



University of Pisa
MSc in Computer Engineering
Information Systems

Social Network - Task 1

Sara Lotano
Lorenzo Simi
Filippo Storniolo
Matteo Suffredini

Academic Year 2019/2020

Contents

1	Introduction	2
1.1	Functional Requirements	2
1.2	Non-functional Requirements	4
1.3	Actor of the system	4
2	Design	5
2.1	Use cases	5
2.2	Class Diagram	6
2.3	Design Pattern	7
3	Implementation	8
3.1	Database Structure	8
3.1.1	Person Table	8
3.1.2	Post Table	8
3.1.3	Comment Table	9
4	Manual of usage	10
4.1	Login	10
4.2	Registration	11
4.3	Add post	12
4.4	Modify post	13
4.5	Delete post	14
4.6	Search post content	15
4.7	Search user posts	16
4.8	Comment functionalities	17
4.9	Edit profile	18
4.10	Search user	19
5	Key-Value Databases	20
5.1	Introduction to Key-Value Databases	20
5.2	Feasibility study	20
5.3	LevelDB	21

1 Introduction

Social Network is an application in which users can create new discussion topics and express their opinion on content shared by other users.

Each user, to access the platform, needs an account that uniquely identifies him within the system.

When the user launches the application, if it is already registered to the system, he must enter his username and password and access the platform.

If the user is not yet registered, he will fill a form containing the information required by the system like: username, password, name, surname, gender, date of birth, country, city, address and telephone number.

The user can view, in the home page of the application, all the posts created by all the users. The user can create a new post, edit or delete an old post. Only the user who created the post can edit the content or delete it.

The user can search for a post in two different ways: by indicating a keyword contained in the text of the post or by entering the username of the person who created the post.

The user can view also all the comments related to a post. As with posts, the user has at his disposal several functionalities: creating a new comment, editing or deleting a previously published comment. Only the user who created the comment can edit the content or delete it.

The user can view the profile of another user and all the related information.

The user must have the possibility to modify his personal information.

1.1 Functional Requirements

- The system shall allow users to communicate, through messages and comments, with other users registered in the system.
- If the user is already registered then he shall enter his username and password in the respective fields and press the **Login** button, otherwise he shall press the **Registration** button.
- If the user pressed the **Registration** button, the system shall show a form with the following mandatory information: username and password and other optional information: name, surname, gender, date of birth, country, city, address and telephone number.

- When the **Save** button is pressed, the system shall store user information.
- If the user is registered and logged in, the system shall show the posts published by all registered users on the social network. For each post, the user shall see: the username of the post creator, the content of the post, the number of comments and the creation date.
- The user shall create new posts. When **Add Post** button is pressed, the system shall store in the database all the information: postID, personID, message content, timestamp related to its creation.
- The user shall edit or delete the posts he has created. When **Enter** or **Delete** button is pressed, the system shall save all the changes in the database and update all the related information in the interface.
- The user shall search posts by indicating a keyword contained in the text of the post or by entering the username of the user who created the post. When one of the two buttons (**Search by Content** and **Search by User**) is pressed the system shall show the results of the search in the post table.
- If the user press **View all posts**, the system shall show again all the posts in the post table.
- If the user selects a post which contains at least one comment, then the system shall show the related comments in the comments table.
- The user shall create new comments. When the **Add Comment** button is pressed, the system shall store all the information in the database: commentID, postID, personID, content of the comment, timestamp relative to its creation.
- The user shall edit or delete the comments he has created. When **Enter** or **Delete** button is pressed, the system shall save all the changes in the database and update all the related information in the interface.
- The user shall search comments by indicating a keyword contained in the text of the comment or by entering the username of the user who created the comment. When one of the two buttons (**Search by Content** and **Search by User**) is pressed, the system shall show the results of the search in the comment table.
- The user shall be able to search for another user by entering the username in the Search User bar.

- When **Search User** button is pressed, the system shall show the information that the selected user has chosen to enter.
- The user shall be able to change his or her personal information. When button is pressed, the system shall display the personal information previously entered. If the user changes any information and **Save** button is pressed, then the system shall save the changes.
- The user shall be able to exit the application. When the **Logout** button is pressed, the system shall delete all local user data.

1.2 Non-functional Requirements

- **Portability:** the code shall be not platform dependent, since the programmer shall provide a auto-executable project which can work in different environments.
- **Readability:** the code shall be readable, since it shall consist of classes divided by role, each of them providing a specific set of functions.
- **Usability:** the application shall be user-friendly, since it shall consist of a graphical interface and the system shall guide the user to the correct choices through custom error messages and hints.
- **Security:** the system shall store password in a secure way, so they will be encrypted and not shown in clear.

1.3 Actor of the system

After the analysis of the requirements, the main actor identified is:

- **User :** a person who has joined the social network and is able to interact with other users and create new contents.

