

**Web Application Project Documentation**

**Table of Contents**

|  |  |
| --- | --- |
| **Title** | **Page** |
| Title Page | 1 |
| Table of Contents | 2 |
| Introduction | 3 |
| Implemented Functions and Use Cases | 4-11 |
| Phase 1: Implementation and Challenges | 4-6 |
| Phase 2: Implementation and Challenges | 7-9 |
| Phase 3: Implementation and Challenges | 9-11 |
| The Final Struggle: Implementations and Challenges | 11 |
| Web Application Comparisons | 12 |
| Novel Features | 13 |
| Limitation | 14 |
| Technologies Used | 15 |
| Improvements and Future Implementation | 16 |
| Conclusion | 17 |
| References | 18 |
| Appendix | 19 |

**Web Application Documentation**

**Abstract**: BabbleT is a of a social interactive web application that allows people to connect and meet new people with similar interests. Our aim is to make the world great again. New users can register for an account for free on the spot. After logging in, users can chat with a user with a desired topic anywhere and anytime. The application has customizable profile, changeable chat backgrounds and a private one on one chat. In addition, it is also mobile compatible. The application is divided into three major phases of development. The entire application is development from scratch without using frameworks.

**Introduction**

In today’s society, human are constantly busy with responsibilities, such as their career as well as families. It leaves us with almost no time to socialize with other people and make new friends. With such measly amount of free time we have, it is also impossible to meet people with similar interests. With this problem getting worse and worse, we wanted to create an app to remedy the situation. This led us to create one of the world’s leading chat application, BabbleT. BabbleT allows an individual to find other users with similar interests on the go. BabbleT is a easy to use, free of charge chat system that users can set up within seconds and start chatting with a user by simply entering a topic. The design concept is to make it as simple and intuitive as possible, while maintaining a beautiful and elegant style that is pleasing to the eyes. The system will add the user into a pool, and will immediately search for any other users that also submitted the same topic, and will continue to search for more matches until the user has selected a target to talk to. Once selected, a chat box will appear for both users and they can begin chatting. The user may also edit their own profile page and check out the profile page of the ones they’re talking to. The application allows users to connect with new people every day, anytime and everywhere in the world. In addition, we have optimized this to use as little data as possible. This way, they can fulfill their responsibilities, while meeting new friends without any conflict in schedules. We hope that this will make a huge impact and make the world great again.

The web application will be free of charge. However users are required to have an account to be able to use it. New users can register for a new account straight on the homepage. Existing users can just simply login to start using the application. After logging in, users will be required to enter a topic of choice (i.e. sports) of what they would like to chat about. The application will then enter the user into a pool search for and list all the users that match the submitted topic. If there are no users matched, then it will continue to search. It will continue to search even after matches show up until the user selects another user to talk to. The user is able to choose any user from the list below to chat with. If a user has selected another user, a chat box interface will open up below connect the two users. The user will be able to enter messages to each other. The user can click on the name displayed in chat to view the profile page. They can also edit their own profile page by pressing the button on the top right side of the bar.

**Implemented Functions and Use Cases**

Our web application is developed based on modular design. The server-side of the application is completely modular. Each unique function that deals with the database has its own separate PHP file. If one function doesn’t work or if we need to improve the code, we can easily change or replace the code without changing other code or files. Our development is can be divided into three phases.

**Phase 1: Implementation**

The first phase of our web application is setting up the database and to develop a login, register, and logout system. The three major functions in phase 1 is login, logout, register. We used HTML forms to get user input. These functions are written in JavaScript and jQuery along with AJAX. The login function requires a username and a password. This function will pass that information to the server-side where we have a php file for dealing with login. The PHP file will send a SQL query to the database and check the username with password. If the login fields are not valid, a function will notify the user about the error and if the login information does not match, the user will also be notified. If the information is correct, the user will be redirected to the homepage where they can start chatting. When a user login a session variable of this status will be saved. Therefore, if a user tries to go to the homepage without logging in, they will be redirected to the denied page.

