Maher Algibhah

Faisal Albellaihi

Tommy Mason

Jonathan Hernandez

Chris Kim

Enzyme

1. **Introduction:**

**1.1 Purpose:**

Sharing ideas with others is a very interesting and desiring goal for many people. This document is the specification for the software “Enzyme” which is a social network application. The software targets people who wants to discuss their ideas with their friends or even the whole world to get their feedback. Enzyme users can share their ideas and thoughts using plain text, photos audio and/or video.

**1.2 System Scope:**

Enzyme is a social network application that utilizes micro-blogging which allows registered users to post messages that can either be public or private and can be directed to a single user or a group of users. All users have the ability to view public messages however registered users can have their own profile, post messages under 140 characters, receive private messages, follow other users, and search for trending topic and users.

**1.3 Overview of the document:**

1. Introduction
   1. Purpose
   2. System Scope
   3. Overview of the document
2. Overall Description

This section explains background information about the system, defining who are the clients and users of the system and describes in detail the system functionality.

2.1 Client characteristics

2.2 User characteristics

2.3 Product functions

2.3.1 General features

2.3.2 In-depth features

2.3.3 Use Cases

* + - 1. View public messages
      2. Create a public message
      3. User registration
      4. User login
      5. User Logout
      6. Delete messages
      7. Direct messages
      8. Tagging a user
      9. Listing relevant topics
      10. Listing trending hashtags
      11. Saving a post
      12. Subscribing to another user’s messages
      13. Timeline activity page.
      14. Post a comment
      15. Create a private message
      16. Searching for specific user or hashtag
      17. Delete profile
      18. Repost message
      19. Customize Profile

1. System Specification

This section explains the system design and showing class diagrams showing the classes and their relations.

1. Assumptions

List all the assumptions made about the system.

1. **Overall Description:**

.

**2.1 Client Characteristics:**

Enzyme is being written for our client Protein Incorporated. The system is important to Protein Incorporated because they want to connect people from all over the world. Protein Incorporated wants to share ideas with other companies and individuals to be able to produce products more efficiently than ever before. Enzyme will allow Protein Incorporated to talk to other companies that have ideas on how to improve efficiency in all aspects of their company. Protein Incorporated will increase in whatever aspect they need to in a drastic way because of their capability to communicate with others.  This will allow companies that do not use Enzyme to see Protein Incorporated’s success and then they too will use Enzyme to connect with others.

**2.2 User Characteristic**

The intended users of Enzyme are those who wish to communicate with others in an efficient way. The users can talk to other people about whatever they wish. The age range of Enzyme is projected for people from 12 years of age to people of 50 years of age. These people can also be parts of groups represented by one account. For groups that are represented by one account, they may choose to put more relevant information on their profile (i.e. an address and not a birthday).

**2.3 Product functions**

All users have the ability to view public messages however registered users can have their own profile, post messages under 140 characters, receive private messages, follow other users, and search for trending topic and users.

**2.3.1 General features:**

1. Universal Desktop application that works in all versions of Windows and Mac OS.
2. Web application allowing the software to run in any web browser including browsers running in mobile devices (Android, Blackberry, Windows mobile and IOS).
3. Secure login/registration system.
4. Text data is represented using UTF8 to provide Multilanguage support.
5. Data is stored in MySQL database.

**2.3.2 In-depth features:**

1. Public messages are available for non-registered users.
2. Customizable profiles with avatars for registered users.
3. Follow registered users.
4. Repost feature.
5. Hashtaging posts.
6. Allow comments on posts.
7. Live private chat.
8. Allow private posts.
9. Upload photos and other files.
10. Allow deleting messages.
11. Notify users for posted messages.
12. Support blacklist keyword to prevent displaying unwanted words.
13. Display relative topics and/or posts.
14. Search posts.
15. Find registered users.