2 Design

2.1 Use cases

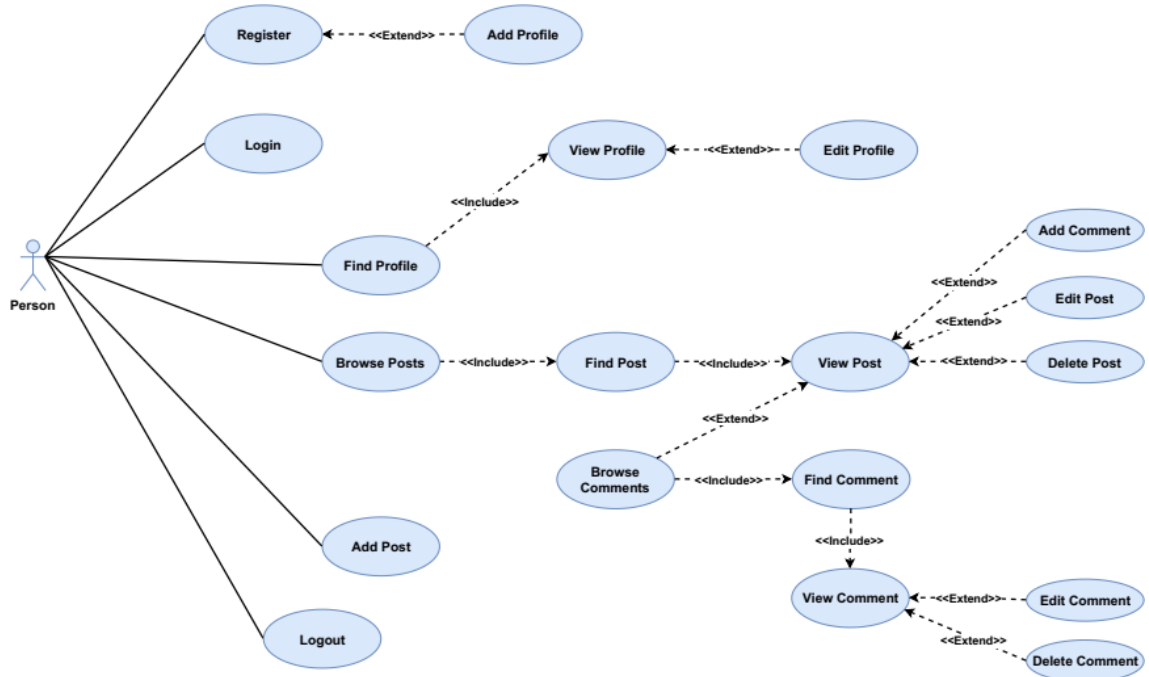


Figure 1: Use cases

This image represents the interaction between the main actor and the application. In particular, only the owner is allowed to modify his own profile and can also view other user profiles. It is mandatory to login into the system to perform any kind of actions except for the registration.

2.2 Class Diagram

The main data structures identified within the system are the following:

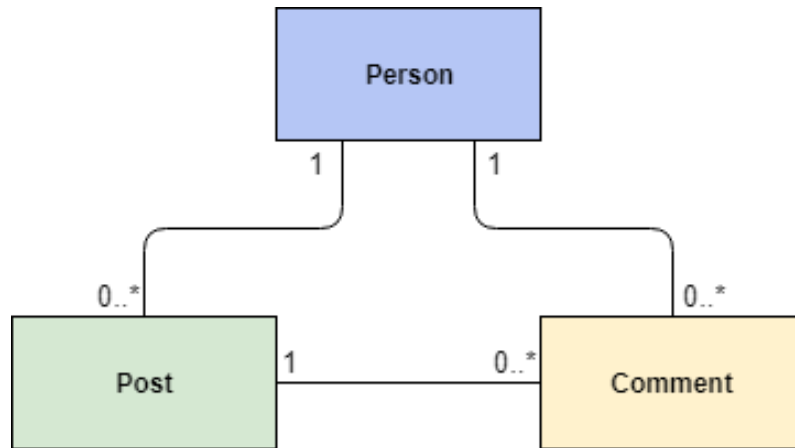


Figure 2: Class Diagram

- A **Person** can write zero or more Posts and zero or more Comments related to a Post.
- A **Post** is written by only one Person and can have zero or more Comments related to it.
- A **Comment** is written by only one Person and is related to only one Post.

2.3 Design Pattern

The design pattern chosen is the **Model-View-Controller** pattern.

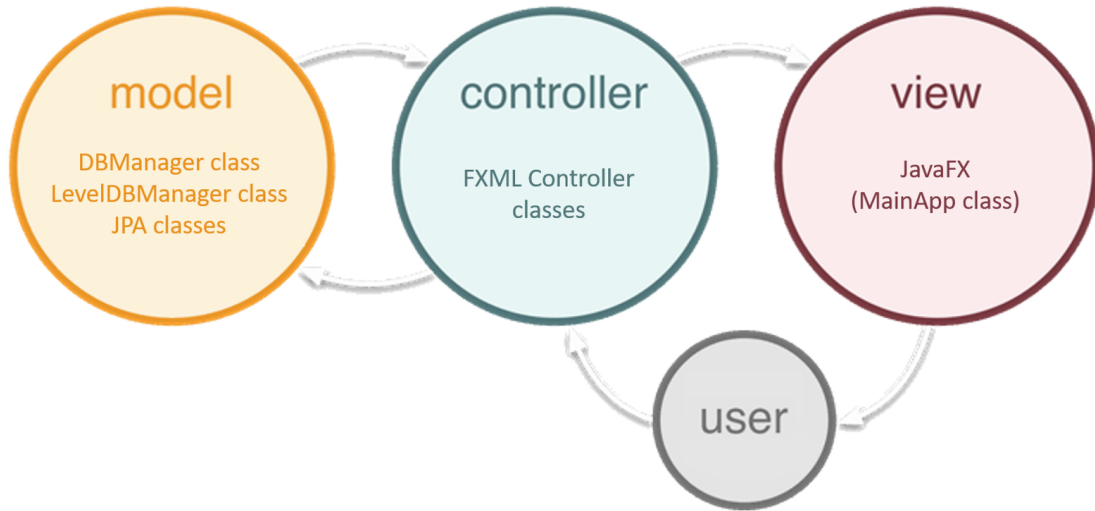


Figure 3: Model View Controller

This pattern divides the application in three part and each one has a well defined purpose.

- **Model:** is the one that has to manage the data.
- **View:** is the one that has to communicate with the user.
- **Controller:** is the one that has to coordinate interactions between view and model.

This pattern is perfectly suitable for the application proposed.