The register function requires a username, password, and a retype password field. These data will be passed by to the server-side where a PHP file will handle it. The PHP file will send a SQL query to the database and insert the new record with the data. Furthermore, another SQL query will be sent to the database to create a default profile for the new user. If a user picked an already existing username in the database or if the fields are not valid or if the two passwords fields do not match, the user will be notified about the problem.

When a user wants to logout, they can click the logout button located on the navigation bar. The session variable of this status will be terminated. The user will be redirected to the login page.

**Phase 1: Challenges**

The first challenge of our journey to success was getting a server that would allow us to host a chat system. We had used a free service from freesubdomain.org to host our project, but they shut us down multiple times and banned us in the end without any warning simply because they disallow hosting of a chat system. We easily lost 2 weeks of work, and we had to search far and wide to find a reliable service that would fit our requirements. We finally found a relatively cheap hosting server named godaddy and we decided to use it.

We also faced some technical difficulties on creating the user\_exists function. We created a functionuser\_exists to check whether or not the user already exists while registering.

**Phase 1: Use Cases:**

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Register | |
| **Actors:** | | User, application, Backend DB |
| **Description:** | | Allows the user to Register in the application |
| **Trigger:** | | The user launches/uses the application for the first time |
| **Preconditions:** | | The user is not already registered in the system  Another user is not already logged in |
| **Post conditions:** | | The user creates an account in the system  User info is saved in the backend DB of the system |
| **Normal Flow:** | | 1. User clicks “Register New User” 2. User fills out the necessary information required to create a new account in the system. 3. User submits form with information 4. System saves the information in the back-end DB 5. User account is created |
| **Alternative Flows:** | | Cancel |
| **Exceptions:** | | 1. Username exists in the system 2. Invalid password 3. Invalid username 4. non-secure password 5. Any of the other mandatory fields from the user register form is not valid |

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Login | |
| **Actors:** | | User, application |
| **Description:** | | Allows the user to log into the application |
| **Trigger:** | | The user opens the application, or logs out. |
| **Preconditions:** | | There is no user logged in |
| **Post conditions:** | | The user is logged into the system. |
| **Normal Flow:** | | 1. The user inputs both a user ID and the associated password. 2. The system approves of the user ID and password and logs the user into the system. |
| **Alternative Flows:** | |  |
| **Exceptions:** | | 1. The user inputs an incorrect user ID and password combination 2. The system rejects the user from logging into the system. |

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Logout | |
| **Actors:** | | User, application |
| **Description:** | | Allows a user to logout of the application |
| **Trigger:** | | User clicks logout button |
| **Preconditions:** | | 1. User must have an active and registered account  2. User must be logged in already |
| **Post conditions:** | | 1. User is logged out of their account |
| **Normal Flow:** | | 1. Application logout button is clicked  2. Application logs out the user  3. Application goes into query to mark account as logged off  4. User is redirected to login page |
| **Alternative Flows:** | |  |
| **Exceptions:** | | 1. User did not click logout button 2. Technical difficulties on server 3. User is disconnected from internet |

**Phase 2: Implementation**

The second phase of our web application is to develop the matching system. The application will find all the users that match the same topic and displays it. This function is implemented by using a form to get the desired topic from the user. A jQuery function used with AJAX is then used to send the topic to the server-side where there is a php file that handles the input. The php will send a SQL query to get a list of users that have searched the same topic. The list is returned to the client-side in a table.

In addition, when a user has searched a topic, two functions implemented with jQuery will run in the background continuously for every fixed time interval. One function is used to check if any user has selected you. The other function is used to update the list of matched users. This will keep finding new users and adds it on to the existing list.

