ASSIGNMENT COVERSHEET

PRAGUECITY UNIVERSITY

Student Name: Sebila Doubaeva	
Class: Java Programming (ESME)	
Assignment: Task 3	
Lecturer: Viktor Černý	Semester: 2104
Due Date: 24.01.2022	Actual Submission Date: 24.01.2022

Evidence Produced (List separate items)	**Location (Choose one)**	
	X	Uploaded to the Learning Center (Moodle)
		Submitted to reception
Note: Email submissions to the lecturer are not valid.		

Student Declaration:	
I declare that the work contained in this assignment was researched and prepared by me, except where acknowledgement of sources is made. I understand that the college can and will test any work submitted by me for plagiarism. **Note:** The attachment of this statement on any electronically submitted assignments will be deemed to have the same authority as a signed statement	
Date: January 24, 2022	Student Signature: Sebila Doubaeva

A separate feedback sheet will be returned to you after your work has been graded.
Refer to your Student Manual for the Appeals Procedure if you have concerns about the grading decision.

Student Comment (Optional)
Was the task clear? If not, how could it be improved?
Was there sufficient time to complete the task? If not, how much time should be allowed?
Did you need additional assistance with the assignment?
Was the lecturer able to help you?
Were there sufficient resources available?
How could the assignment be improved?

```
            Set<String> participants = new LinkedHashSet<>(nameList);
            String date = new SimpleDateFormat( pattern: "dd/MM/yyyy").format(Calendar.getInstance().getTime());

            currentConversation = new Conversation(messages, participants ,path, name, date, messages.size(), favorite:
            recentConversations.add(currentConversation); // adding this conversation in the list
            pathText.setText("Path: " + currentConversation.getPath());
            updateTab(recentTab);  // that will be updated fo the tabs
            updateTab(allTab);
        } else showExceptionDialog( option: 4); // return dialog message with detected error
    }
}

/**
 * This method used for generating the messages. The resulting message text will be
 * split into subtexts due to the pattern obtained from the iteration of the list
 * of emotion objects. If the previous name of the sender of the message is equal
 * to the current one, then the textFlow with the name and the profile Pane will
 * not be displayed.
 *
 * @param message message from message object
 * @param name name from message object
```

(excerpt from the project)

Task 3

Sebila Doubaeva

January 24, 2022

1 Abstract

This report consists of an evaluation of the final project, a comparison and description of design changes compared to the original design, a description of the testing process and its results, the user documentation, as well as possible improvements of this project.

