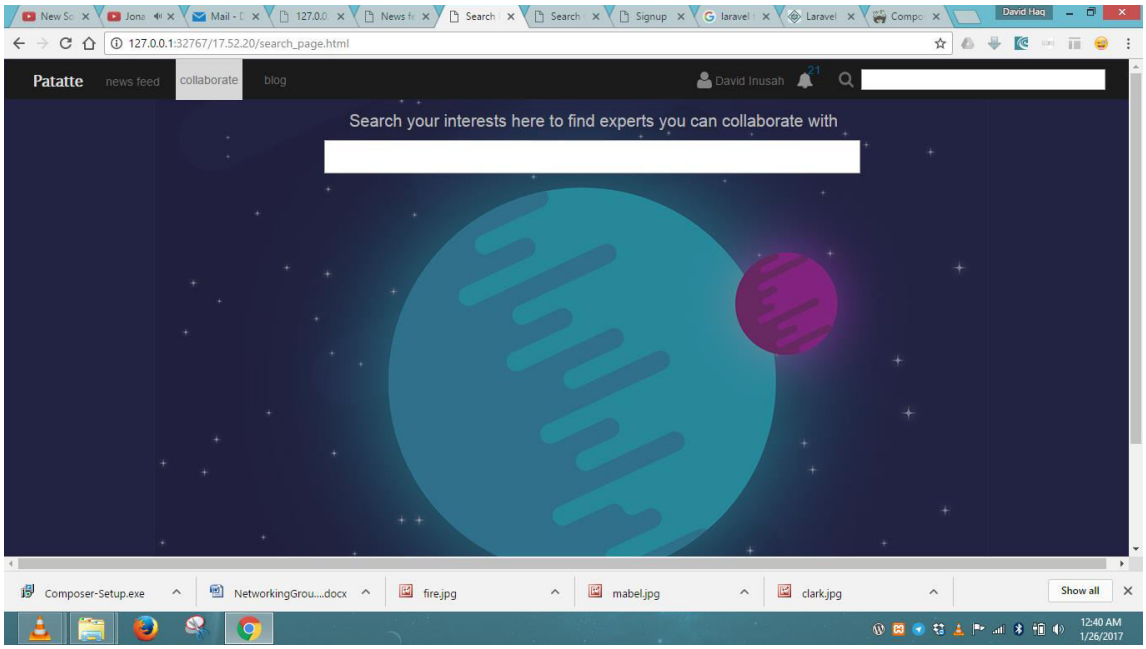

Figure 3.4: Search Page

3.1.2 Activity Diagram

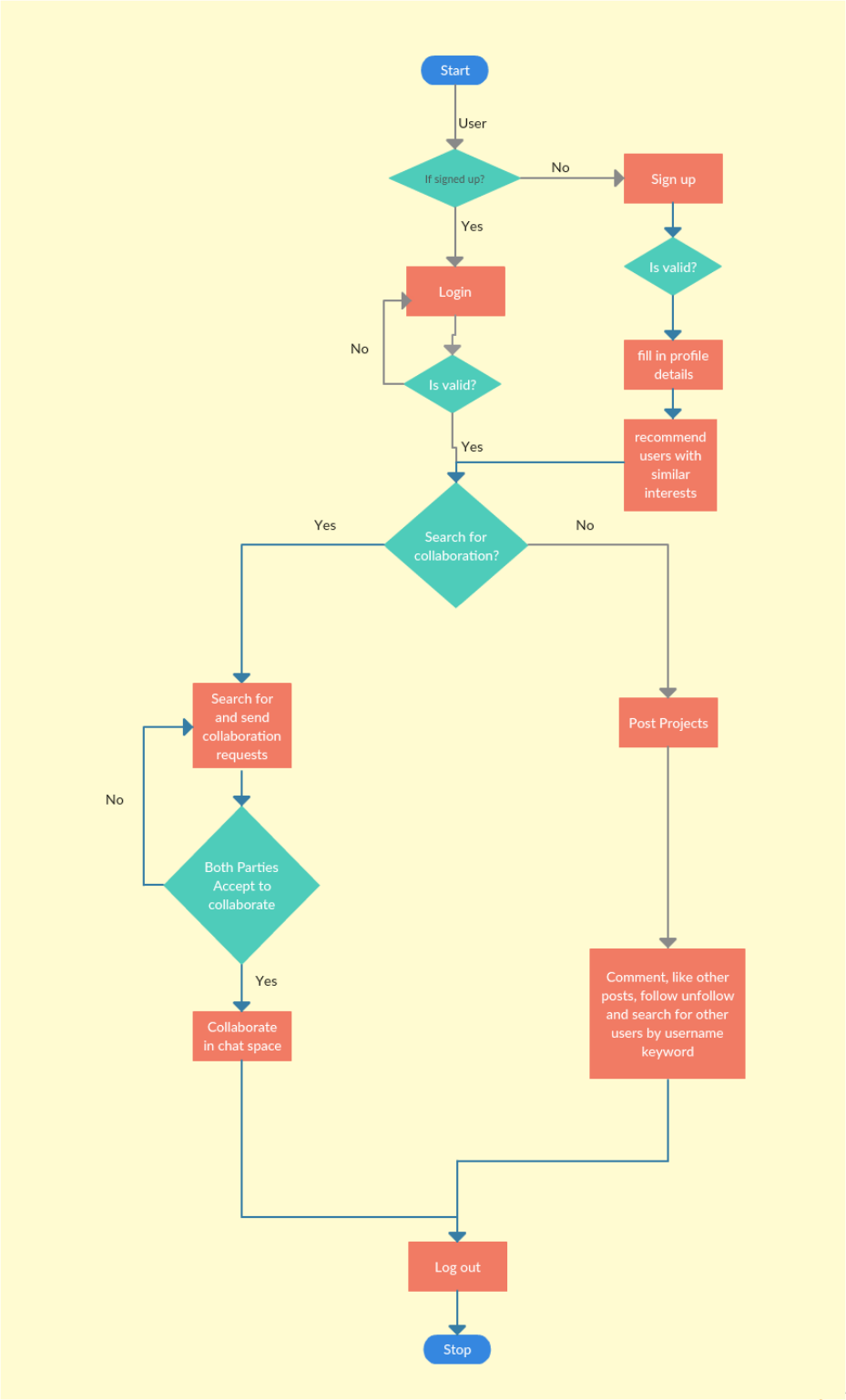

Figure 3.5: Activity Diagram

3.1.3 Class Diagram

In the implementation of the various functionalities, various static classes were used. They performed database queries, data returns and other tasks in the background to make to functionalities and flow of the system happen.