User can select any of the users listed. When the user selects a user from the list, the application will automatically create the chat for both users. A jQuery function used with AJAX is used to send the selected user to the server-side. A PHP file handles this input and sends a SQL to the database to update the other user’s record. With the continuous function, the other user will be able to know that they have been selected. When a user has been selected, the search field and the list of users will be hidden to prevent the user from picking another user or searching for another topic. As a reminder, this is a one on one chat not a multi-user chat room. Only one chat can exist at a time therefore, a user is not allowed to search while a chat is active.

If a user want to search another topic after a search without selecting a user, their previous record of the topic in the database will be updated to the new topic. Users can only search one topic at a time.

**Phase 2: Challenges**

We have faced multiple challenges throughout this phase, for example the race conditions in matching users together. Race condition happens when two process are both trying to access and modify the same resource at the same time. This causes one process to lose what it was originally going to write. We had to make sure that the tables are updated according to the sequence of events. One case of this would be if our requirements of creating a chat would be fulfilled only when both users click the select button. Because it is almost impossible for us to figure which action happened first, we combatted the problem by changing the requirement to only one user selecting the chat. Read more on this subject in the limitation section.

**Phase 2: Use Case**

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Match Topic | |
| **Actors:** | | User, Application, Backend DB |
| **Description:** | | Allows users to be matched with other. |
| **Trigger:** | | The user launches/uses the chat set up. |
| **Preconditions:** | | User type in their topic in the search box. |
| **Post conditions:** | | Application prints out people that have the same criteria as the user. |
| **Normal Flow:** | | 1. User type in the search box. 2. User click submit/search. 3. The application looks in the backend SQL server for a match. 4. The application prints out the back the list of the users that are matched with the same topic and the button to select them. |
| **Alternative Flows:** | | Cancel |
| **Exceptions:** | | 1. Search length too long. 2. No users have the same search criteria. 3. Server not working 4. Users are disconnected from the internet. |

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Select Match | |
| **Actors:** | | User, Application, Backend DB |
| **Description:** | | Allows users to select the person they want to chat with. |
| **Trigger:** | | The users are matched with other. |
| **Preconditions:** | | * The users are matched with each other. * The users are logged in. |
| **Post conditions:** | | User enters chat with selected person |
| **Normal Flow:** | | 1. The user click on the select match button. 2. System creates a new record in the chatTable of the SQL database. 3. User enters into the chat interface with the matched user. |
| **Alternative Flows:** | | Cancel |
| **Exceptions:** | | 1. Server not working 2. Users are disconnected from the internet. 3. None of the users click the select button. |

**Phase 3: Implementation**

The third phase of our web application is to develop the chat system. It includes the message sending, receiving functions and also the quit chat function. When a user selects a user, the application will display the chat box, the quit button and the send message field to the user.

Two continuous functions run in the background at a set time interval implemented by jQuery, used with AJAX. One function will check constantly, if the other user has sent any new unread messages. New messages will be appended to the chat box and once a message has been read, the record of the message is updated with the status. The other function will check if the other user has quit the chat. If the other user has quit, the chat box, message field and quit button will be hidden and the search button will be displayed again. User can then search for another topic.

User can send a message by filling in the input box and the user can then utilize the submit button in order to send the message. The field is implemented by a HTML form. The data is sent to a PHP file in the server-side by a jQuery function used with AJAX. The PHP file will use a SQL query to insert a new record with the message along with the user’s ID. Also, the user’s own message is appended onto the chat box.

The user can quit the chat by clicking the quit button above the chat box. The quit function is implemented with jQuery and AJAX. It will execute SQL in a PHP file in the server-side. The SQL query will update the status of the chat. The chat box, message field and quit button will be hidden and the search box will appear again.

**Phase 3: Challenges**

This phase was by far the most challenging phase in our project. The first problem we faced was that our application would print out the same “last message” multiple times. To combat this problem, we included a status (being either 0 or 1) with each message. Each new message will start with a tag ‘status’ 0 to indicate that it is unread. As the other user checks for a new message, if we find a message with the status 0, we will print it, then change the status to 1 to indicate that it has already been printed(so we won’t print it again during the next search).