Each part in the system has its function that will be working with the other system functions in order to make the user interface more friendly and familiar for the user. An example will be when the favorite’s mechanism works alongside with the trending hashtags and or tweets. The system will extend the usage of these functions in the system in order to make it easier for the user in this case have that information more accessible. The main job of the system is to execute what the user wants to do as fast and efficient as possible. Another example of this efficiency can be seen when the user deletes a message. The use expects to have that message gone from both the timeline and other pages displaying their posts. This system will always make sure this is the case for the user and the only way this will be possible is through many functions that the system will have to catch these sort of things. In other words the system is in charge of all the things that will be effecting the user through their user interface experience. All the information that the system will be storing is going into a database and will only be made accessible to the admins of the site.

**2.3.3 Use Cases:**

This section will cover different scenarios showing different interactions between users and the system.

**1st case**

**Title**: View public messages

**Description**: users including unregistered users view public messages posted by registered users.

**Main Flow**:

1. User double clicks on the application icon

2. System displays a window dialog asking for user’s credentials or selecting anonymous user (not registered user).

3. Displaying the system main page showing the latest public posted messages.

**Alternate flow:**

1. User goes to the website of the system using a browser

2. System displays a window dialog asking for user’s credentials or selecting anonymous user (not registered user).

3. Displaying the system main page showing the latest public posted messages.

**2nd case**

**Title**: Create a public message

**Description:** user indicates that he/she wants to send a public message enters the message, and the system displays the message to all users.

**Main Flow**:

1. User clicks on the message window

2. System makes the message window active

3. User types the text message

4. User clicks on the "send public message" button

5. System verifies that the message is less than 140 characters

6. Add the message to the "database"

7. System updates display to include the new public message

**Alternate flow:**

1. User clicks on the message window

2. System makes the message window active

3. User types the text message

4. User clicks on the "send public message" button

5. System finds message is less than 140 char

6. System alerts user that message is too long and asks user to make the message shorter

**3rd case**

**Title**: User registration

**Description:** Users create an account

**Main Flow**:

1. User clicks on the register button

2. System brings up the register window

3. User enters desired username

4. User enters password

5. User enters date of birth

6. User clicks register button

7. System checks to see if desired username is taken

8. System confirms it is not and registers user

9. System brings home screen back up

**Alternate flow:**

1. User clicks on the register button

2. System brings up the register window

3. User enters desired username

4. User enters password

5. User enters date of birth

6. User clicks register button

7. System checks if desired username is available

8. System says it is taken

9. System brings up registration window again

**4th case**

**Title**: User login

**Description:** A user indicates that they would like to login to the website. The system will check the credentials that are provided against what has been stored.

**Main Flow**:

1. User clicks on the login button

2. User enters username

3. User enters password

4. User clicks login

5. System checks to see if the password and username correspond to what is stored

6. System confirms that it is correct

7. System brings the user back to the home screen logged in

**Alternate flow:**

1. User clicks on the login button

2. User enters username

3. User enters password

4. User clicks login

5. System checks to see if the password and username correspond to what is stored

6. System says it is not

7. Alerts user that one of the fields is not correct

**5th case**

**Title**: User Logout

**Description:** when the user desires to logout, the user will click the logout button.

**Main Flow**:

1. User clicks the logout button

2. Checks to see if there is a user logged in

3. System logs user out

**Alternate flow:**

1. User clicks the logout button

2. Checks to see if there is a user logged in

3. System says that you need to log in before logging out.

**6th case**

**Title**: Delete messages

**Description:** When a logged in user decides to delete a new or previous message that he or she has posted, they can click on a delete button.

**Main Flow**:

1. User decides that he or she wants to delete a message

2. User goes to message and presses delete message button

3. The user is asked if they are sure about trying to delete the message

4. User presses the yes button

5. Message is deleted

**Alternate flow:**

1. User decides that he or she wants to delete a message

2. User goes to message and presses delete message button

3. The user is asked if they are sure about trying to delete the message

4. Users says no

5. User is sent back to their news feed and the message isn’t deleted

**7th case**

**Title**: Direct messages

**Description:** A logged in user is allowed to start a chat with another user that allows them to privately communicate with each other.