3 Implementation

3.1 Database Structure

The database is composed by the following tables:

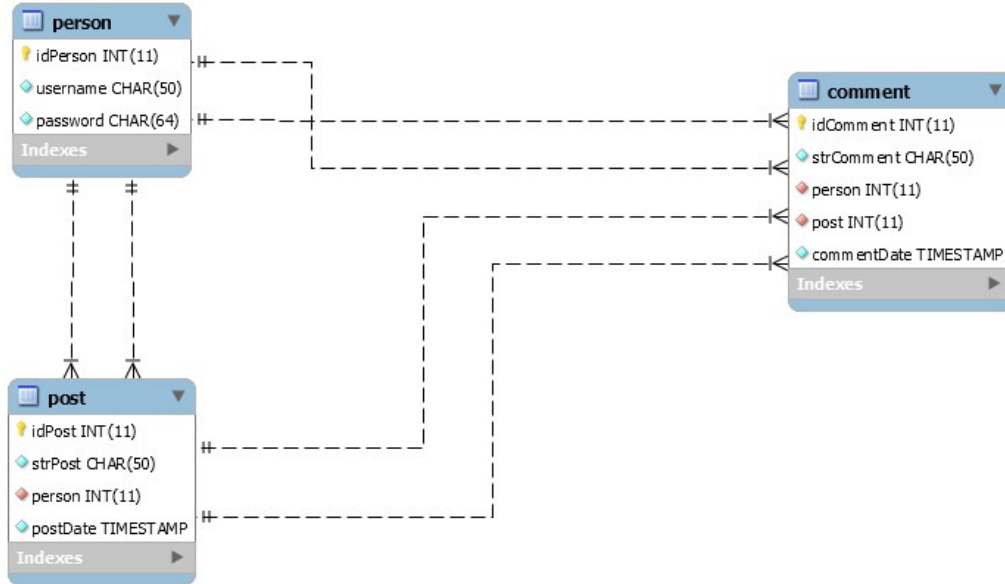


Figure 4: Database structure

3.1.1 Person Table

The Person Table has three fields:

- **IdPerson:** It will represent the primary key of the table and will contain the unique Id associated to a single person.
- **Username:** It will contain the user nickname.
- **Password:** It will contain the user password.

All these fields must be not null.

3.1.2 Post Table

The Post Table has four fields:

- **IdPost:** It will represent the primary key of the table and will contain the unique Id associated to a single post.

- **strPost:** It will contain the post content represented as a string. The maximum post length is 50 characters.
- **Person:** It will contain the person Id of the person who published the post.
- **Date:** It will contain the time in timestamp format when the user published the post.

All these fields must be not null.

3.1.3 Comment Table

The Comment Table has five fields:

- **IdComment:** It will represent the primary key of the table and will contain the unique Id associated to a single comment.
- **strComment:** It will contain the comment content represented as a string. The maximum comment length is 50 characters.
- **Person:** It will contain the person Id of the person who published the comment.
- **Post:** It will contain the post Id the comment is associated with.
- **Date:** It will contain the time in timestamp format when the user published the comment.

All these fields must be not null.

4 Manual of usage

4.1 Login

If the user is already registered to the system, he must enter:

- Username
- Password

and press the **Login** button. If username and password are correct, the user will be logged into the system.