Contents

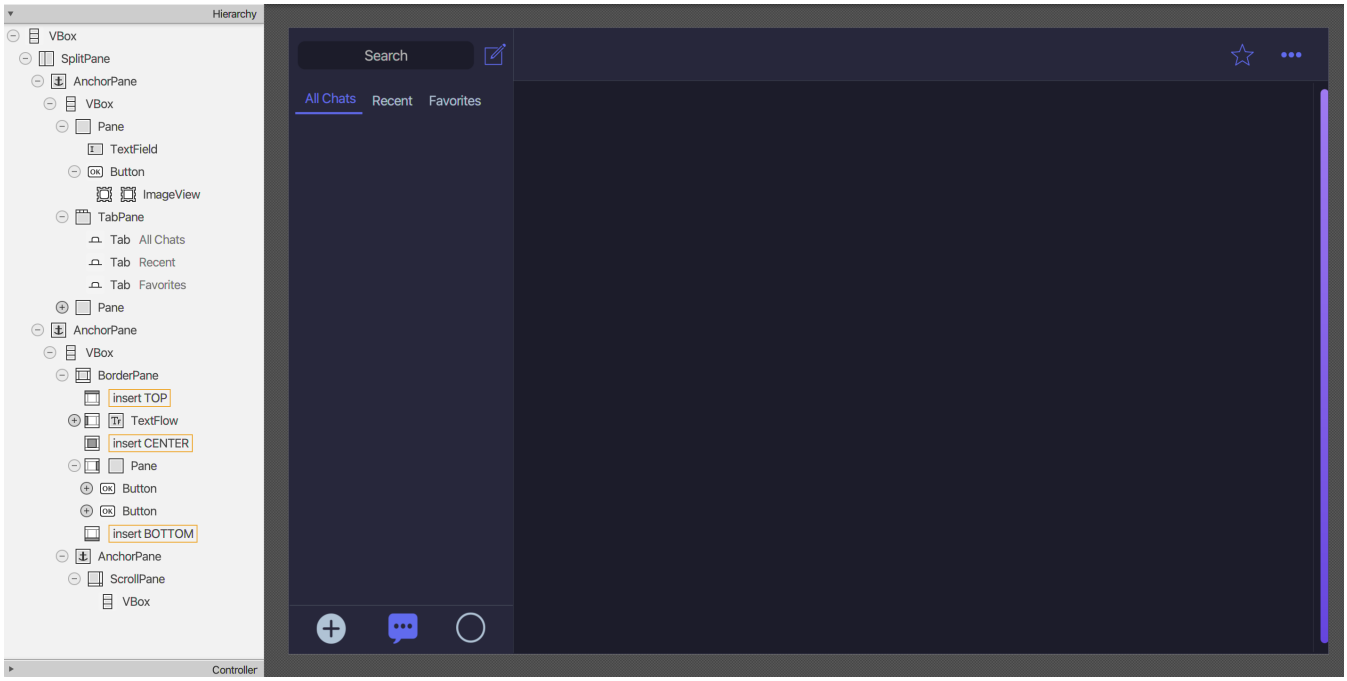
List of Figures

2 Changes from The Original Design

As the coding time increased, the original design of the project receded more and more into the background. Several factors contributed to this. The first factor was the elementary ignorance of Java FX: the existence of some components was recognized only by chance, when searching for adaptations of completely different methods, for example, a vertical box was replaced with a list, and a simple panel with a more suitable border pane. The second factor was that the understanding of the scale of the work ahead came only as the code and methods were written, for example, there was a problem of saving data after closing the application.

For a more convenient perception of the subsequent descriptions of changes, look at Figure1.

Figure 1: View from the Scene Builder

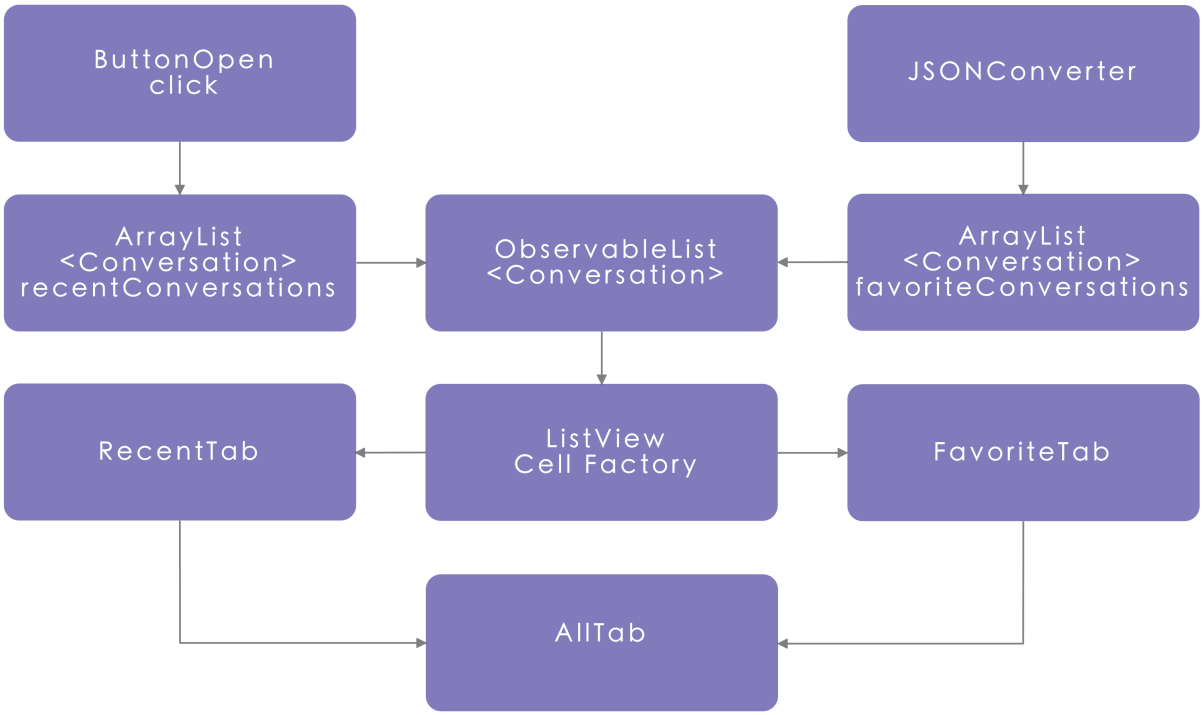


The most obvious change is the use of the SplitPane component to resize the chat more conveniently. However, this change entailed the need to replace the upper panel containing the path to the BorderPane, since when dragging to the left, all the components of the panel (text with the path, buttons) also shifted.