We encountered many different problems with getting both user\_id and print them out in the chat box. We solved this by using two different printing function, gettingMessage.php and sendMessage.php, each will print out messages for different users.

**Phase 3: Use Cases:**

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Send message | |
| **Actors:** | | User, Application, Backend DB |
| **Description:** | | Allows the users to talk to each other. |
| **Trigger:** | | The user launches/uses the chat set up. |
| **Preconditions:** | | * The users are matched with each other. * The users are logged in. |
| **Post conditions:** | | User manages to send the message. |
| **Normal Flow:** | | 1. User click on the send button 2. Message is send through to the SQL server. 3. Updates the database with the new message. |
| **Alternative Flows:** | | Cancel |
| **Exceptions:** | | 1. Message length too long. 2. User not connected to the internet. 3. SQL server under maintenance. 4. One of the users quit the chat. 5. The input is null/empty. |

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Receive Message | |
| **Actors:** | | Users, Application, Backend DB |
| **Description:** | | Allows the users to receive the message. |
| **Trigger:** | | The users are on the chat interface. |
| **Preconditions:** | | * The users are matched with each other. * The users are logged in. * The user is on the chat interface. |
| **Post conditions:** | | * Users manage to receive the chat from the SQL database. * The message is displayed to the users. |
| **Normal Flow:** | | 1. The application checks the server for new message 2. If there is a new message based upon on the chatID. 3. The application checks if the user’s current chatID matches with the new message’s chatID. 4. If they are the same, append it on the display interface. |
| **Alternative Flows:** | | Cancel |
| **Exceptions:** | | 1. The chatID are not the same. 2. There are no new messages. 3. The message’s length is way too long. |

**The Final Struggle: Implementation and Challenges**

The last few functions remaining from the three major phases are developing the profiles and styling the web application. When a user register a default profile is made for them automatically. If a user wants to change their profile page, they will have to be logged in. In the navigation bar, they can click on “Profile” button. User will be redirected to the user’s profile page. The user can then click on “Edit Profile” button which will redirect them to edit profile page.

On the edit profile page, users are able edit the name, about, phone, email and profile picture link. To change the profile picture, users are required to provide a valid picture URL in the field. Users are required to have all the fields filled up because the fields that are left empty will be saved as well. Therefore, a user will have to edit all the fields before saving the changes. jQuery used with AJAX will send the data to a PHP file in the server-side. The PHP file will execute a SQL query to update the database with the information. It was a struggle creating the profile, as we also have to update the chat box with the profile picture from the profile.html page. We were able to solve this by implementing more SQL select lines to the profile table with the individual printing function with the user’s unique userid.

Lastly, the styling on the application is all done by using CSS3 and Bootstrap. In addition, to make our web application mobile friendly and responsive, Bootstrap is used to implement that. The functions and implementation mentioned above is only the major functionality in the application. There are many more minor functions that made what our application to be.

**Web Application Comparisons**

The following are some similar web applications. The pros are the elements that we liked about the application and what we plan to implement it into our own project. The cons are the elements that we disliked about the application and will change or improve in our project.

**Omegle/Chatroulette:**

Pros: The application allows user to connect people with similar interests.

Cons: The application does not have detailed user profiles.

**Facebook/MySpace:**

Pros: The application is able to save chat and has user profiles.

Cons: The application cannot search users by interests.

**Tinder:**

Pros: The application can do distance search and has user profiles

Cons: The application has limited interest search (male vs female).

We decided to choose and combine the Pros of the above. As we are also users of apps above, we feel that the apps above are all missing some sort of feature to make it perfect. Our concept is a chat system which will have both search by interest and user profiles. We believe this combination is the perfect blend of what it takes to create the most successful chat system.

**Novel Features**

We noticed a lack of features in many of the social media but could be found in other places. We envision a product that will combine many of the works currently out there to create the ultimate social media platform.