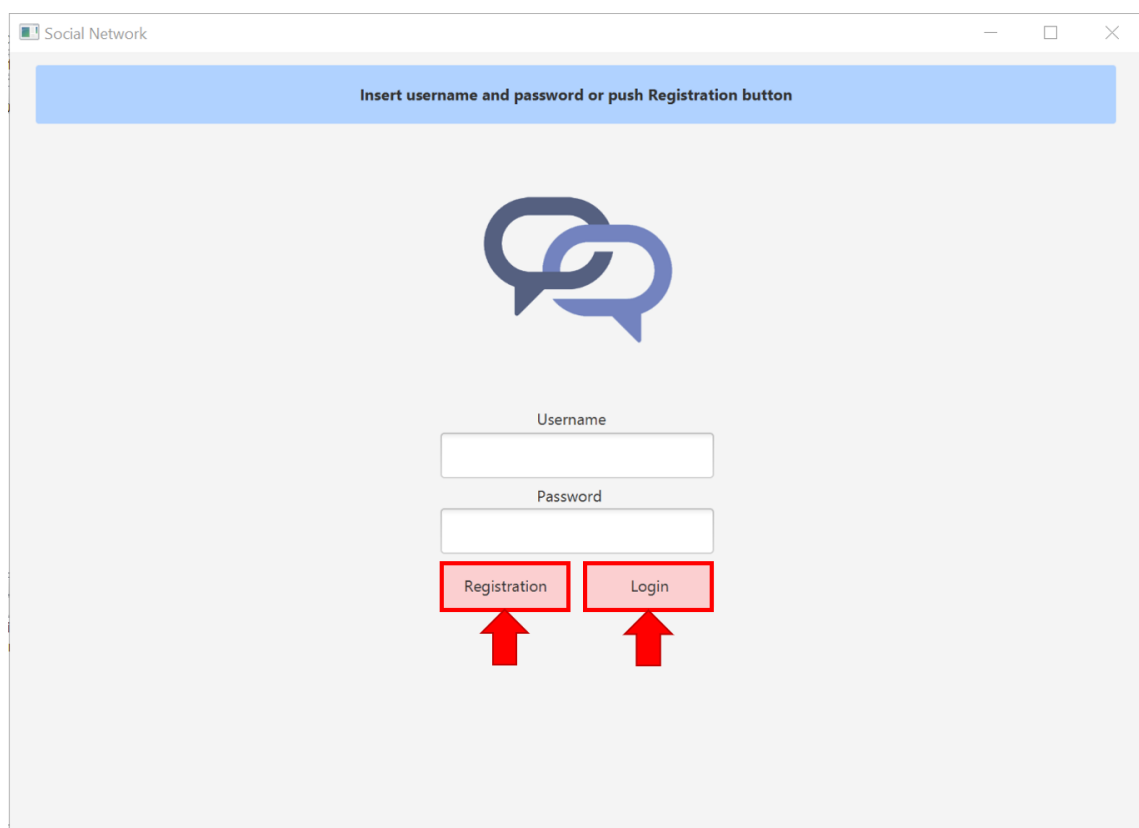
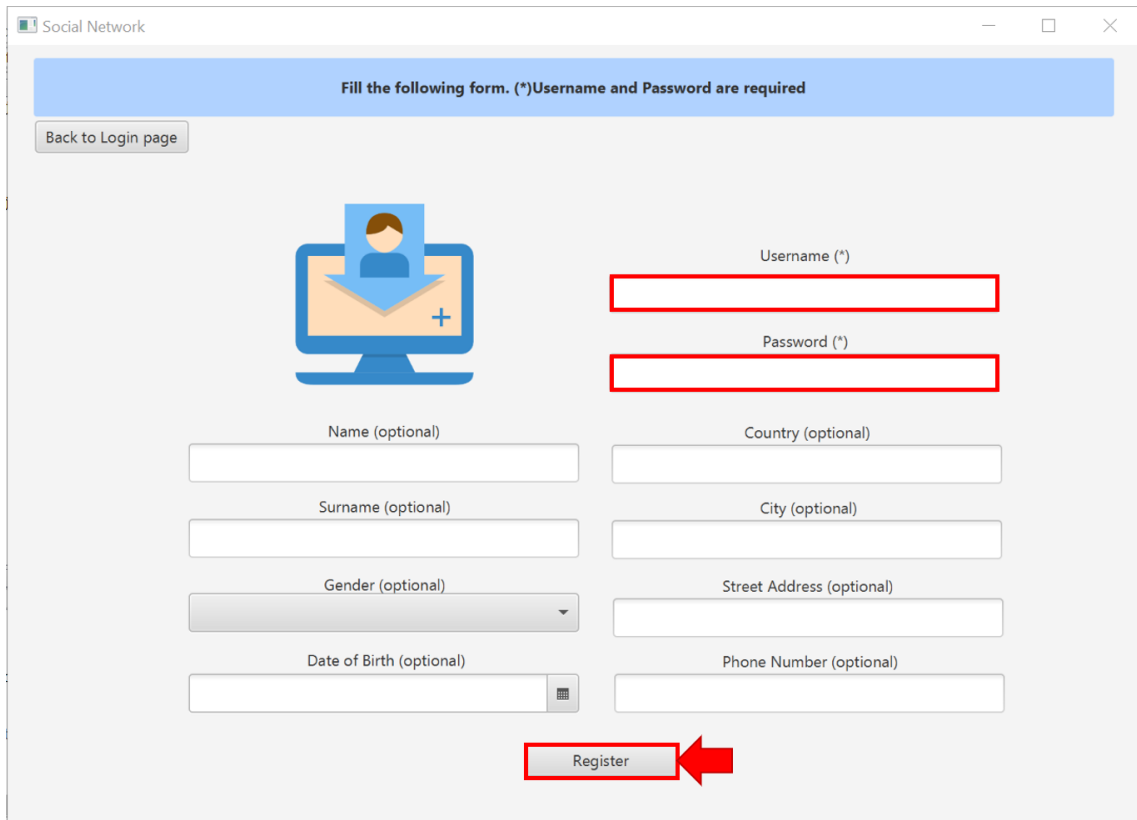
A screenshot of a web browser window titled "Social Network". The browser's address bar is empty. The page has a light blue header bar with the text "Insert username and password or push Registration button". Below the header, there is a large blue speech bubble icon. Underneath the icon, there are two input fields: the first is labeled "Username" and the second is labeled "Password". Below these fields are two buttons: "Registration" and "Login". Both buttons are highlighted with red rectangular boxes, and red arrows point upwards to each button. The background of the page is a light gray.

Figure 5: Login phase

If the user is not yet registered, he has to press the **Registration** button.

4.2 Registration

The user must enter the two fields relating to the username and password. In addition, he can enter other optional information such as: name, surname, gender, date of birth, country, city, address and telephone number. At the end, he must press the **Register** button through which he will access the home page.



The image shows a web browser window titled "Social Network". Inside the window, there is a registration form. At the top, a blue banner contains the text "Fill the following form. (*)Username and Password are required". Below this banner is a "Back to Login page" button. The form itself is divided into two columns. On the left column, there is an icon of a computer monitor with a person and a plus sign, followed by input fields for "Name (optional)", "Surname (optional)", "Gender (optional)" (a dropdown menu), and "Date of Birth (optional)" (with a calendar icon). On the right column, there are input fields for "Username (*)", "Password (*)", "Country (optional)", "City (optional)", "Street Address (optional)", and "Phone Number (optional)". At the bottom center of the form is a "Register" button, which is highlighted with a red rectangle and a red arrow pointing to it from the right.

Figure 6: Registration phase

4.3 Add post

The user can write a new post by inserting the text into the box highlighted in the figure. Each post has a limit of 50 characters. After writing the content, the user can press the **Add Post** button.

The screenshot shows a web application titled "Social Network". At the top, a blue banner displays the message: "Hello Matteo, you've written 7 posts and 4 comments". Below this, on the left, is a sidebar with a speech bubble icon, a "Search User" input field with a "Search User" button, and two buttons: "Edit your Profile" and "Logout".

The main content area is divided into two sections. The top section features a table with the following data:

User	Post	Comments	Date
Mario	In two days i will go to walk on the beach	4	2018-12-13 07:05:24.0
Gianmarco	If I'm free tomorrow, I'll go to swim in the sea	14	2018-12-11 10:12:26.0
Sara	Today I'm going to study!	5	2018-12-11 07:23:15.0
Lorenzo	Yesterday I went to the restaurant	9	2018-12-05 23:13:25.0
Filippo	Two days ago i went to study	12	2018-12-05 05:20:46.0
Franco	Yesterday I went to walk in the mountains	1	2018-11-27 02:18:44.0

Below the table are search filters: "Search Post", "Search by Content", "Search by User", "View all posts", and a "Delete" button.

The bottom section of the main content area contains a form for adding a new post or comment. It has a header "Insert post or comment here:" and a text input field labeled "New post". Below the input field are two buttons: "Add Post" (highlighted with a red box and a red arrow) and "Add Comment".

To the right of the form is a table for comments, which is currently empty and displays the message "Nessun contenuto nella tabella". It has columns for "User", "Comment", and "Date". Below this table are search filters: "Search Comment", "Search by Content", "Search by User", and a "Delete" button.