The second important change was the abandonment of the folder system. Opening secondary windows too often did not seem to be the most convenient solution for the user, so it was decided to replace them with a TabPane containing three tabs : FavoriteTab, recentTab and AllChatsTab.

The logic of filling these tabs is indicated in Figure 2, from which it can be seen that each conversation passes through a CellFactory, in which each element is given shape and order before being placed in the ListView. Further, this choice made easy to get information about the selected conversation, therefore, to display the chat without going through the file selection dialog again.
Considering that if there is a list, it follows that it is possible and even necessary to add a search system, which makes it easy to search for the necessary file. Therefore, a textField has been added, writing the file name will display the appropriate items in the list.

Figure 2: The logic of creating Tabs



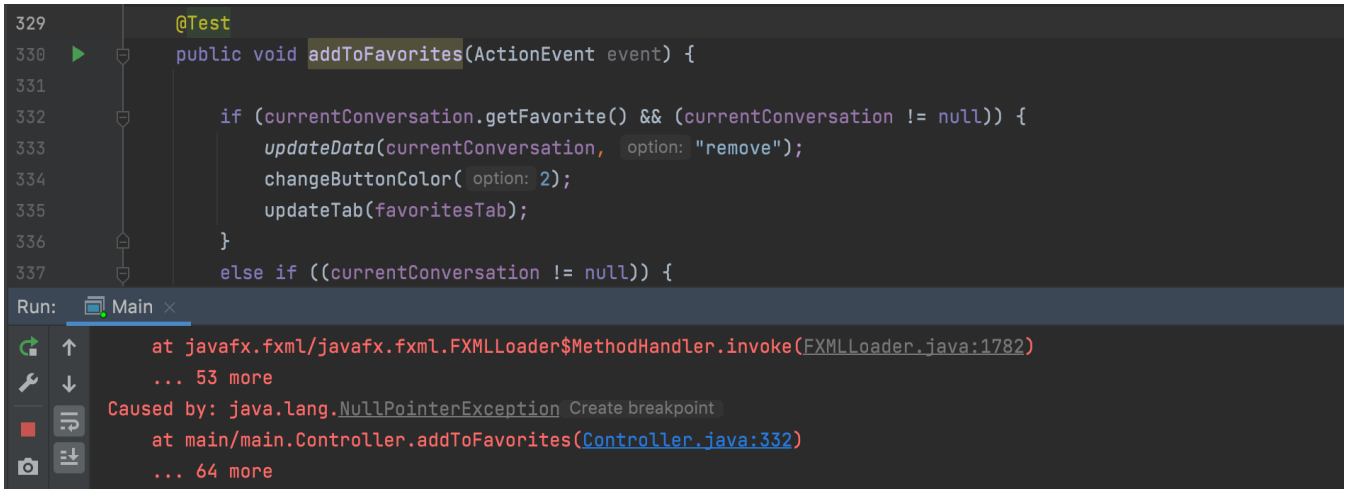
The graph also shows the presence of a JSON file converter. The fact is that there was a need to save chats to favorites. For recentTab it was possible to use a simple list without storing it for a long time, but with favorites it was important not to lose data after closing the application so that this function does not lose its original sense. Therefore, the choice fell on the most understandable JSON format, and with the help of the GSON tool, it was possible to convert an object to a string and from a string back to an object. Probably one of the simplest logical solutions in the project.

Otherwise, there are no significant differences with the initial design, apart from replacing the GridPane with a BorderPane for displaying messages.

3 Testing Process

The application testing part is one of the most important parts of the project, perhaps even the most. In any case, writing this application, the testing process took the same amount of time as writing the code itself.

Figure 3: The testing process in IntelliJ



This part of the project can be divided into four categories :

- Unit Testing : testing various methods of the class, propably the most used.
- Integration Testing : testing of a group of interacting modules (module main and data).
- System Testing : performance and functioning of an entire application (the least used).
- Usability Testing : performance and functioning of the user interface.