**Main Flow**:

1. Logged in user clicks on the direct message button

2. A chat box shows up asking the user who they want to start a chat

3. User inputs a username

4. System checks if the user is online

5. The user is online

6. System sends message and starts the chat with the other user

**Alternate flow:**

1. Logged in user clicks on the direct message button

2. A chat box shows up asking the user who they want to start a chat

3. User inputs a username

4. User inputs a message in the chat box

5. System checks if the user is online

6. The user is offline

7. The message is still sent, but user is returned to the home screen

**8th case**

**Title**: Tagging a user

**Description:** A user can notify another user of a specific message.

**Main Flow**:

1. User includes the other user’s username preceded by the @ symbol in the chosen message.

2. System catches strings led by the @ symbol and inputs the username to check validity

3. User of the username is notified of being mentioned in the message.

**Alternate flow:**

1. User writes the message with an @ followed by a username

2. System catches the @username and inputs the username

3. If username does not exist, system stops and notifies user

**9th case**

**Title**: Listing relevant topics

**Description:** A user can add hashtags to a message to show up on relevant searches.

**Main Flow**:

1. User includes relevant topics preceded by the # symbol at the end of the message

2. System catches values with the # symbol in the message

3. System keeps track of hashtag values in a list based on number of times it is hashtagged

**10th case**

**Title**: Listing trending hashtags

**Description:** User can click on trending topics link to see a list of top 10 most popular hashtags, displaying the most favorited post for each hashtag.

**Main Flow**:

1. User clicks on trending topics link

2. System calls hashtag list and displays the list in order of popularity

3. For each hashtag, the system calls up the most favorited hashtagged post

**11th case**

**Title**: Saving a post

**Description:** User can click on a favorite button by any message and save that message on a favorites list.

**Main Flow**:

1. User clicks on favorite button

2. System copies the message and saves it to the user’s favorites list

3. System keeps count for each message how many times it has been saved.

**Alternate flow:**

1. User clicks on favorite button

2. User is not registered and system redirects to register page

**12th case**

**Title**: Subscribing to another user’s messages

**Description:** A user can follow another user’s messages by subscribing to the other user’s profile.

**Main Flow**:

1. User clicks on the desired profile

2. User clicks on subscribe button for that desired profile.

3. The system adds the user to the subscribed list..

**Alternate flow:**

1. User clicks on list of followers

2. User clicks on subscribe button for the desired profile who follows the user.

3. System adds the user to the subscribed list

**13th case**

**Title**: Timeline activity page.

**Description:** If a user is logged in, the timeline page will load the last activities done by the followed users. If user is not logged in, then the user can only see public messages, and the timeline page will load random public messages.

**Main Flow**:

1. A user logs in into user’s account

2. User clicks on the timeline button.

3. System checks the users’ subscribed list.

4. System loads the followed user’s private and public messages.

5. System loads the timeline page

**Alternate flow:**

1. User clicks on the timeline.

2. System loads random public messages

3. System loads the timeline page

**14th case**

**Title**: Post a comment.

**Description:** Logged on users can comment on another user’s messages. Depending on the privacy settings for the other users, the message might need to be approved or might be posted immediately without approval.

**Main Flow**:

1. User login

2. User clicks on the message that the user wants to comment in.

3. User enter text into the comment box

4. User clicks on the submit comment button.

5. System notify the message poster about a new comment has been posted.

6. System checks the privacy settings

7. System reloads the post with the new comment

5. System loads the timeline page

**Alternate flow:**

1. User login

2. User clicks on the message that the user wants to comment in

3. User enter text into the comment box

4. User clicks on the submit comment button.

5. System notify the message poster about a new comment has been posted.

6. System checks the privacy settings

7. System request approval from the message poster

8. Message poster approves the comment

9. System reloads the post with the new comment

**15th case**

**Title**: Create a private message

**Description:** User indicates that he/she wants to send a private message enters the message, and the system displays the message only to specific users.