Figure 7: Add post

4.4 Modify post

The user can only modify the content of the posts created by him. This is possible by selecting the post to be edited and clicking on it to make the content editable. The edit is saved by pressing Enter.

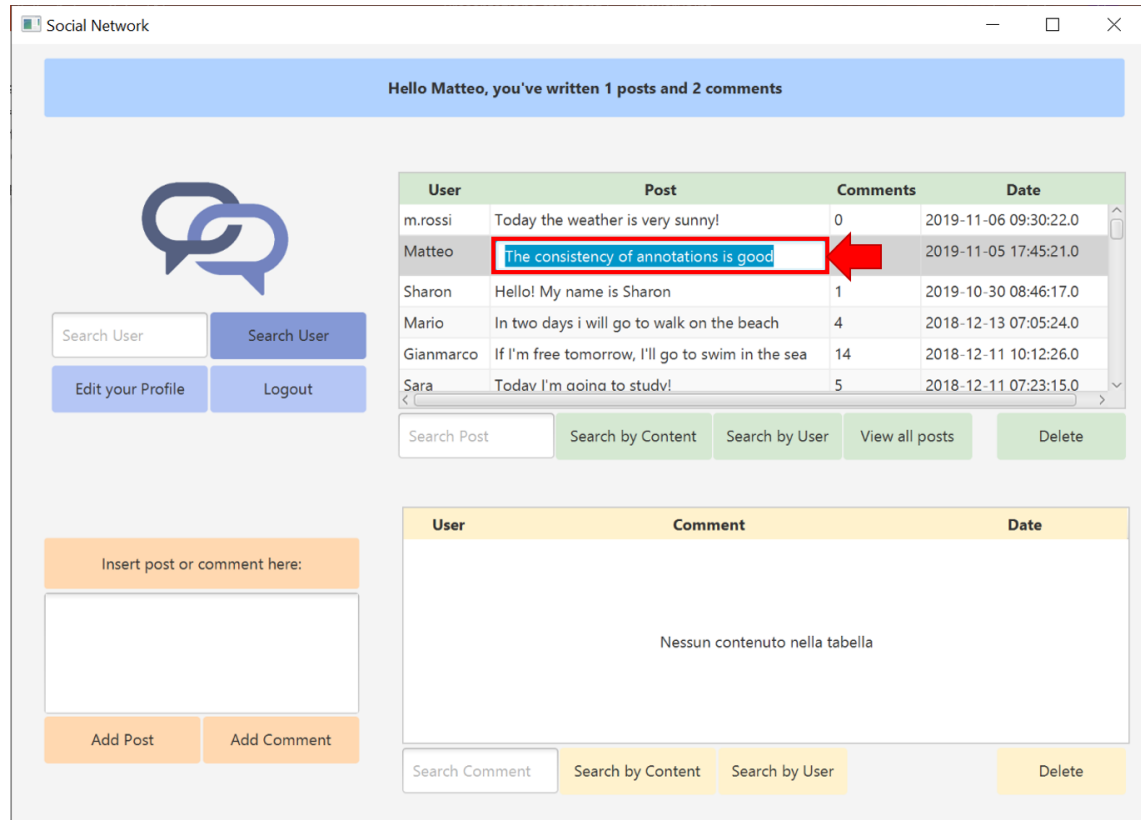


Figure 8: Modify post

4.5 Delete post

To delete a post, the user must select it and press the Delete button below the table. Also in this case the user cannot delete posts created by other users but only those created by him.

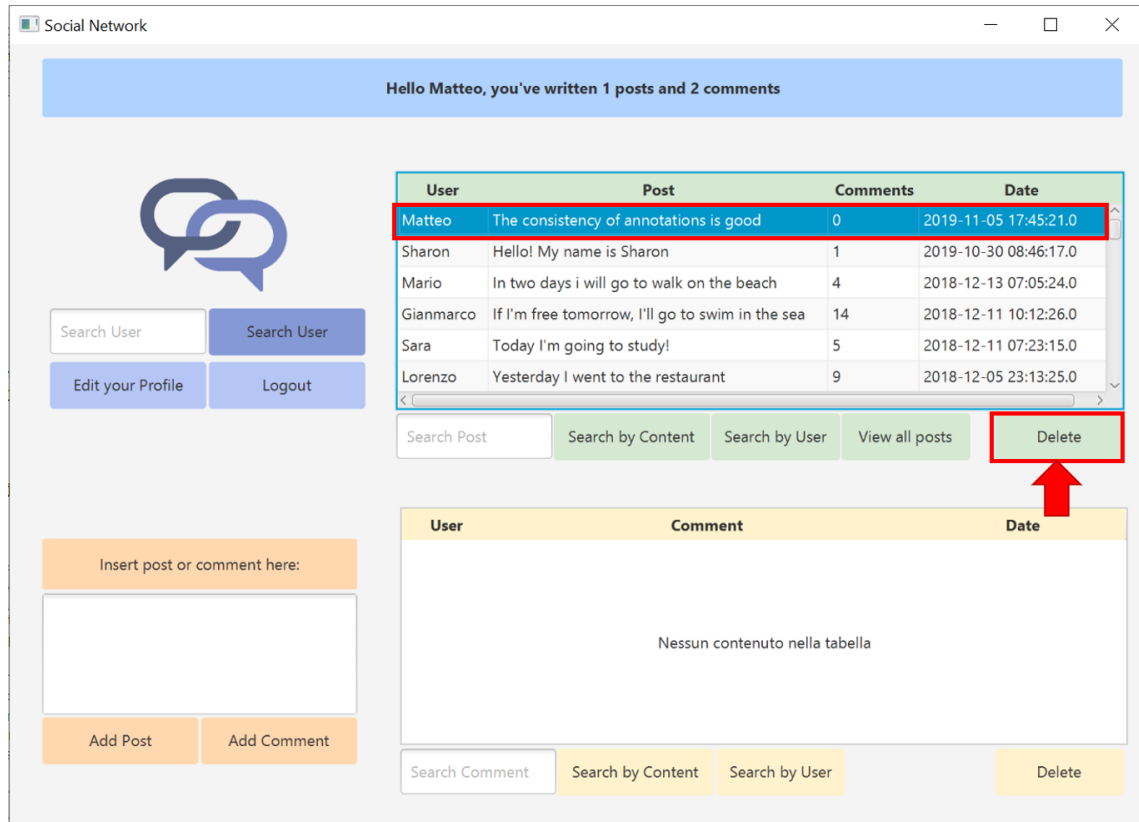


Figure 9: Delete post

4.6 Search post content

The user can search for a post by indicating a word inside the content of the post. The user must enter the word to be searched into the Search post bar and then press the **Search by Content** button.