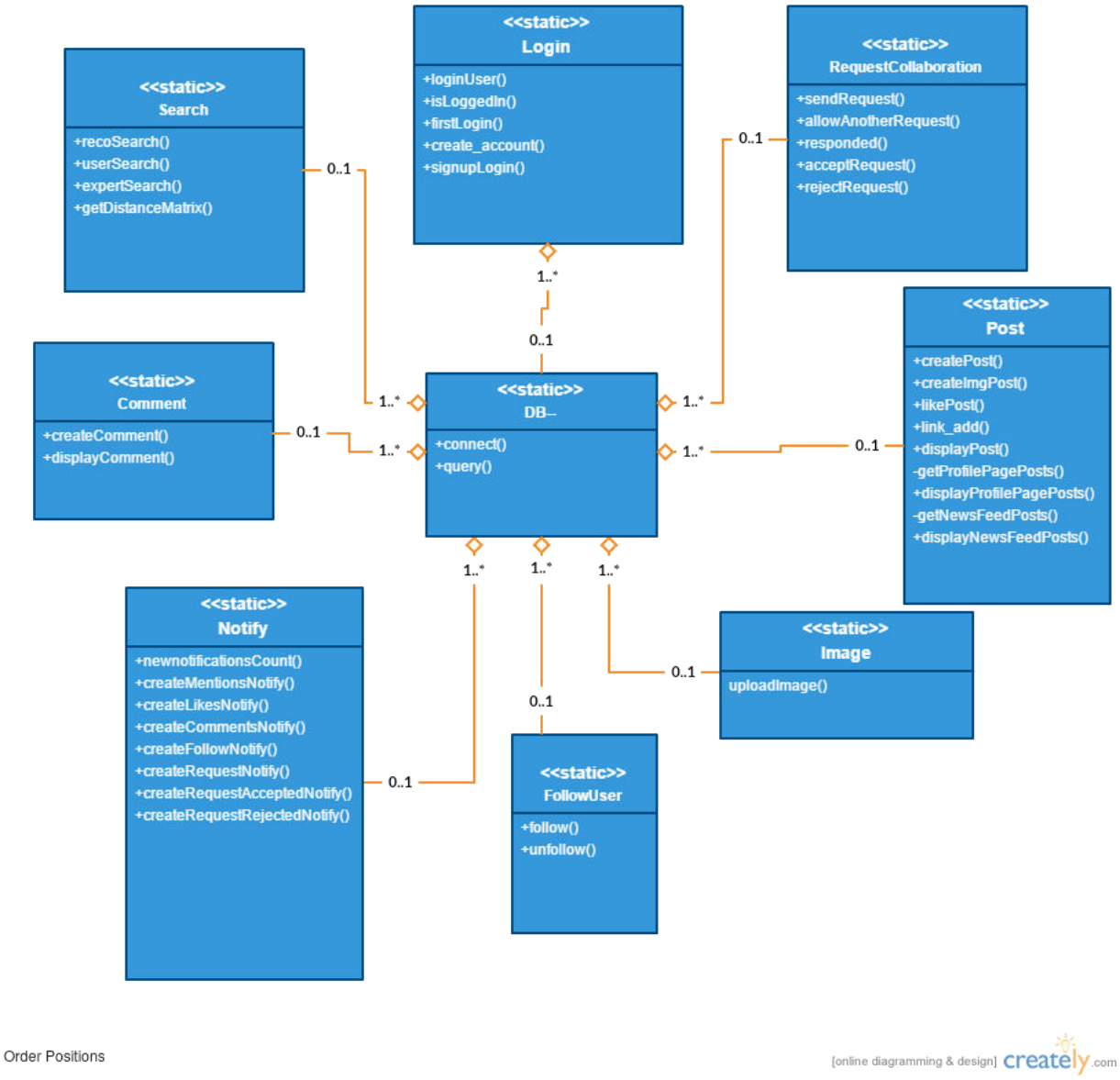

Figure 3.6: Class Diagram

3.1.4 Database Schema

The following schema represents the arrangement of tables in the database and the functional dependencies between the tables.

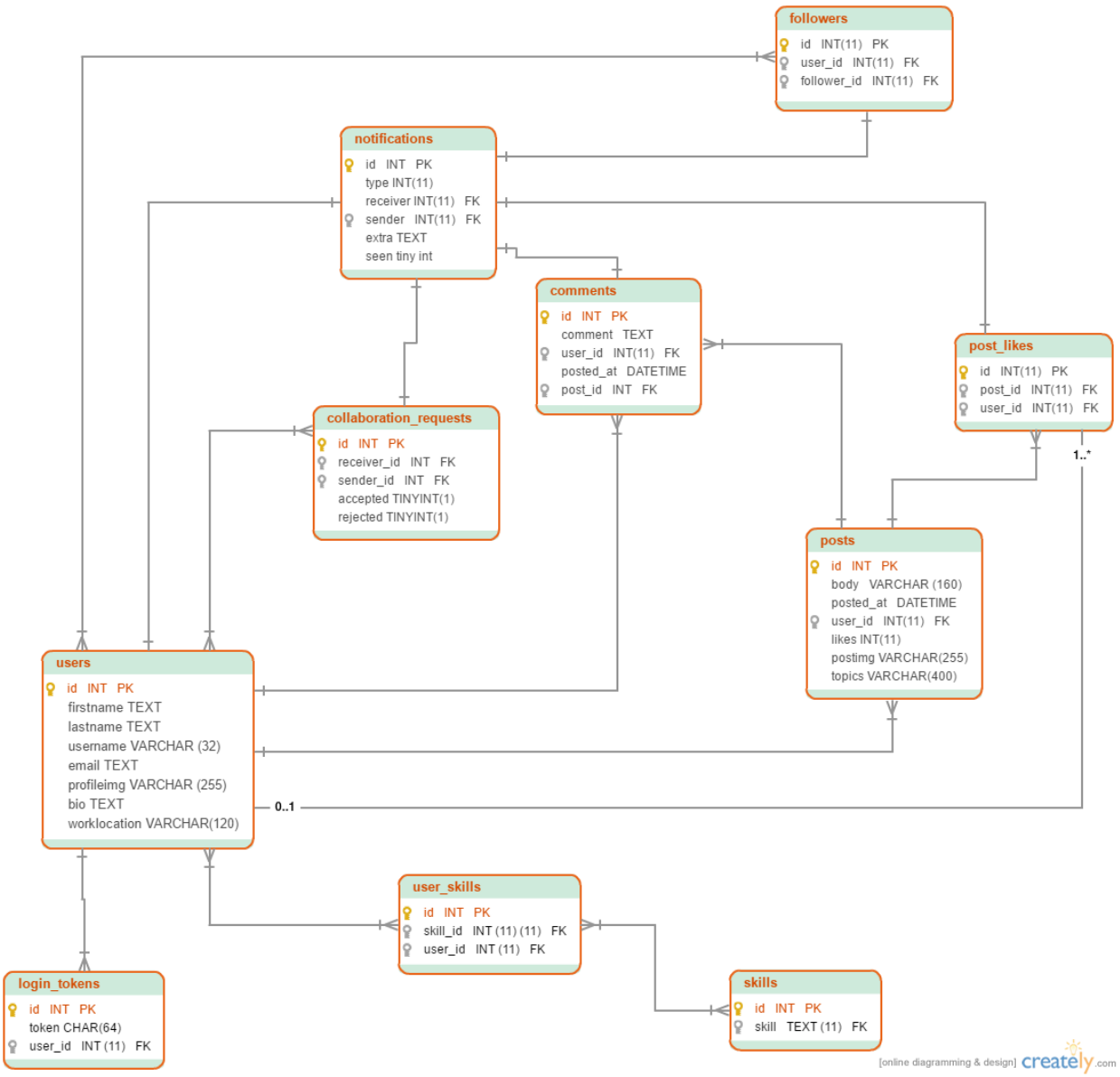

Figure 3.7: Database Schema

Chapter 4: Implementation

This chapter sets out in detail the project's tools of implementation, constituent modules and project implementation techniques.

4.1 Implementation Methodology

4.1.1 Tools

The system has been prototyped iteratively until a satisfying iteration is found. The wire framing tool used is Axure. The system is built in PHP 7.0 and data is stored to a MYSQL database. The user interface of the project is written in HTML5 and CSS3. The CSS is adopted from Materialize's CSS Framework. Some of the backend functionality was implemented by following a social network building tutorial series on a YouTube channel called 'howCode'. The system was built locally on an APACHE web server.

4.1.2 Development Model

The project was developed on a prototyping model. This model allows for the prototype of the system to be first built an incrementally tested and rebuilt till a final iteration emerges.

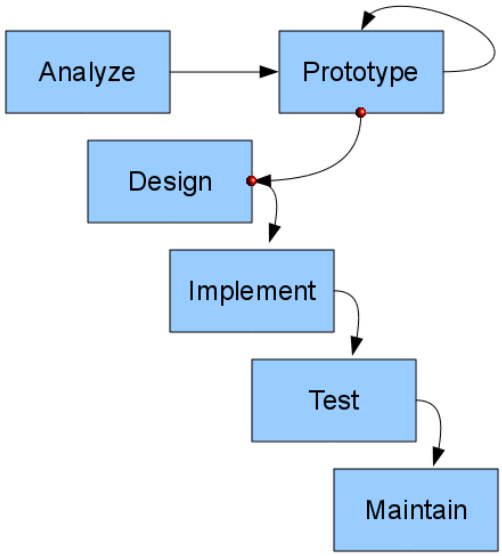

Figure 4.1: Prototype Model flow

4.2 Project Modules

4.2.1 Class component implementation and interaction

DB: This class contains the:

- connect() method that connects the rest of the application to the database.
- query() method that queries the database.

Login : This class contains the:

- loginUser() function that logs in a valid user. This function achieves the user login functional requirement. It returns an error message that will be an empty string if the login action was successful and will return a relevant error message if the login was not successful.
- isLoggedIn() function that returns a user's id if the user is logged in or false if the user is not logged in. This helps ensure system security by ensuring that a user that wants to access the system is always authorized.

- firstLogin() function that determines whether a user is logged in for the first time or not. This function helps the system to decide whether to display the profile_info page (where new users fill in more account details) or not.
- Create_account() function that create a new user. Achieves the sign-up functional requirement. It returns an error message that will be an empty string if the sign-up action was successful and will return a relevant error message if the sign-up was not successful.
- signupLogin() function that lets the user login right after they signup. Redirects the page to the profile_info page where users fill in profile details. This prevents the user from having to login in as an extra step.