**Main Flow**:

1. User clicks on the message window

2. System makes the message window active

3. User types the text message

4. User clicks on the "send private message" button

5. System bring up a window where the user can write the users names that he/she wants them to see the message

6. System verifies that the message is less than 140 characters

7. Add the message to the "database"

8. System notify the specific users about the message

**Alternate flow:**

1. User clicks on the message window

2. System makes the message window active

3. User types the text message

4. User clicks on the "send private message" button

5. System bring up a window where the user can write the users names that he/she wants them to see the message

6. System finds message is more than 140 char

7. System alerts user that message is too long and asks user to make the message shorter

**16th case**

**Title**: Searching for specific user or hashtag

**Description:** A user can search for specific user by typing the username, the user also can search for hashtag by typing the symbol # before a relevant keyword or phrase (no spaces)

**Main Flow**:

1. User input the username that he/she wants to search for, into the box next to the “search” button)

2. User clicks on "search" button

3. System search for the inputted username into the users list

4. System show up the user

**Alternate flow:**

1. User input the symbol #before a relevant keyword or phrase (with no spaces), that he/she wants to search for, into the box next to the “search” button)

2. User clicks on "search" button

3. System search for the inputted hashtag

4. System bring up all the messages that have the inputted hashtag

**17th case**

**Title**: Delete profile

**Description:** A user can delete his/her profile

**Main Flow**:

1. User clicks on “My profile” button

2. System brings up the user’s profile window

3. User clicks on “delete my profile”

4. System brings up a confirmation window with two buttons “Cancel”, “Delete”

5. User clicks on “Delete” button

6. System delete the user form the users list

7. System brings up a message saying “your profile has been deleted successfully”

8. System brings up the main menu window

**Alternate flow:**

1. User clicks on “My profile” button

2. System brings up the user’s profile window

3. User clicks on “delete my profile”

4. System brings up a confirmation window with two buttons “Cancel”, “Delete”

5. User clicks on “Cancel” button

6. System brings up the user profile window

**18th case**

**Title**: Repost message

**Description:** user can repost a message that has been posted

**Main Flow**:

1. User selects a message that has been that has been posted

2. User selects the repost button

3. System generates a messages using the message that got reposted

4. User types additional content

5. User clicks send

6. System displays message and sorts it