Basically, the most of testing part was based on Unit tests. This allows to quickly check whether the next code change has led to regression, to the appearance of errors in the already tested places of the program, and also facilitates the detection and elimination of such errors. For example, testing that one or another method works correctly when an event is committed (pressing a button).

In contrast to Unit, Integration Testing had a larger scale and was used to check the correct connecting and development of methods working with the database. So, for example, it was necessary to check that clicking on the add to favorites button really translated and saved the received data to a JSON file. And in the same way, deleted correctly the data from file and displayed the result in the main module.

One of the most frustrating in terms of time parts was System Testing. However, it made it

possible to identify unforeseen scenarios. For example, on the Figure 3, if the add to favorites button was pressed, and the conversation is not open, then it gives a Null Pointer error, easily correctable, but not foreseen.

The last part of the tests went to GUI tests. It was important to make sure that all the components are clear and easy to use, as well as having a friendly basis.

After going through the main types of testing, I would like to add my own: reading. Even after applying all the testing methods, it is possible not to notice the error that was written accidentally. For example, in my case, the messages were not displayed when all the tests showed that the program was functioning correctly. The fault turned out to be the scroll pane, the visibility of which was disabled during the test of the application theme switching. And everything really worked correctly, it just wasn't visible! The mistake was incredibly stupid, but finding it cost an unbelievable amount of time and effort. Therefore, I will definitely use this method in the future.

4 Possible Improvements

Despite significant modifications compared to the original design, the application can still be improved and have more functionalities:

- Bring the GIF image replacement system to automatism by inviting the user to add patterns values himself and save this data to the database.
- Put messages in a ListView to be able to search by word.
- Improve the file parsing method, making it more flexible.
- Create parsing for formats other than .msg and .txt.
- Find solutions to speed up the application.
- Add the possibility to increase or decrease the size of the text of conversation.
- Add checks on the added file (do not add the same file again).