Notify: Implements the following functions that achieves the notification system functional requirement:

- newnotificationsCount() function that returns a count of all notifications of a given user.
- createMentionsNotify() function that notifies a user of a mention by another user in their post.
- createLikesNotify() function that notifies a user of a new like on one of their posts.
- createFollowNotify() function that notifies a user of a new follow on their account.
- createRequestNotify() function that notifies a user that they have received a collaboration request.
- createRequestAcceptedNotify() function that notifies a user that their collaboration request has been accepted.

- createRequestRejectedNotify() function that notifies a user that their collaboration request has been rejected.

FollowUser: This class implements the following functions that achieves the 'follow' and 'unfollow' functional requirements:

- Follow() function that links a user to another user and his/her posts
- Unfollow() function that unlinks a user from another user and his/her posts

Image: This class implements the following functions that achieve the image posting functional requirement:

- uploadImage() function that uploads an image to 'Imgur' and updates the post table with the image link. Return a relevant error message to a particular fault if the upload was not successful or returns 'done' if it was successful.

Post: This class implements the following functions that achieves the posting, liking and deleting functional requirement:

- createPost() function that create an imageless post.
- createImgPost() function that uses the uploadImage function from the Image class to create a post that contains an image.
- deletePost() function that allows a user to delete a post that he/she posted.
- likePost() function that allows a user to like all posts.
- link_add() function that allows a user to mention another user in a post by adding a link to their name when they type it in a username with an '@' sign. It is also used in the news feed to add profile page link to a username with an '@' sign.
- getProfilePagePosts() private function that posts published by a particular user.

- displayProfilePagePosts() function that takes posts returned by getProfilePagePosts() and displays them.
- getNewsFeedPosts() function that gets all posts of the people a user is following and returns them.
- displayNewsFeedPosts() function that displays posts returns by getNewsFeedPosts().

RequestCollaboration: Implements the following functions that achieves the sending, accepting and rejection of collaboration request functional requirements:

- sendRequest() function that sends a collaboration request to a user whom a collaboration is desired from.
- acceptRequest() function that accepts a received collaboration request.
- rejectRequest() function that rejects a received collaboration request.

Search: Implements the following functions that achieves the recommendation search and the search by skill/interest keywords that returns a list of users and their proximity to the searching user functional requirements:

- recoSearch() function that searches for users with similar interests and returns them as recommendations to a new user that has not followed any users yet.
- getDistanceMatrix() function that sends a curl request of the searching user's work shed location and the found expert's work shed location to the Google Maps Distance Matrix API. The function returns the distance and duration elements of the json formatted string returned by the Google's Distance Matrix API.

Comment: Implements the comment post and comment display functionality.

- Comment() function that posts a comment to a particular post
- displayComments() function that displays all comments on a post.

4.2.2 External Integrated Components:

This section explains the integration process of the three APIs used in this project. They are **Imgur**: an online image hosting and sharing community, **SendBird**: a cloud based team collaboration tool and **Google Distance Matrix API** all of which connect to external applications by the OAuth 2.0 authentication framework.

According to Mitchell Anicas a Former Senior Technical Writer and current Software Engineer at Digital Ocean, "OAuth 2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as Facebook, GitHub, and DigitalOcean. It works by delegating user authentication to the service that hosts the user account, and authorizing third-party applications to access the user account. OAuth 2 provides authorization flows for web and desktop applications, and mobile devices" (Anicas, 2014).

This is a diagrammatic representation of how both APIs were integrated using this framework.

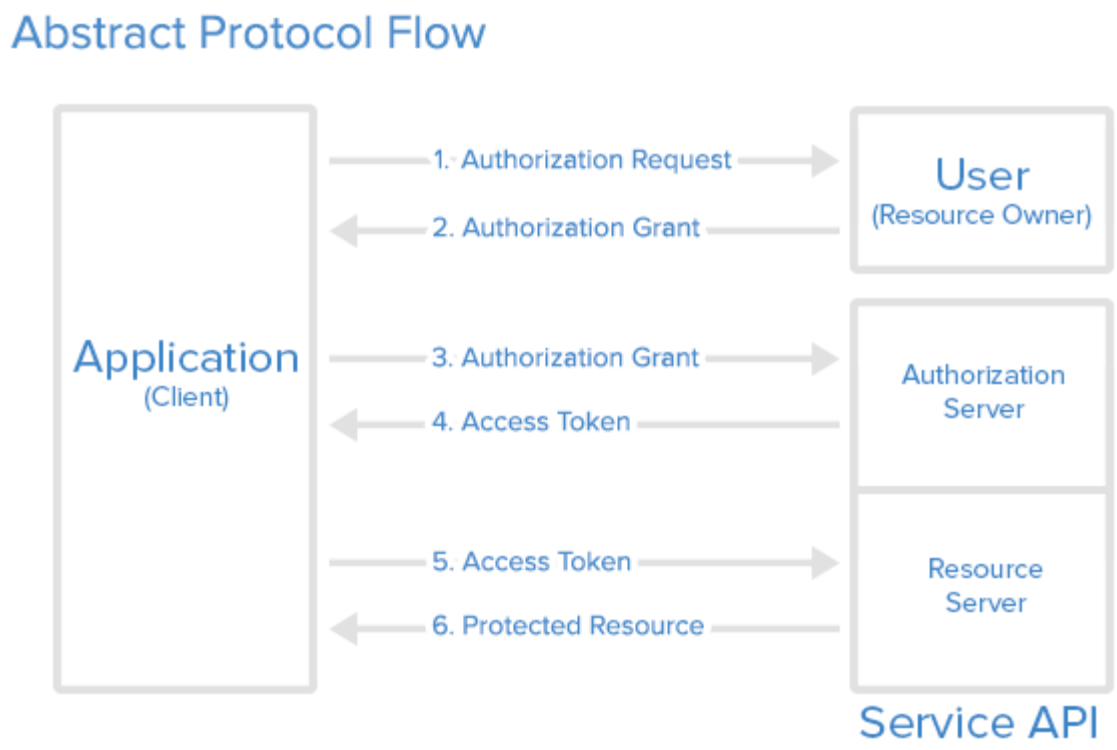

Figure 4.2: O-Auth 2.0 flow

4.2.2.1 SendBird API

SendBird was integrated into the project to allow users that agree to collaborate to share multimedia and interact privately. The project only allows users that agree to collaborate to have access to this feature. To do this I registered my application with SendBird.com and was provided a client secret, id and OAuth Tokens for every authorization request. If the tokens are recognized for a request made, an access token is a passed back to the Web app and the app sends that token to access the chat resources of SendBird.

4.2.2.2 Imgur API

This is an image hosting service that is integrated into the project to allow users to share images and gifs of their projects on the network. To do this I created an account on Imgur and registered my application on it. The registered application was provided a client ID, a client_secret and an access token and a refresh token. The uploadImage function in the Image class connects to the Imgur API and uploads the file to my imgur account. The link of the uploaded file is returned and updated into my postimg column in my posts table in the database.

4.2.2.3 GoogleMaps Distance Matrix API

This is a Google Maps API from Google that provides distance and travel duration information for any Google Maps known locations provided as 'origin' and 'destination'. To do this, my code sends out a 'Curl' request of a formatted URL to the Google Distance Matrix API with the origin and destination parameters of the searching user and the found user. The URL has a unique key attached to it that was generated when I registered the web app with the Google API.

4.3 Project Implementation Screenshots

Below are screenshots of the user interface of the actual project implementation.