Users are able to create the profile. When a user first registers, the application will automatically create a default profile page. User can then login to edit their profile. Profile page contains a profile picture, an about field, a phone number field, a name field and an email field. User can edit any of the fields. To change the profile picture, the user has to provide a link to an image. Users have the freedom of writing anything in the fields.

In addition, users are able to view each other’s profiles from the chat. When a user clicks on the link on the other user’s username in the chat box a new web page will popup. This web page will display the other user’s profile. Users can also see each other’s profile picture in the chat as a small icon beside the username.

Furthermore, users are able to change the background color of the chat to their liking. They can choose from yellow, grey, black, azure, baby blue, teal, green and blue. They are able to change their preferences anytime while chat is active.

The chat messages are contained in a colored bubble simulating texting on a phone. User’s own message will be contained in bubble with a green gradient background. The other user’s message will be contained in a bubble with a blue gradient background.

The application is also mobile compatible, all the elements on each of the web page will be automatically resized to properly display on the device. It can be viewed on any screen resolutions. It can also be view on any popular devices including iPhone, tablets, iPads, iMac, desktops, Android, laptops and etc.

In addition, the chat is a one on one and it is also private. Furthermore, our application also provides a FAQ page where most of the user’s problems are answered. It contains how-to questions with simple detailed tutorial answers.

**Limitation**

There are limitations in the web application. Here are some currently known limitations:

* Username and password cannot contain space, periods, or any other special characters.
* Here is a scenario of a race condition: When User A click match with User B and right before the application hides the table from User B, User B picks someone else. The problem will become User A picks User B and User B picked someone else that was not User A instead of not being able to pick.
* Another limitation is when User A click match with User B and User B click match with User A at the same time then they will both update their own request with userid = 0 and status = 1. Status 1 means the user has already matched with another user so other users cannot select him. This will create a problem in the application.
* History is not kept after the user quit the chat. In other words, if a user is matched with the same user as last time then there will not be any chat history of previous messages. If user refreshes the page or closes the tab and reopens while the chat is proceeding, the current chat with the user will be deleted. The user can no longer chat with the other user. Users are required to click quit button to quit chat and cannot refresh or close the page while chat is active.
* The current functionality of the web application can only allow one on one chat conversation.
* There is a limit on the number users that are chatting at the same time due to the limit on the server connections. It has unstable server connections.
* There is a limit on the number of character a user can send in a message.
* There is a limit on the number of character used in password, username, and in the search field.
* User can only be chatting with only one user. While user has chat active, they cannot search for another topic. User has to quit the current chat to search for a another topic.
* If User A selects User B,User B cannot select another user and cannot decline the match. The application will automatically create the chat for both User A and User B. If User B does not want to chat with User A, the only way to decline is to click on the quit button to quit the chat. User B can then search for same or different topic to match or select another user.
* If a user chooses to edit their profile, they will have to fill in every field of the profile. Otherwise, if there user leave some fields empty then after they click save changes, the empty fields will empty out the current data. User can click the profile button to go back to profile page without saving changes after they filled in the fields.
* The application does not have session management. If the user leaves the window for a long time, the application will not automatically logout the user.

**Technologies Used**

The technologies and techniques that we have used in our web application are:

* HTML
* CSS3
* JavaScript
* jQuery
* AJAX
* PHP
* Bootstrap
* MySQL

The server-side of the web application uses only MySQL and PHP for database transactions. On the client-side of the web application uses HTML, JavaScript, jQuery, AJAX, CSS3 and Bootstrap. JavaScript and jQuery is used for the functionality of the application. jQuery is also used with AJAX to communicate with the server-side to execute transactions. We used CSS3 and Bootstrap to make our application mobile friendly and responsive. In addition, it also used for the styling the application.

We choose to use AJAX because it is essential to a chatting system. We do not want the web page to be refreshed after every new message has been received. In addition, we made use of jQuery because it makes the code of AJAX call much simpler and can be easily used compared to the JavaScript syntax. jQuery is used with AJAX in all of our communications that had to deal with the database,

Our team had more background experience with PHP than any other server-side programming languages such as ASP, JSP, Ruby, Perl, etc. Therefore we choose to use PHP in our server-side. MySQL is often used with PHP so we used MySQL to implement all the database transactions.