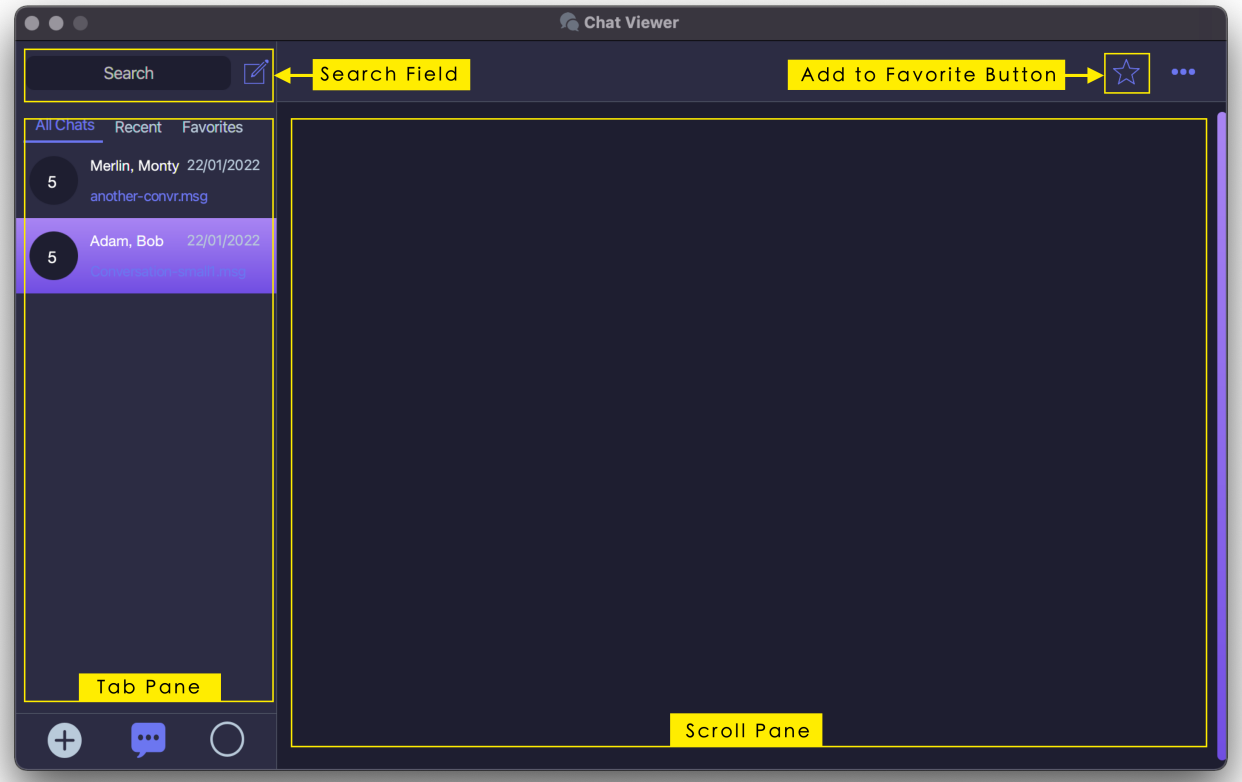
5 The User Documentation

The application contains the following components with the corresponding functions :

- Search field to find conversation by file name
- TablePane with three chats tabs : all chats, recent and favorites.
- Add to favorites Button.
- Scroll Pane for display conversation.
- The Button to change the current application theme.
- Split Pane Separator for resize chat window.
- Open Button to open a file select dialog.
- Path Text Flow to display file path.

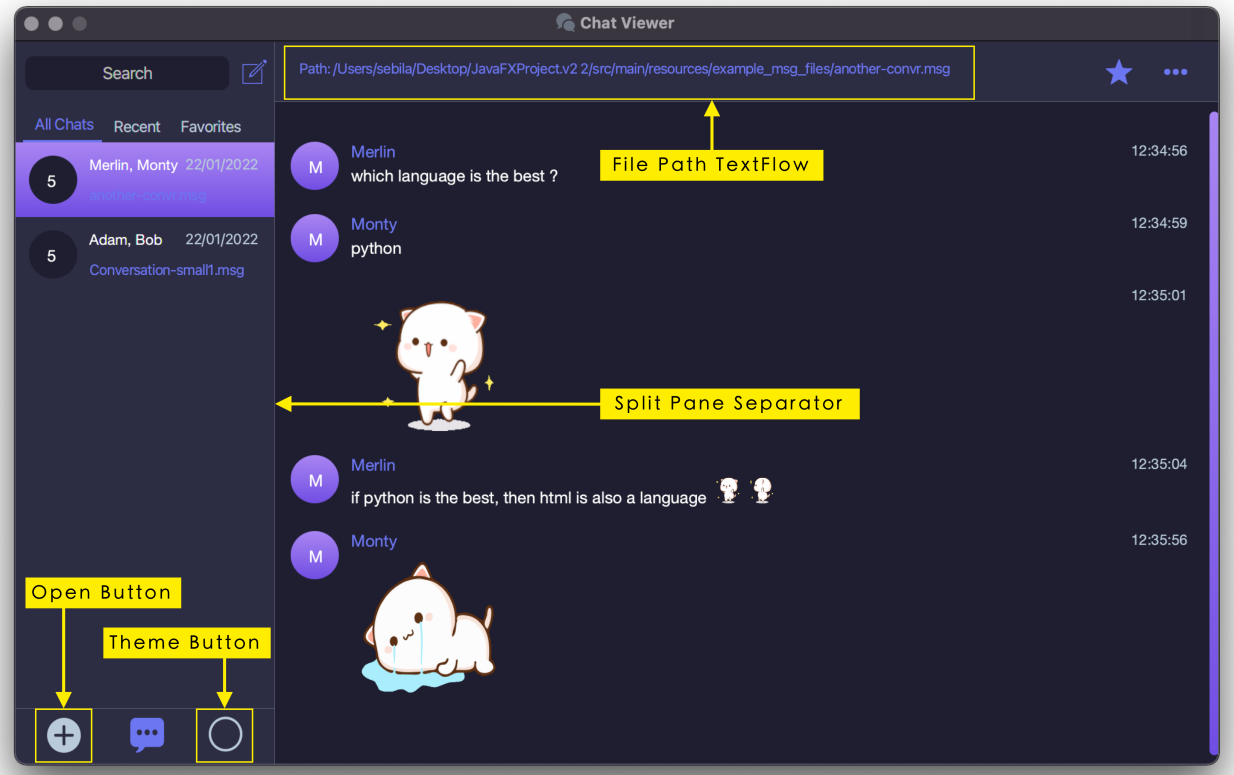
At the beginning of working with the application, open a new file by clicking the open button, and then select the file in the dialog window. In the case of correct file format (.txt or .msg format) and absence of any damage, the conversation will appear in the Scroll Pane, otherwise, the user will receive a message with a corresponding error.