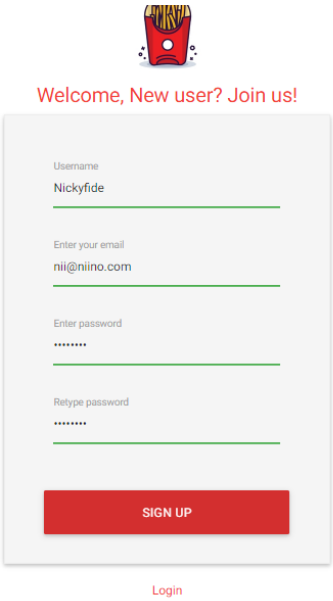

Figure 4.3: Sign-up Page

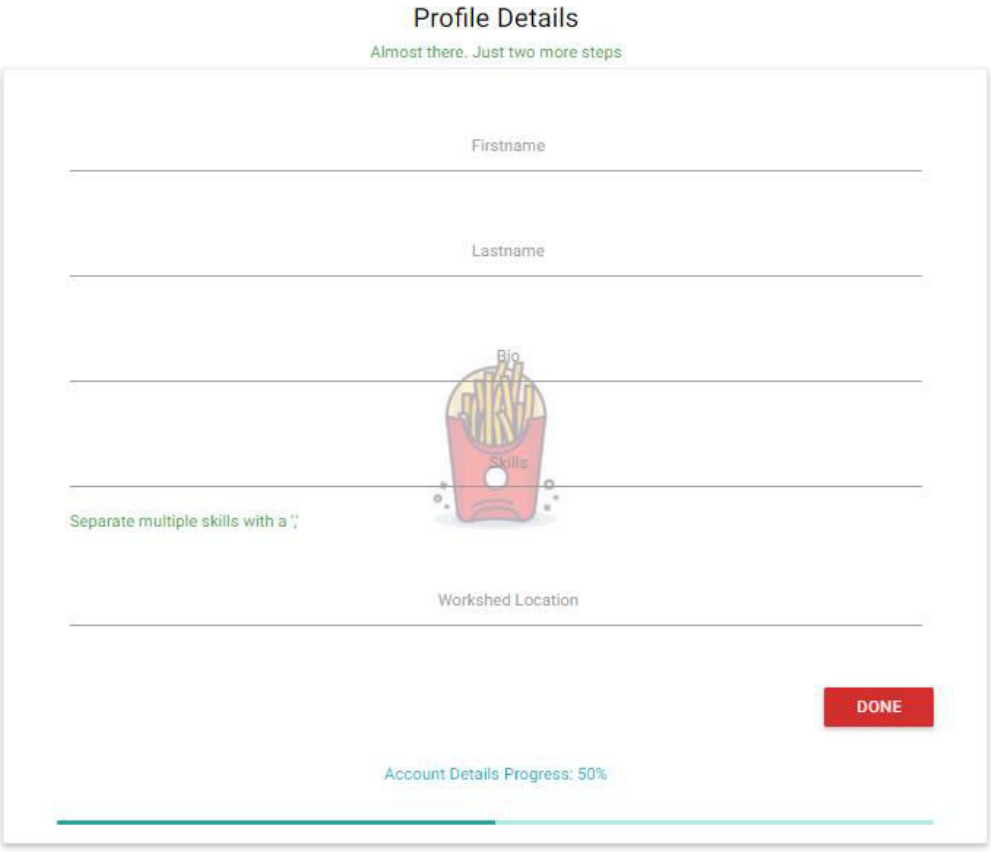

Figure 4.4: Profile Details Page

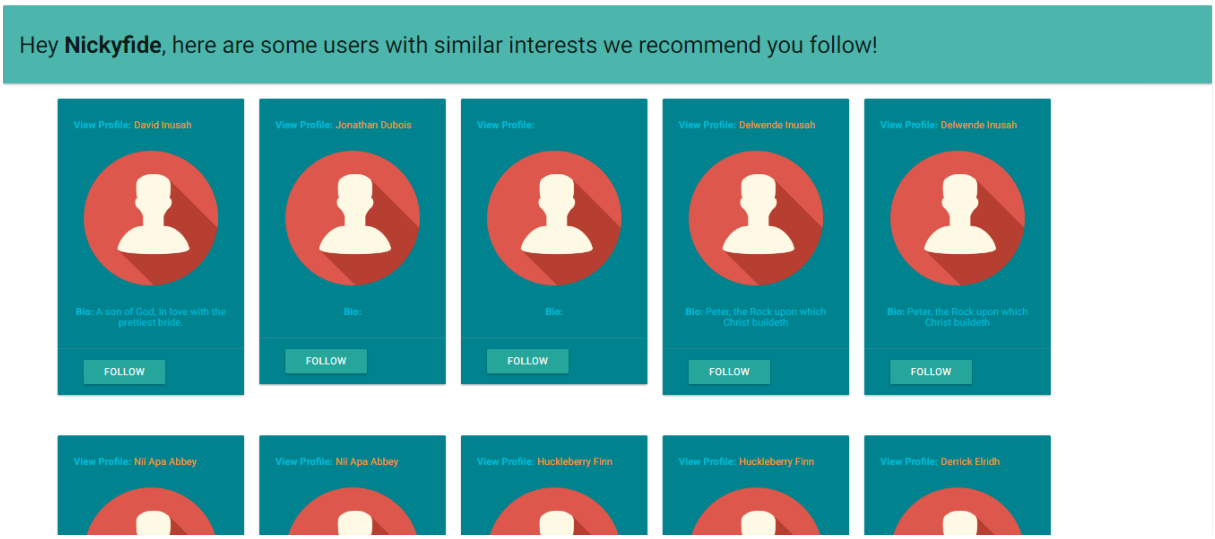

Figure 4.5: Recommended Users Page

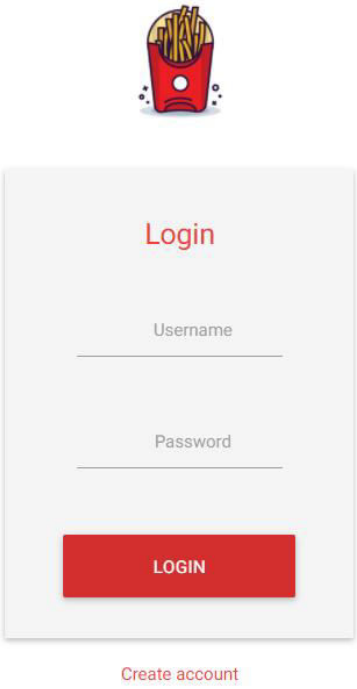

Figure 4.6: Login Page

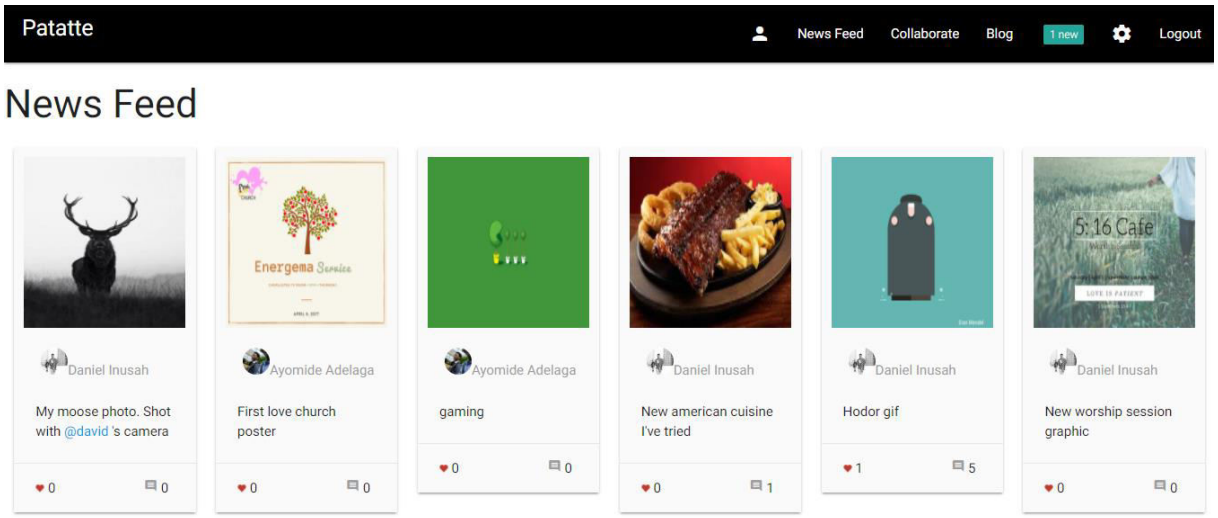

Figure 4.7: News Feed Page

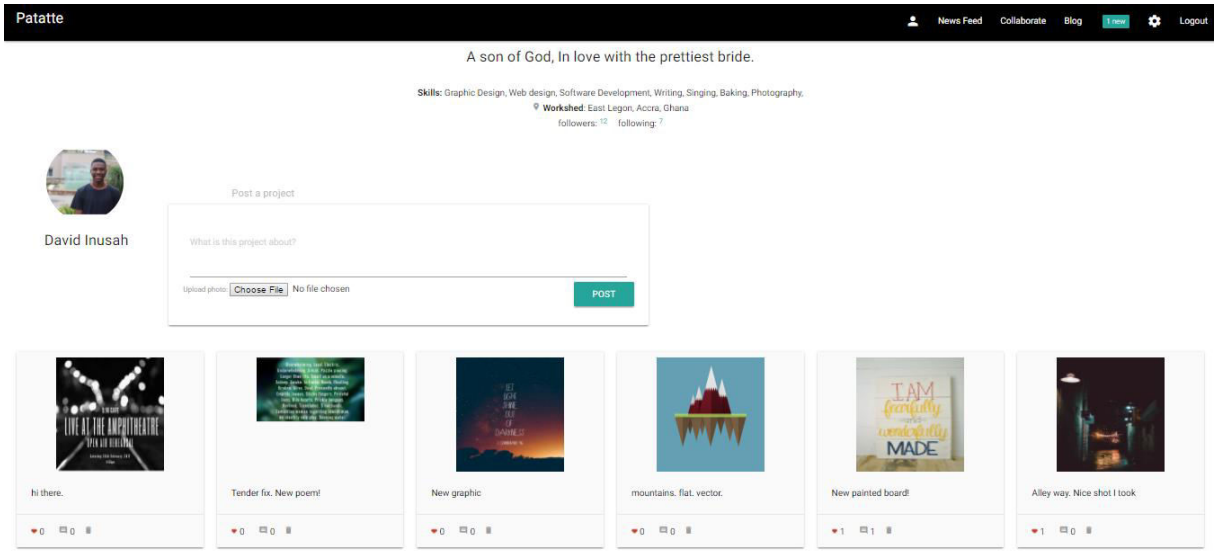

Figure 4.8: Profile Page

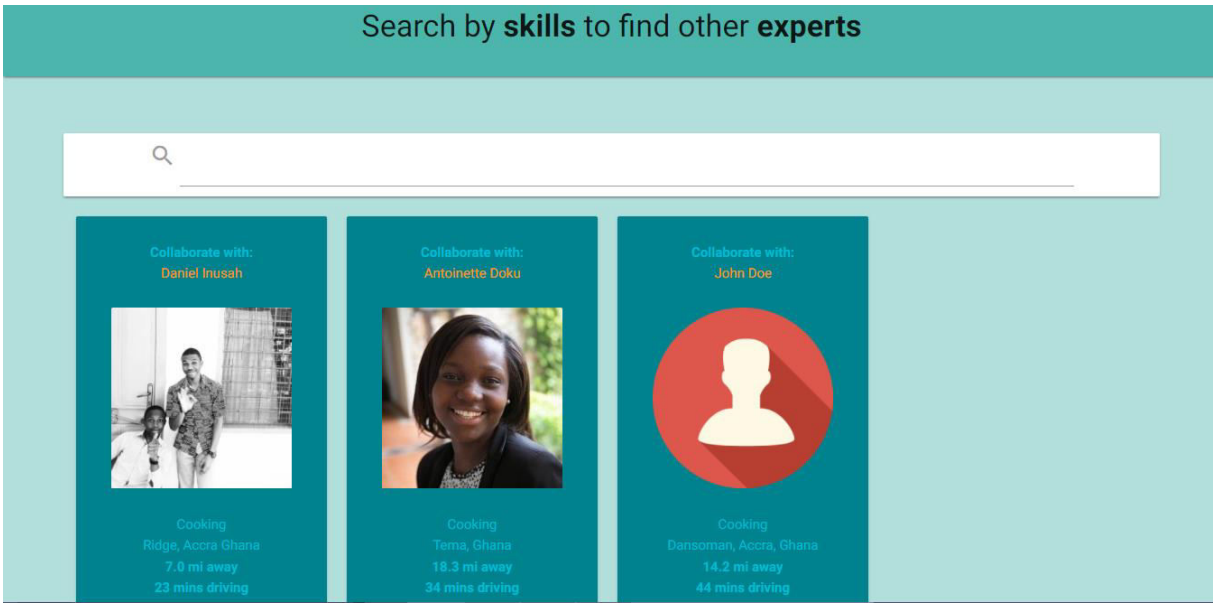

Figure 4.9: Search Page with Results for 'Cooking' Search

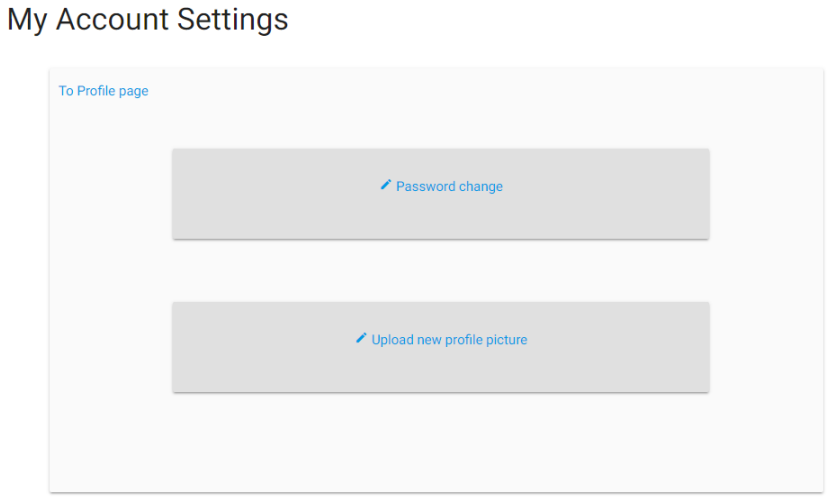

Figure 4.9: Settings Page

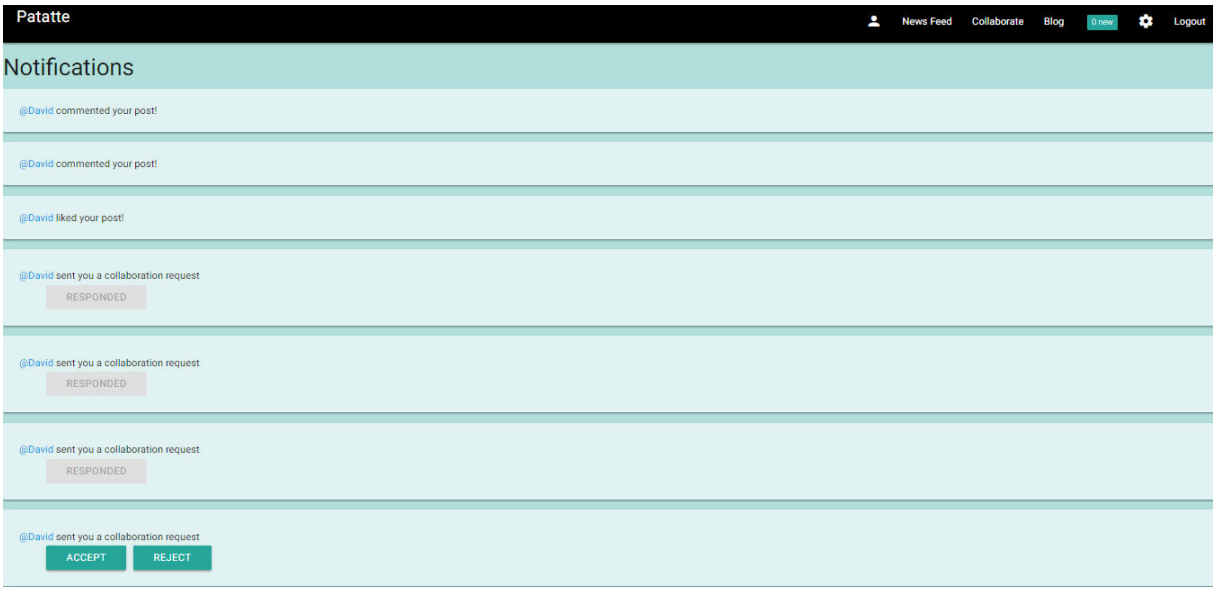

Figure 4.9: Settings Page

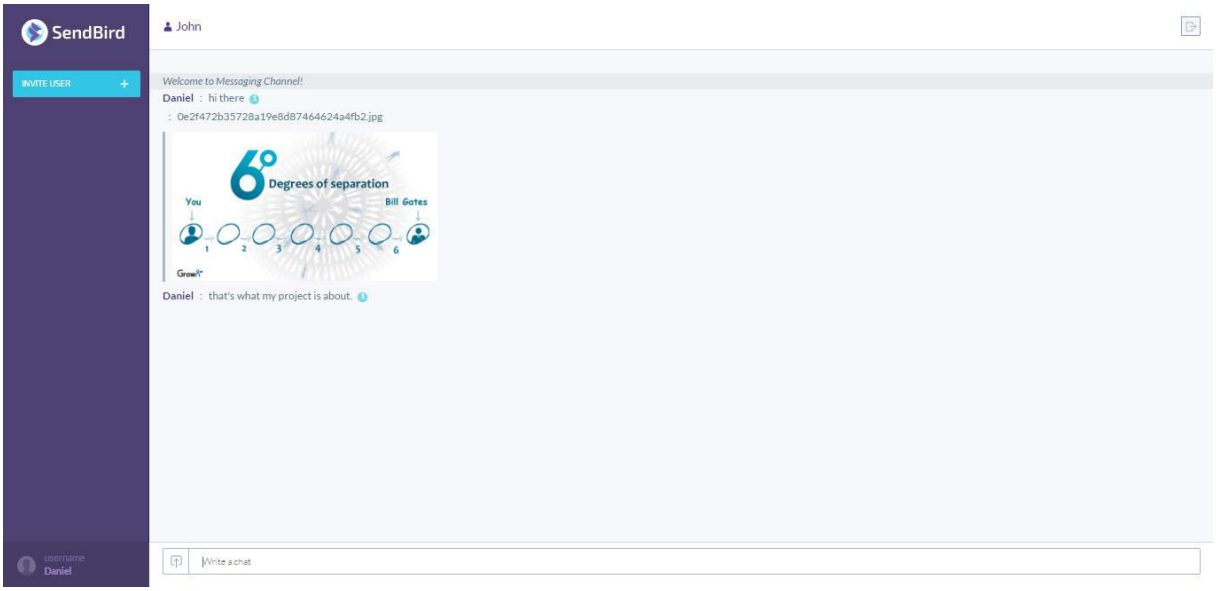

Figure 4.10: Chat Page

Chapter 5: Testing and Results

This chapter details the tests conducted on the project system to that it works in the ways it is expected to work. The testing plan followed, test classes and cases used and the test case results generated will be outlined and reported in this chapter.

5.1 Test Plan

This section outlines a blueprint for tests conducted on the system to identify faults and errors. It includes a Unit Test Plan, an Integration Test Plan, System Test Plan and Acceptance Test Plan.

5.1.1 Unit Test Plan

The various functions of every class will be tested using the black box testing method using the Equivalence Class Technique of selecting test cases.

Table 5.1: Unit test plan

Class to test	Unit to be tested	Test case classes	Description
DB	Connect() Query()	Database select queries Database count queries Database update queries Database insert queries	These queries test the connection to the database through the connect() function and tests the query() the function by querying the database for results or responses and returning those responses.
Login	loginUser() create_account() isLoggedIn() firstLogin() signupLogin()	1. Logging in with wrong passwords and wrong usernames. 2. Logging in with an authorized usernames and passwords. 3. Signing up with non-unique usernames 4. Signing up with valid usernames, passwords	These tests verify the Login class' implementation of the login and signup functional requirements.

		and email addresses. 5. Return id of the logged in users. 6. Check if a logged in user is logging in for the first time.	
Notify	newnotificationsCount() createMentionsNotify() createLikesNotify() createFollowNotify() createRequestNotify() createRequestAcceptedNotify() createRequestRejectedNotify()	1. Valid user ids and expected parameters.	These tests verify the Notify class' implementation of the notifications functional requirement.
FollowUser	Follow() Unfollow()	1. Invalid user ids. 2. valid user ids and expected parameters.	These tests verify the FollowUser class' implementation of the follow and unfollow user functional requirements.
Image	uploadImage()	1. Uploading images larger than 10MB. 2. Uploading images with right size and accurate parameters	These tests verify the Image class' implementation of the image post upload functionality.
Post	createPost() createImgPost() deletePost() likePost() link_add() getProfilePagePosts() displayProfilePagePosts() getNewsFeedPosts() displayNewsFeedPosts() getNewsFeedPosts()	1. Invalid user ids. 2. valid user ids and expected parameters.	These tests verify the Post class' implementation of the post creation, liking, deleting and displaying functional requirements.