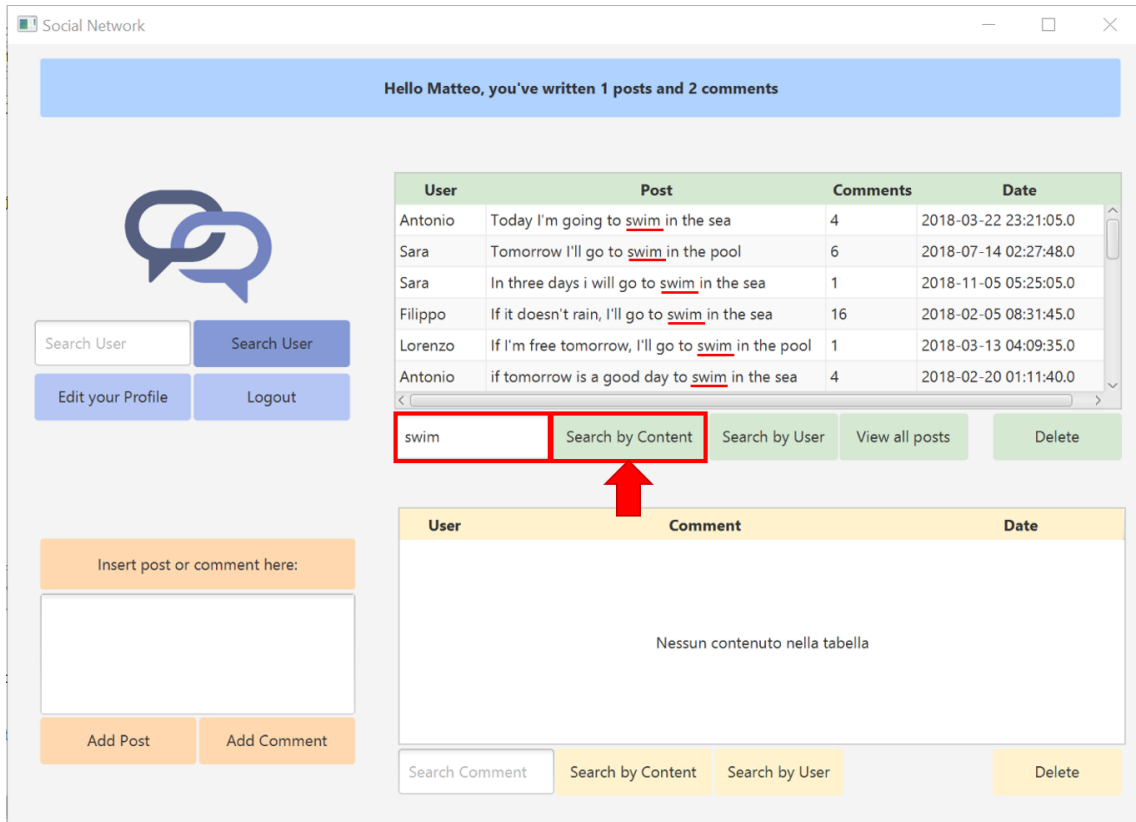


Figure 10: Search post content

4.7 Search user posts

The user can search for all posts written by a given user by indicating the username of the creator of the post. The user must enter the username into the Search post bar and then press the **Search by User** button.