We used Bootstrap mainly because it can easily implement a responsive design and become mobile compatible. Bootstrap can be easily integrated so it saved us a lot of time. With that mentioned, we are able to focus on the development of the application. It is also used with CSS3 to stylize our application.

Finally, JavaScript is used in the client-side to implement and control the user interface. It is used with HTML. The result of using of these technologies allowed us to create a cross platform, user friendly application with an elegant design.

**Improvements and Future Implementation**

There are numerous of possible future improvements for our web application. The following are some ideas we plan on implementing:

* Private chat rooms that would require a password to join.
* Facebook/google+ integration.
* Web browser games. Users will be able to play games together while they are chatting.
* Send photos, videos, voice messages, files.
* Group chat. Users will be able to talk to more than one person.
* Friends list. Users have the option to add user to their friends list.
* Implementation that lets the user know if strangers as well as friends are within a radius range.
* Connect people based upon interests by a certain distance between them.
* Save chat conversation and allows user to view the chat history.
* A regional chat for each country or province.
* Search by topic, gender, country and city. For example, User A want to talk about computers and users matched has to also be a male and is from Tokyo, Japan.
* Video chat. Users will able to talk via video streaming.
* Suggest similar topics based on user’s previous searched topics
* User can have multiple chats, each with a different user and topic
* Premium account, users will have access to special features such as chat skins, emoticons and etc.
* Users can monetize and upload their own chat skins and emoticons for sale.
* Users are able to customize their profile page including the template of the page.
* Encrypted database.
* Have a list of popular/trending topics searched on the web application.
* Dynamic theme on application, theme changes based on holidays and local weather. If the local weather of the user is rainy then the theme will change to gloomy.
* Customer support
* FAQ User forms
* SQL Injections defensive security mechanism
* Secure http connection
* Random topic generator
* Automatically translate foreign language to main language
* Web application available in different languages
* Danger/Help shout, allows users to call for help which will alert users in the nearby area to come and assist or avoid areas
* A ban system, will ban users that abuses the application or harm other users

**Conclusion**

In conclusion, our application will make the world great again. BabbleT is a social interactive web application. The application allows people meet with new people around the world, anywhere and anytime. People can search a topic and the application will automatically find all the compatible users. They can select one of the users and start chatting. The chat is one on one and is private. In addition, users can personalize their profile and change the chat background to their preference. Users can click on the username of the other user in the chat to visit their profile. The application is mobile compatible. Users can use the application on phones, tablets, laptops and etc.

We have encountered numerous of challenging obstacles but we were able to pull through. We divided our development evenly in our team. Some members would work on server-side while the rest works on the client-side at the same time. As a result, we were able to solve problems and develop efficiently and quickly.

The lessons we learned is that planning carefully before the start of development is very important because it will minimize the amount of adjustments during the development phases and back tracking to modify. We have learned a lot about a chatting system, it was not as easy as it seems. Implementing a chat system from scratch without using frameworks or built-in functions gave us valuable experience in the technologies we used. If we can do this project again, we would try using a framework. Overall, we are satisfied and proud of what we have developed over the past months.

**References**

<http://chatroulette.com/>

<http://dev.mysql.com/doc/refman/5.7/en/>

<https://www.facebook.com/>

<https://www.gotinder.com/>

<http://php.net/manual/en/langref.php>

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

<http://www.w3schools.com/bootstrap/default.asp>

<http://www.w3schools.com/css/default.asp>

<http://www.w3schools.com/html/default.asp>

<http://www.w3schools.com/jquery/default.asp>

<http://www.w3schools.com/js/default.asp>

<http://www.w3schools.com/php/default.asp>

<http://www.w3schools.com/sql/default.asp>

**Appendix**