RequestCollaboration	sendRequest() acceptRequest() rejectRequest()	1. Invalid user ids. 2. valid user ids and expected parameters.	These tests verify the RequestCollaboration class' implementation of the requesting, accepting and rejecting collaboration

			requests functional requirements.
Search	recoSearch() userSearch() expertSearch() getDistanceMatrix()	1. Invalid user ids or usernames. 2. Invalid locations. 3. valid user ids, usernames and locations and expected parameters.	These tests verify the Search class' implementation of the user recommendation, username keyword search and skill keyword search functional requirements.

5.1.2 Integration Test Plan

Integration testing will be carried out on the various PHP pages that utilize the class functions to archive the functional requirements of the project. The black box testing approach will be used. The type of black box testing technique that will be used is the State-based testing approach as the changes in state of the system will be observed against the input provided.

Table 5.2: Integration test plan

Pages	Functionality to test	Test case/state changes	Description
login.php	login functionality	Logging in and being directed to the news feed/index.php page	Tests the login page component interactions to login only registered users.
create-account.php	Sign-up functionality	Signing up and being directed to the account details/profile_info.php page	Tests the component interaction of the create-account page to register and setup user accounts.
profile_info.php	Adding account details functionality	Adding account details and being redirected to recommended_users.php or the user	Tests the component interaction on the profile_info page to update user profile information.

		recommendations pages.	
Recommended_users.php	Recommending users with similar interests to newly signed up users.	Display found users within the system with new users' skills.	Tests the component interaction on the recommended_users page to recommend similar interest/skill users to new users
Index.php	displaying posts of all followed users	Displaying posts	Tests the component interaction on the index page to display posts of all followed users.
profile.php	Central page for displaying users' project portfolios and account information.	Display user information and portfolio	Tests the component interaction on the profile page to display a user's profile information and project portfolio.
Collaboration_search.php	Finds other users for purposes of collaboration by searching with skill keywords.	Display users with skills being searched for	Tests the component interaction on the collaboration_search page to find and display users for collaboration.
notify.php	Displays all new notifications of a user User can accept or reject all collaboration request notifications.	Notify users of interactions with account on the social network Accept and redirect to sendbird chat application or reject collaboration requests	Tests the component interaction on the notify page to display all user notifications
collaborate.php	Message and share media with collaborating users	Sendbird chat API for sharing media and messaging	Tests the component interaction on the collaborate page to send and receive media and messages