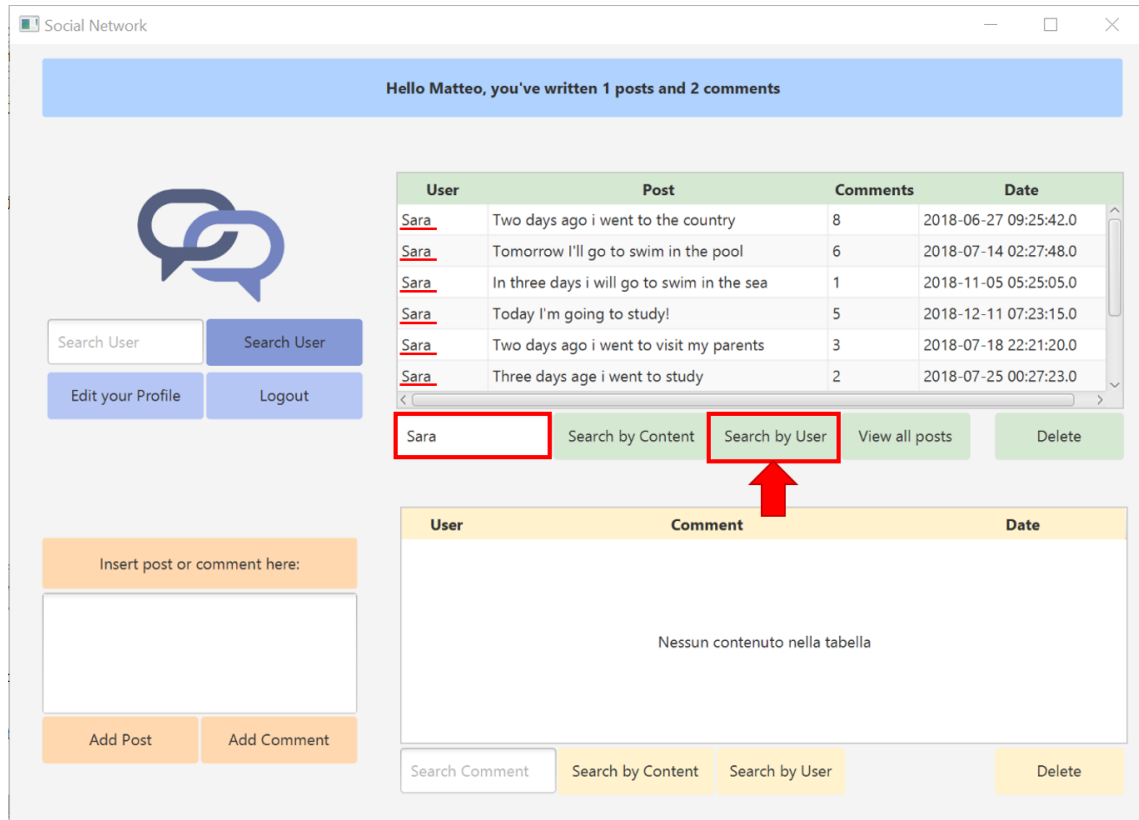


Figure 11: Search user's posts

4.8 Comment functionalities

To view all comments related to a post, the user must click on the corresponding post. The comments will be displayed within the comment table.

The screenshot displays a web application titled "Social Network". At the top, a blue banner reads "Hello Matteo, you've written 1 posts and 2 comments". Below this, on the left, is a sidebar with a speech bubble icon, a "Search User" input field with a "Search User" button, and buttons for "Edit your Profile" and "Logout". The main content area features a table of posts. The post by Gianmarco, "If I'm free tomorrow, I'll go to swim in the sea", is highlighted with a red border. A large red arrow points from this post to a second table below, which lists comments for that post. The comment table has columns for User, Comment, and Date. Below the comment table are search filters: "Search Comment", "Search by Content" (highlighted with a red box), "Search by User", and a "Delete" button (also highlighted with a red box). On the left side of the main area, there is a section for adding content with the text "Insert post or comment here:", a text input field, and "Add Post" and "Add Comment" buttons (the latter is highlighted with a red box).

User	Post	Comments	Date
Matteo	The consistency of annotations is good	0	2019-11-05 17:45:21.0
Sharon	Hello! My name is Sharon	1	2019-10-30 08:46:17.0
Mario	In two days i will go to walk on the beach	4	2018-12-13 07:05:24.0
Gianmarco	If I'm free tomorrow, I'll go to swim in the sea	14	2018-12-11 10:12:26.0
Sara	Today I'm going to ...	5	2018-12-11 07:23:15.0
Lorenzo	Yesterday I went to ... restaurant	9	2018-12-05 23:13:25.0

User	Comment	Date
Mario	Awesome!	2018-12-13 01:57:30.0
Filippo	Awesome!	2018-12-13 00:43:10.0
Tiziana	I'm not sure that's a good idea...	2018-12-12 23:30:43.0
Sara	What a beautiful idea	2018-12-12 22:18:37.0
Filippo	I'm not sure that's a good idea...	2018-12-12 21:07:04.0
Antonio	Awesome!	2018-12-12 19:57:36.0

Figure 12: Comment functionalities

All features offered for posts are also offered for comments. Thus, the user can perform the following operations: adding a new comment, editing or deleting an old comment, searching for a post by a word in the content of the comment or searching for comments written by a particular user.

4.9 Edit profile

The user, by pressing the **Edit your Profile** button, can access the information he has previously entered. To save the changes, the user must press the **Save** button.

Social Network

Hello Matteo, you've written 7 posts and 4 comments

Edit your Profile

User	Post	Comments	Date
Mario	In two days i will go to walk on the beach	4	2018-12-13 07:05:24.0
Gianmarco	If I'm free tomorrow, I'll go to swim in the sea	14	2018-12-11 10:12:26.0
Sara	Today I'm going to study!	5	2018-12-11 07:23:15.0
Lorenzo	Yesterday I went to the restaurant	9	2018-12-05 23:13:25.0
Filippo	Two days ago i went to study	12	2018-12-05 05:20:46.0
Franco	Yesterday I went to walk in the mountains	1	2018-11-27 02:18:44.0

Social Network

Here you can view or modify your info.

Back to Home page

Profile Form:

- Username (*): Matteo
- Password (*):
- Name (optional): Matteo
- Country (optional): Italy
- Surname (optional): Suffredini
- City (optional): Camaiore (LU)
- Gender (optional):
- Street Address (optional): Via Salanetti, 1
- Date of Birth (optional):
- Phone Number (optional):

Save

Figure 13: Edit profile

4.10 Search user

The user can view the profile of another user by entering the username in the Search user bar and pressing the **Search User** button. In this way, a new window will be opened containing all the information relating to the selected user.