**Alternate flow:**

1. User selects a message that has been that has been posted

2. User selects the repost button

3. System generates a messages using the message that got reposted

4. User clicks send

5. System and stores the message

**19th case**

**Title**: Customize Profile

**Description:** User can edit their profile

**Main Flow**:

1. User clicks on their profile

2. User clicks edit profile

3. System brings up selection windows

4. User clicks profile picture

5. System brings up file window

6. User selects picture set

7. System replaces old picture with new

8. System redirects to their profile page with the new picture

**Alternate flow:**

1. User clicks on their profile

2. User clicks edit profile

3. System brings up selection windows

4. User clicks background

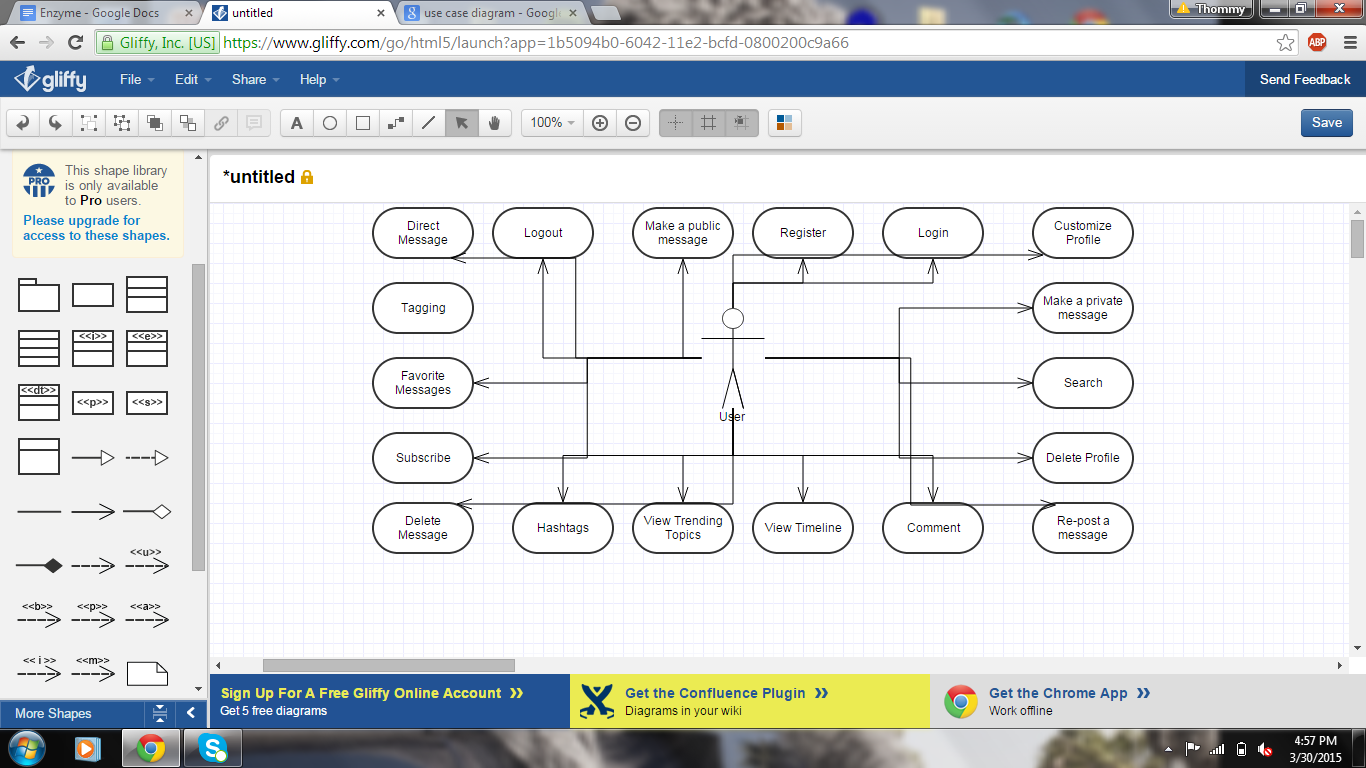
5. System brings up file window

6. User selects picture to set

7. System replaces old picture with new

8. System redirects to their profile page with the new picture

Figure # (1)



1. **System Specifications**

The system consist of three layers.

**Layer 1: Database layer**

The system uses MySQL as the database to store the data which includes user information data and messages data. MySQL is an open source relational database management system which means it stores data in tables (Users table and Message table).

Users Table

|  |  |  |  |
| --- | --- | --- | --- |
| userID | username | password | birthdate |

Message Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| messageID | userID | content | hashtag | date |

Comment Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| commentID | messageID | userID | comment | date |

Profile Table

|  |  |  |
| --- | --- | --- |
| PrfileID | userID | background |

**Layer 2: Application Layer**

The system is built using Java programing language. It uses MVC pattern.

**Model classes:**

**User class**: Class representing a user in the system. It stores username, password, birthdate, and a list of messages posted by the user. It includes the methods needed to get and set the above data.

**Message class**: class representing a posted message. It stores messageID, messageContent, a reference to the user object who posted the message, postingDate, and hashtag of the message. It includes the methods needed to get and set the above data.

**Profile class**: A class representing profile settings like background color. It stores profileID, userID, and background. It includes the methods needed to get and set the above data.

**Comment class**: A class representing comment on a message. It stores commentID, messageID, userID, comment, and date. It includes the methods needed to get and set the above data.

**View classes:**

MainWindow class: A GUI class showing the main screen of the system. From this screen, a user can log-in, log-out, view messages, view profile settings, and etc.