Figure 4: Vision of the program at file opening



As soon as the conversation is shown, the related information will be displayed on the left side in the recent and all chat tabs, containing the file path, the names of the interlocutors, the number of messages, as well as the date of the addition of the file.

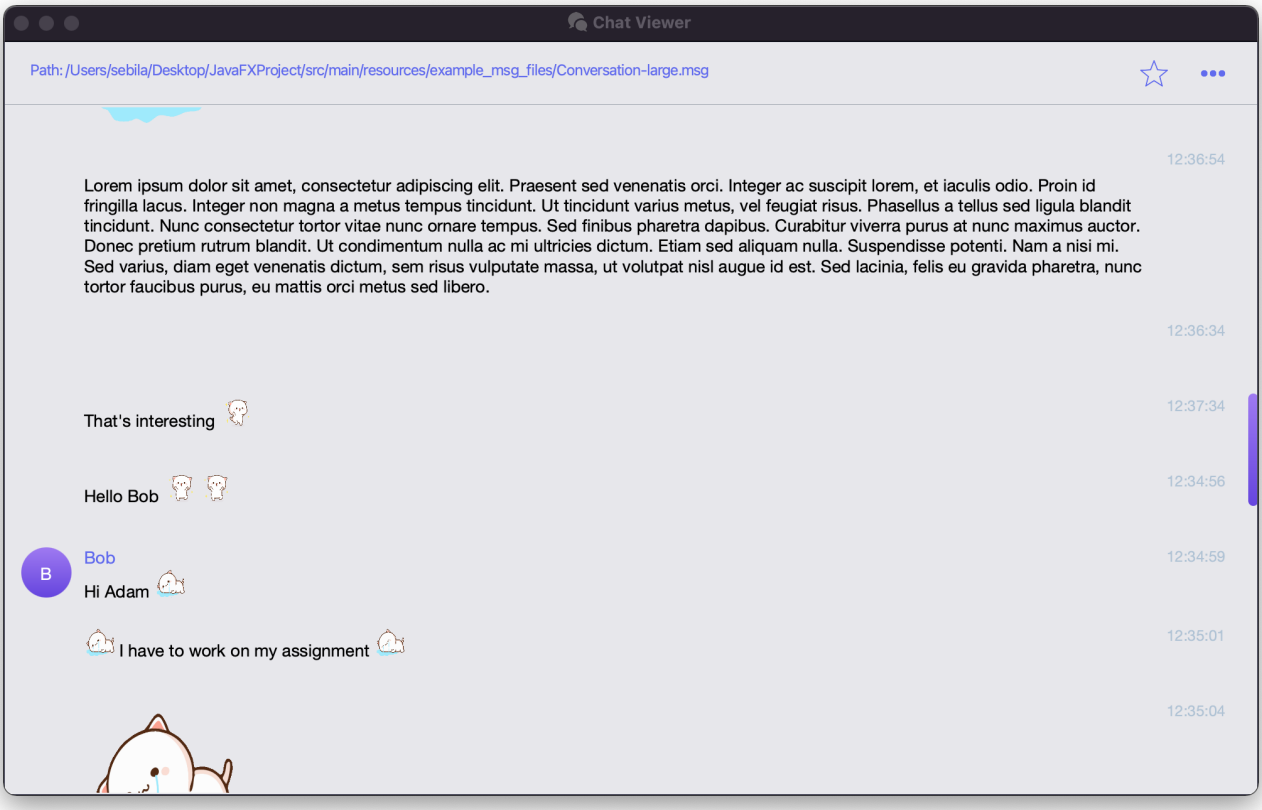
Figure 5: Vision of the program at startup



Clicking the add to favorites button will add the current conversation to the favorites tab. All the added ones will be saved and displayed in this tab on subsequent launches of the application. If the conversation is already in favorites and the button is pressed, it will delete this conversation from the data.

The theme's button allows user to change the color scheme of the application to a light theme. Clicking again will change the theme to dark. Look at Figure 6.

Figure 6: Light theme



Annex

Figure 7: Current Class Diagram