profileimgupload.php	Upload profile picture	Upload and display a profile image to a user account	Tests the component interaction on the profileimageupload page to upload a user's profile image to the user's profile page.
change-password.php	Changes a user's password	User password updated in database	Tests the component interaction on the change-password page to change a users password.
logout.php	Logs out a user	Logged out user. No access to system and account information and posts	Test the log page functionality.

5.1.3 System Test Plan

The whole system will be tested by reviewing the use cases of the user and comparing to see if all of them have been achieved. This will be done by following the expected flow of the system diagrammatically represented in earlier chapters. This testing phase verifies that the project has achieved the functional requirements of the system.

Table 5.3: System test plan

Test case	Work flow to test	Description
Registered user login	1. Login	Test the login functionality for existing users.
Unregistered user sign-up	1. Sign-up 2. Profile details 3. Recommended users	Tests the account setting up flow for new users
Logged in user system usage	1. Posting an image and text project. 2. Liking posts.	Tests the various functionalities of the system available to the user once their accounts have

	3. Commenting on posts. 4. Deleting owned posts. 5. following/unfollowing users. 6. Requesting collaborations. 7. Viewing followed users posted projects. 8. Accepting/rejecting collaboration requests. 9. Searching for users by username keywords. 10. Searching for users by skill keywords. 11. Changing user profile picture. 12. Changing user account password. 13. Messaging and sharing media with accepted collaborators.	been setup or they have logged in.
System log out by logged in users.	1. Log out	Tests whether user is able to log out of system.

5.1.4 Acceptance Test Plan

This last testing phase is meant to find out whether the project will be accepted by the end user or not. It validates the project software. For this phase, only Beta testing with real world end users will be carried out.

Table 5.4: Acceptance test plan

User profile	Feedback and insights
This section will profile users based on their gender, exposure to social network usage, technical knowledge of computers and user skills/interests/talents.	This section will draws insights from how the user goes about using the system with as little assistance as possible. Also document explicit user feedback

5.2 Test Results

This section provides a detailed description of all test case results.

5.2.1 Unit Testing

These are the actual test samples and results for unit testing.

Table 5.5: Unit testing results

Class and functions	Test case based on test plan equivalence classes	Expected result	Observed result
DB			
`Connect()`	`DB::connect()`	Successfully connects to database.	Successfully connects to database.
`query($query, $params=array())`	`DB::query('SELECT * FROM notifications WHERE receiver=:userid ORDER BY id DESC',' array(':userid'=>2)')`	Returns rows of selected queries.	Returns rows of selected queries.
	`DB::query('Select count(*) from users')`	Returns count result	Returns count result.