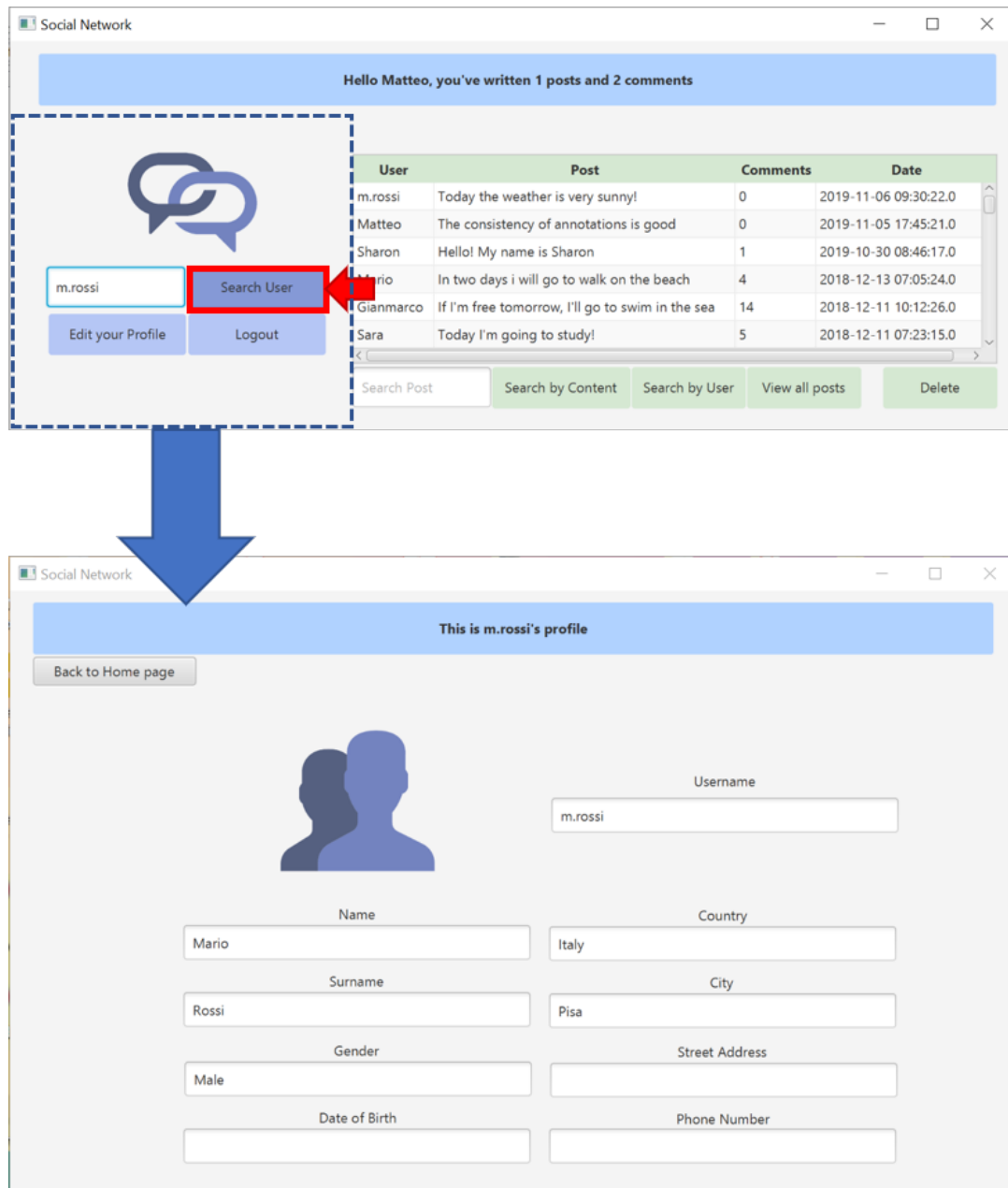


Figure 14: Search user

5 Key-Value Databases

5.1 Introduction to Key-Value Databases

A Key-Value Database is a kind of database different from the traditional relational database. In fact here tables do not exist, but there is just a sequence of key-value couples. It means that every value can be retrieved just using a key that must be unique in the bucket. This approach is pretty useful when there is no need to use features typical of relational databases, such as joining tables to retrieve some related information about some attributes. Plus, they do not provide a predefined schema nor types on attributes, that makes Key-Value Databases more flexible.

5.2 Feasibility study

In this section the project will be analyzed to decide if it should be implemented using a standard Relational Database or a Key-Value Database.

Just to remember, the project is a social network where the registered people are allowed to read posts of the other users, to type a new post, to add comments to posts, to edit their own profiles and to view other profiles.

Let's now think about an operation that involves the comments of a post: if a person wants to read comments of a post, the application has to retrieve comments considering the post and the user whom has previously commented that post.

In this example is shown how the fields are related, because it exists a relation between a post, the list of comments to that post, and the user whom has written each of that comments.

Because of that, implementing this part as Key-Value Database will be really though and hard to handle, so it is preferred to use a standard Relational Database to represent those fields.

Using a MySQL Database, all of those information can be retrieved in an easy way using join operations between different tables(Post Table, Person Table, Comment Table).

The situation is really different for the operations that involve the view/edit of a profile. Since only username and password are required, a lot of profile fields may not be set. So, in a relational database, the table representing the optional fields of a user must have a number of columns equals to the number of optional fields, but in each row, representing a person, a lot of those columns may be set to null. Plus, since it is not requested any interaction between profile fields and other attributes but the id related to the user, there is no need to use a relational database to represent those information, so in this case a Key-Value database is recommended because of flexibility.

5.3 LevelDB

In this project the key-value database used is **LevelDB**. LevelDB is a fast key-value storage library written at Google in C++ language that provides an ordered mapping from string keys to string values.[1]

Since the application is written in Java language, a Java porting of LevelDB is used.[2]

The main functions offered by LevelDB are:

- **getValue(Key):** This function will return the value associated to the Key passed as argument, if there is a match.
- **putValue(Key,Value):** This function will store in the database the couple key-value passed as argument. If a value, associated to the same key, already exists then the value passed as argument will overwrite the old one stored.
- **deleteValue(Key):** This function will delete the couple key-value related to the key passed as argument, if stored in database.

Since the key must be unique, it will be represented with a fixed pattern. In this application, the LevelDB database has to store just the optional profile fields of a certain registered person.

Each registered person has a unique id that represents him in the social network, so part of the key must include it.

The key is constructed from three different fields:

- a string that represents the entity
- the unique id mentioned before
- the attribute

These three fields are linked using a special character “ : ”. In particular, in this case, the first two fields of the key will represent the unique person, while the last one will represent an optional profile field.

For instance, let's suppose that a person with id equals to 1 has filled some of the optional profile fields (name and surname for example), LevelDB will store two different couples of key-value: one with key **username:1:name** and the other with key **username:1:surname**.

References

- [1] Google. *LevelDB*. <https://github.com/google/leveldb>. 2014.
- [2] Dain. *Porting Java of LevelDB*. <https://github.com/dain/leveldb>. 2019.