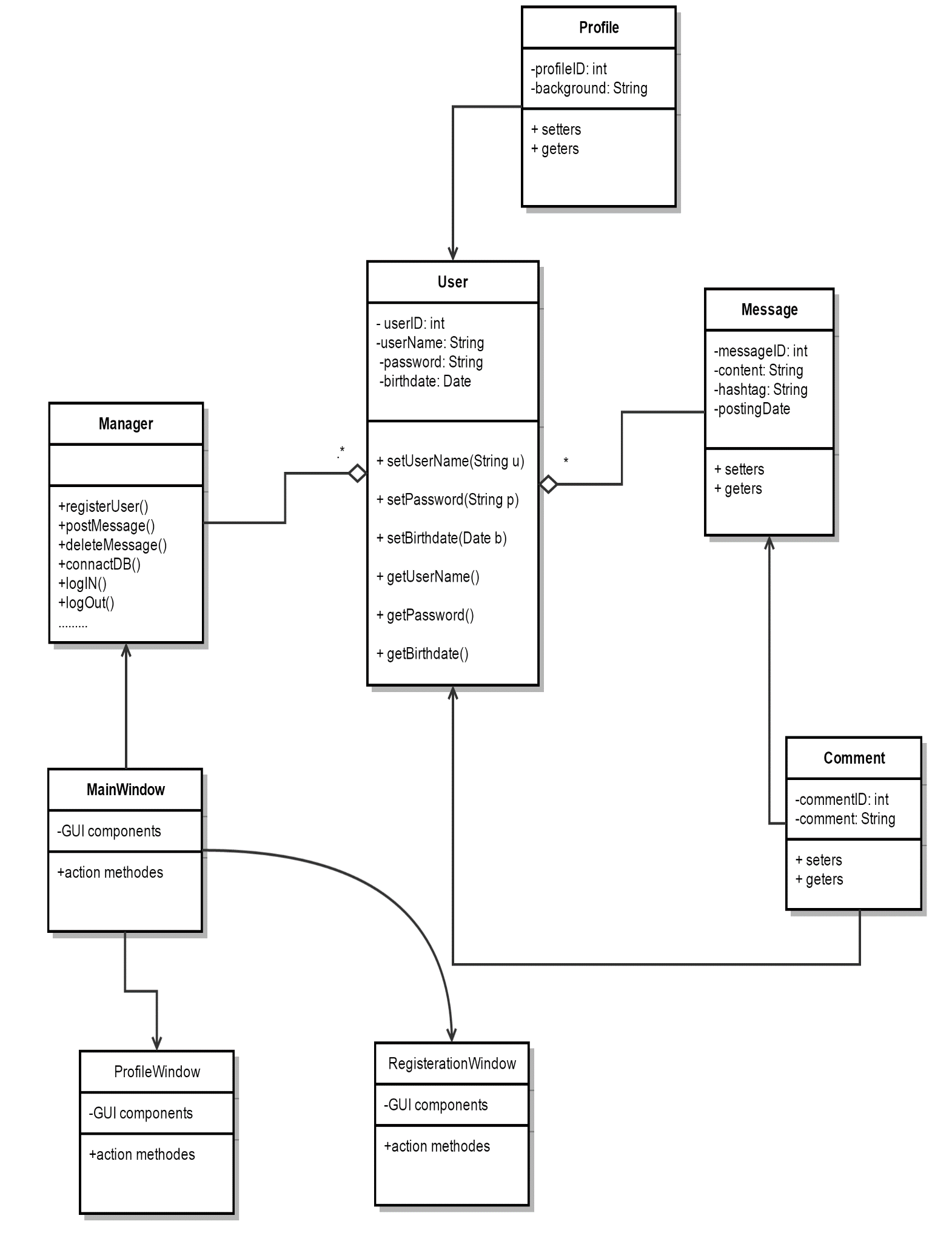
**RegestringWindow calss**: A GUI class showing the registration form allowing the user to create an account.

**ProfileWindw class**: A GUI class showing the profile settings of a user. This screen allows updating the profile settings.

**Controller class:**

**Manager class:** A class to read and update data represented by the model classes in response to commands received from the View classes. It also includes methods to connect to MySQL database to retrieve/stores date

Figure # (2)



1. **Assumptions**.

List of assumptions made on the software:

* Users have good internet connection.
* Web server is available and supports MySQL database.
* Users know what buttons on screen do
* Users will always remember their password
* Users will always remember their username
* Users know how to refresh the page if it is not loading correctly
* Users can read English
* The user will know where to write a message
* The user will know how to customize their profile page
* The user will know how to search the site
* The user will know how to subscribe to other users
* The user will know where on the main page the trending tweets are located
* The user should know where the options button is