	DB::query('Update users set username=:username where username='David'',' array(':username'=>'Danielson')') DB::query('Insert into posts values (\'\', :postbody, NOW(), :userid, 0, \'\', \'\')', array(':postbody'=>'hi there', ':userid'=>2));	Returns true Returns true	Returns true Returns true

Login
 	
loginUser($username,$password)
	Login::loginUser('non-existent user', 'wrong password')
	
	//existing user with correct password.
	Login::loginUser('David', 'rootroot')
	
	//creating account with an existing username
	Login::create_account('David',happyboy,'happyboy','mail@mail.com')
		echo Invalid user
		
		log user in and directs user to index.php page
		
		echo username already in use.
			echoes invalid user
			
			logs user in and directs user to index.php page
			
			echo username already in use.

create_account($username,$password1,$password2,$email)	//creating an account with a non-existing username Login::create_account('Joe233,happyboy,'happyboy','mail@mail.com')	Create new user account for Joe233	Creates new user account for Joe233

isLoggedIn()
 	
firstLogin($userid)
	Login::isLoggedIn()
	
	//userid of new user Joe233
	Login::firstLogin(15)
	
	//userid of David, an
		Returns id of logged in user
		
		Returns true
			Returns id of logged in user
			
			Returns true

signupLogin($username, $password)	existing user Login::firstLogin(2) Login:: signupLogin ('Joe233','happyboy')	Returns false Logs Joe233 in and directs him to profile_info.php page.	Returns false Logs Joe233 in and directs him to profile_info.php page.

FollowUser
 	
Follow($userid, $followerid)
	//valid userid,followerid
	FollowUser::follow(15,2)
	
	//invalid userid,followerid
	FollowUser::follow(0,-1)
		Echo 'user followed'
		
		Echo 'unable to follow user'
			Echo 'user followed'
			
			Echo 'unable to follow user'

Unfollow($userid, $followerid)	//valid userid,followerid
	FollowUser::unfollow(15,2)
	
	//invalid userid,followerid
	FollowUser::unfollow(0,-1)
		Echo 'user unfollowed'
		
		Echo 'unable to unfollow user'
			Echo 'user unfollowed'
			
			Echo 'unable to unfollow user'

Image uploadImage($formname, $query, $params)	Image::uploadImage('postimg', "UPDATE posts SET postimg=:postimg WHERE id=:postid", array(':postid'=>1))	Upload the image to Imgur and updates the post table with the link returned from Imgur	Uploads the image to Imgur and updates the post table with the link returned from Imgur

Post
 	
createPost($postbody, $loggedInUserId, $profileUserId)
	//logged in userid equals profileid
	Post::createPost('new job', 15, 15);
	
	//logged in userid does not equal profileid
	Post::createPost('new job', 2, 15);
		Insert a new post record into the posts table
		
		Echo 'incorrect user'
			Inserts a new post record into the posts table
			
			Echoes 'incorrect user'

createImgPost($post body, $loggedInUserId, $profileUserId)	Post::createImgPost(' new job', Login::15, 15);	Insert new post into post table with an imgur link to image	Inserts new post into post table with an imgur link to image.
deletePost($postid, $followerid)	//valid postid and userid Post::deletePost(1, 15)	Return true	Returns true
	//invalid posted and userid Post::deletePost(0, -1)	Return false	Returns false
likePost($postId, $likerId)	Post::likePost(1, 2)	Update 'like' record in likes table.	Update 'like' record in likes table.
link_add($text)	Post::link_add('another job @David')	Return text with link to David's profile on @David	Returns text with link to David's profile on @David
getProfilePagePosts ($userid)	getProfilePagePosts(15)	Fetch all user posts from database	Fetches all user posts from database

displayProfilePagePosts($userid, $username, $loggedInUserId)	//valid parameters
	displayProfilePagePosts(15, 'Joe233',15)
		Return a view of all user posts.	Returns a view of all user posts.

	//invalid parameters
	displayProfilePagePosts(0, '43dsd',-1)
		Return an empty view.	Returns an empty view.

getNewsFeedPosts($userid)	getNewsFeedPosts(15)	Fetch all followed users' posts from database	Fetches all followed users' posts from database

displayNewsFeedPosts($username, $loggedInUserId)	//valid parameters
	displayNewsFeedPosts('Joe233',15)
		Return a view of all followed users' posts.	Returns a view of all followed users' posts.

	//invalid parameters		

	displayProfilePagePosts(0, '43dsd',-1)	Return an empty view.	Returns an empty view.

RequestCollaboration
sendRequest($userid, $followerid)
	//valid sender and receiver ids
	RequestCollaboration::sendRequest(2, 15)
	
	//invalid sender and receiver ids
	RequestCollaboration::sendRequest(-1, -2)
		Insert new request record into collaboration_requests table
		
		Echo 'unable to send request'
			Inserts new request record into collaboration_requests table
			
			Echoes 'unable to send request'

acceptRequest($requestid)	//for an existing request of id=2 RequestCollaboration::acceptRequest(2)	Update 'accepted' column from 0 to 1 in collaboration_request table where id=2	Updates 'accepted' column from 0 to 1 in collaboration_request table where id=2
rejectRequest($requestid)	//for an existing request of id=2 RequestCollaboration::rejectRequest(2)	Update 'rejected' column from 0 to 1 in collaboration_request table where id=2	Updates 'rejected' column from 0 to 1 in collaboration_request table where id=2
Notify newnotificationsCount($userid)	Notify::newnotificationsCount(2)	Return the number of all notifications where 'seen'=0	Returns the number of all notifications where 'seen'=0
createMentionsNotify(text = "")	Notify::createMentionsNotify('new job @David')	Insert a new notifications table record of type 1.	Inserts a new notifications table record of type 1.

createLikesNotify($postid = 0)	Notify::createLikesNotify(1)	Insert a new notifications table record of type 2.	Inserts a new notifications table record of type 1.
createCommentsNotify($comment=0)	Notify::createCommentsNotify('nice work')	Insert a new notifications table record of type 3.	Inserts a new notifications table record of type 1.
createFollowNotify($followerid, $userid)	Notify::createFollowNotify(15,2)	Insert a new notifications table record of type 4.	Inserts a new notifications table record of type 4.
createRequestNotify($sender, $receiver,$requestid)	Notify::createRequestNotify('Joe233','David',1)	Insert a new notifications table record of type 5.	Inserts a new notifications table record of type 5.
createRequestAcceptedNotify($sender, $receiver)	Notify::createRequestAcceptedNotify('Joe233','David')	Insert a new notifications table record of type 6.	Inserts a new notifications table record of type 6.
createRequestRejectedNotify($sender, $receiver)	Notify::createRequestRejectedNotify('Joe233','David')	Insert a new notifications table record of type 7.	Inserts a new notifications table record of type 7.
Search recoSearch($username)	Search::recoSearch('Joe233')	Return an array of users that have similar skills with the provided user	Returns an array of users that have similar skills with the provided user
userSearch($tosearch)	//valid username Search::userSearch('David')	Return a user with the same or inputted username	Returns a user with the same or inputted username
	//invalid username Search::userSearch('s	Return an empty array	Returns an empty array

	as899’) //valid skill keyword and userid Search::expertSearch(‘design’,’15’)	Return all users with the specified skill keyword.	Returns all users with the specified skill keyword.
expertSearch($keyword,$userid)	//invalid skill keyword Search::expertSearch(‘sadadas’,’15’)	Return an empty array().	Returns an empty array().
getDistanceMatrix($expert, $origin, $destination)	//valid parameters Search::getDistanceMatrix(‘David’, ‘Tema, Ghana’, Accra, Ghana)	Return distance and duration json string between origin and destination.	Returns distance and duration json string between origin and destination.
	//invalid parameters Search::getDistanceMatrix(‘rerr, ‘4e66’, Accra, Ghana)	Return an empty json string	Returns an empty json string

5.2.2 Integration Testing

This section shows the test results for integration testing.

Table 5.6: Integration testing results

Integration page	Test case	Expected result	Observed result
login.php	Logging in	User should be logged in directed to index.php.	User is directed to index.php.
create-account.php	Signing up	User account should be created and user directed to profile_info.php	User account is created and user is directed to profile_info.php

profile_info.php	Adding user profile details.	User adds profile details which are stored in database and is redirected to recommended_users.php	User adds profile details successfully and is redirected to recommended_users.php
Recommended_users.php	Find and display users with new users skills.	If there are existing users with some or all of the new users skills they are displayed as recommended users to follow. User can view their profile and chose to follow them. On clicking done, they are directed to the profile page.	Users with similar skills are successfully displayed. User can follow them or view their profiles first before following them. User is directed to the profile page when they click done
Index.php	Display posts	Users should see only posts of users they are following and like and comment on posts displayed.	Users see posts of only users they are following and are able to like and comment on posts displayed..
profile.php	Display user information and portfolio	Users should see their profile info and their collection of shared posts. They should also be able to like, comment and delete their posts.	User profile information is displayed and their shared posts are displayed too. They are also be able to like, comment and delete their posts.
Collaboration_search.php	Display users with skills being	Users with skills similar to the skill keyword in the search	Users with skills similar to the skill keyword in the search

	searched for	should be returned, with a link to their profile page, their skill, the distance between the searching user and the found user and the duration it would take to drive to them.	are displayed with a link to the found users' profile page, desired skill, distance between the searching user and the found user and the duration it would take to drive to them.
notify.php	Notify users of new likes, comments, follows, collaboration and requests.	User should be able to see new notifications. They should also be able to accept or reject collaboration requests. Accepting or rejecting requests also should notify the requesting user.	Notifications are displayed on the page with an array to string conversion error. Collaboration requests when accepted / rejected update the response to the database and notify the requesting user.
my-account.php	Account settings page for uploading new profile picture or changing password.	User should be able to upload new profile picture or change password or both.	User is able to change password and upload profile picture. User is also able to return to profile page.
logout.php	User logged out.	User should be logged out with no access to system and account information and posts	User is be logged out with no access to system and account information and posts

5.2.3 System Testing

This section shows the test results for integration testing.

Table 7: System testing results

Test case	Expected workflow	Observed workflow
Registered user login	1. Login	2. A registered user is logged-in in one step.
Unregistered user sign-up	1. Sign-up 2. Profile details 3. Recommended users	1. An unregistered user's account is created in one step. 2. They are logged in automatically after this but have to provide profile details mandatorily in another step. 3. After which similar skilled users are recommended to them. User is able to finish this step with or without 'following' any of the recommended users.
Logged in user system usage	1. Posting an image and text project. 2. Liking posts. 3. Commenting on posts. 4. Deleting owned posts.	1. User is directed to the index.php/news feed page on complete system entry successfully. 2. User views displayed posts successfully.

	5. following/unfollowing users. 6. Requesting collaborations. 7. Viewing followed users posted projects. 8. Accepting/rejecting collaboration requests. 9. Searching for users by username keywords. 10. Searching for users by skill keywords. 11. Changing user profile picture. 12. Changing user account password. 13. Messaging and sharing media with accepted collaborators.	3. User likes posts successfully. 4. User comments on posts successfully. 5. User views their profile page successfully. 6. User posts a new project with an image successfully. 7. User deletes an earlier posted project successfully. 8. User searches for another user by typing in their username successfully. 9. User follows found user successfully. 10. User searches for another user by typing a desired skill. 11. User finds a collection of users and makes a choice about who to collaborate with. 12. User views selected user's profile and sends him/her a collaboration request. 13. Users views his/her

		notifications and see a new like, follow and collaboration request. 14. User accepts collaboration request. 15. Slack channel is open between both user to facilitate their collaboration. 16. User changed profile picture 17. User changes password.
System log out by logged in users.	1. Log out	1. User logs out in one step.

5.2.4 Acceptance Testing

This section shows the test results for integration testing.

Table 8: Acceptance testing results

User profile	Feedback
Male user. Technical computer knowledge. Programming skills.	This user generally found his way around the application without assistance. However he responded to the hoverable feature of post by clicking on the post. He expected the post to pop up and present more project detail. This feature is absent from the application. His feedback was that the application is simple and he felt he had navigated about it well for his first usage. He said it would also be great if the application showed a user where he is on the application by coloring

	the tabs in the navbar.
Male user. Technical computer knowledge. Photography skills.	This user was able to easily navigate through the system on his own. His feedback was that the system is simple to use and suites the target users. However, he said it would be great if the system had a credibility system that prevents users from having to sift through every returned user in the collaboration search.
Female user. Technical computer knowledge. Interested in volunteering.	This user was able to use the application intuitively. Her feedback was that the system should not only recommend users for new users but for existing users as well because that would continuously help people expand their connection base. She gave the system an intuitiveness rating of 4/5.
Male user. Basic computer knowledge. Basic social network usage knowledge.	This user found some difficulty using the system and asked for help using the web application twice. He said that very uneducated informal experts may not be able to use the system because of some of the web icons used. However, he gave the system an intuitiveness rating of 4/5.
Female user. Basic computer knowledge. Not very conversant with using social networks	This user managed to use the application well without much assistance. She asked for help understanding some of the icons used, but was able to figure out the rest of the system. She said she would like to use the application because it seemed like a convenient way to reach other experts she does not know personally.

5.3 Testing Phase Summary.

Except for the error on the notify.php page, the project passed all its tests, performing the expected functional requirements. The project was also accepted favorably by all 5 users that it was tested with. By these result, the project meets it requirements both functionally and non-functionally. Non-functionally it has achieved requirements of being simple and very intuitive to use. It has also been secure to use since passwords have been hashed, inputs have been sanitized.

Chapter 6: Conclusions and Recommendations

6.1 Conclusions

The project addresses the main problem of a difficulty in finding informal experts like carpenters, painters etc… and semi-formal experts like graphic designers, web developers etc. Tested users have also expressed an interest in the project because they believe it is a necessary tool that will help them find experts to collaborate with easily.

By leveraging the geographical defying nature of the internet, the project will connect numerous informal experts in a simple way that will allow them to collaborate.

6.2 Recommendations

6.2.1 Limitations

The project was not developed with any particular architecture which could leave loop holes for security exploitations. Also the project does not verify user accounts against their email addresses which could be a big security concern. Finally, the project has no way of assuring users of a certain level of quality of the experts it recommends.

6.2.2 Future Works

To boost security, Laravel will be used to re-implement the project. Current functionalities will be integrated into Laravel's MVC architecture protecting the model from the view through the controller's encapsulation.

Also, a user rating system will be introduced to help searching users decide on returned experts more quickly. This is to prevent them from having to sift through numerous returned profiles before they can choose to collaborate with a given user.

Finally, there will be recommendations of who to follow for all users and not just new users.

Appendix

Table A: Classes and the functional requirements they have implemented

Class	**Functional requirement implemented**
DB	Connects to the database and queries it.
Login	Logs in a current user, creates an account for a new user. Connects to the database through the DB class
Notify	Notifies user of mentions, likes, requests for collaboration and follows. Connects to the database through the DB class
FollowUser	Helps the user 'follow' or 'unfollow' another user's account. Connects to the database through the DB class. Also 'followed' users are notified through the Notify class
Image	Uploads an image to imgur through the imgur API and stores the link to the image in the database. Connects to the database through the DB class.
Post	Uploads posts to a user's profile and other user's newsfeeds. Uses the Image class to upload image posts. Also provides methods to like and display user posts. Connects to the database through the DB class.
RequestCollaboration	Deals with the sending and receiving, acceptance or rejection of collaboration requests between users. Uses the Notify class to notify users of received collaboration requests. Connects to the database using the DB class.
Search	Finds users with similar interests and recommends them to new users. Also finds users by searching with username keywords.
SearchRank	Returns a ranked list of users (based on their proximity from the user) based on searching with skill/interest keywords.

References

Anicas, Mitchell. "An Introduction To Oauth-2". *Digital Ocean*. N.p., 2017. Web. 17 Apr. 2017.

Facts, LinkedIn. "Topic: Linkedin". *www.statista.com*. N.p., 2017. Web. 16 Apr. 2017.

Milette, G., Schneider, M., Ryall, K., & Hyland, R. (2009). Exploiting social context for expertise propagation. *Proceedings Of The 32Nd International ACM SIGIR Conference On Research And Development In Information Retrieval - SIGIR '09*, 835-835. http://dx.doi.org/10.1145/1571941.1572155

Pal, A. (2015). Discovering Experts across Multiple Domains. *Proceedings Of The 38Th International ACM SIGIR Conference On Research And Development In Information Retrieval - SIGIR '15*, 923-926. http://dx.doi.org/10.1145/2766462.2767774

Smirnova, E. (2011). A model for expert finding in social networks. *Proceedings Of The 34Th International ACM SIGIR Conference On Research And Development In Information - SIGIR '11*, 1191-1192. http://dx.doi.org/10.1145/2009916.2010114

Zhang, J., Ackerman, M., Adamic, L., & Nam, K. (2007). QuME. *Proceedings Of The 20Th Annual ACM Symposium On User Interface Software And Technology - UIST '07*, 111-114. http://dx.doi.org/10.1145/1294211.1294230