The Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

VAVA

Project Documentation

MessCode

By Noemi Farkas
Oliver Izsák
Kamilla Kisová
Roman Korzhov
Martin Šváb

Teacher: Mgr. Ing. Miroslav Reiter, MBA

Year: 2021/2022
Date: April 26th, 2022

Table of Contents

1 Project Vision

1.1 Vision

The application's goal is to make a private and effective communication system for a company. It will help with making the employers' and employees' communication more confidential and organised. By making it more organised, problem-solving will improve, and distributing work-related tasks will be easier.

1.2 Usage Scenario

Lidli is a medium company that provides app development for customers. The company has an Owner that actively manages the company and its employees, 3 Project Managers, and a dozen Developers. They all are included in different projects and are focusing on improving their communication to prevent misunderstandings and accelerate development.

Up until now, they have used Facebook's Messenger both for individual and group communication. The Owner fears that as the company grows, Messenger's options and features might not be enough anymore. He is also growingly concerned about privacy issues and would like to move to a more private, customised app to communicate on. For a company that started working with important data, it is in its best interest to move to a safe environment.

1.3 Know your users

★ Employer

The Owner is always striving to help out the Project Managers and the Developers wherever he can. He has to have a form of contact with every employee but does not like the current logistics of Messenger. He is always worried about forgetting to add a new employee on Messenger. He is also reluctant to move to other communication/chat apps since most of them have a monthly-based payment system for additional features. Safety and privacy are issues as well that he wants to address, as he is planning to expand the company. Most of the time he is away from the office, but he makes sure to hop in for at least an hour every day to sort out offline things. Other times, his laptop is always by his side and spends a couple of hours writing with his employees. As he likes to keep track of the projects' progress, he is included in all of the group chats. In case he notices some kind of problem or misunderstanding in a project he is able to immediately inform the Project Manager in charge.

★ Project Manager

Project Managers are focused on their assigned projects and teams. Once the Owner assigns a project to them, they invite colleagues to a group chat where most of the communication is going down. Currently, there are 3 Project Managers. They use MessCode for assigning respective tasks, informing & navigate team members and solving misunderstandings. This takes up a couple of hours from every workday. Throughout the different points of the project, they make sure to report to the owner and ask about additional tasks that might have come up.

★ Employee

Employees are responsible for the realisation of the project. This group of people needs to know exactly what is happening on the project at any time. They spend most of their time in the office. They spend an hour or two each workday on the app to sort out project tasks. Whenever they are ready with their tasks, they report to their respective Project Managers.

1.4 User's language

★ **Chat**: an online room for texting

Both individual and group chats are very important. Employees can contact each other through simple chats anytime, without the need to “add” people first. Group chats are created by the Employer, where the Project Managers can add respective Employees there.

★ **Project**: a group of tasks to achieve a goal

The project has a handful of tasks associated with it, with a strict end goal in sight. It also has a deadline connected to it. Finished projects mean that respective group chats are no longer needed for it. Each project has a Project Manager and a couple of Employees assigned to it. Project Managers are usually assigned 1-2 projects at a time, but Employees can be assigned to any number of projects, depending on their field of work.

★ **Team**: a group of employees with a common project

Every team is assigned a single project. Whenever a project comes up, the Employer assigns the project to a Project Manager. The Project Manager then picks out the Employees needed for the tasks, thus creating the team.

1.5 Most common tasks

As an Employer, the owner wants to:

- ★ Constantly monitor projects and their group chats
- ★ Register new group chats for projects
- ★ Consult with Project Managers and Employees privately
- ★ Quickly register new Project Managers to the app

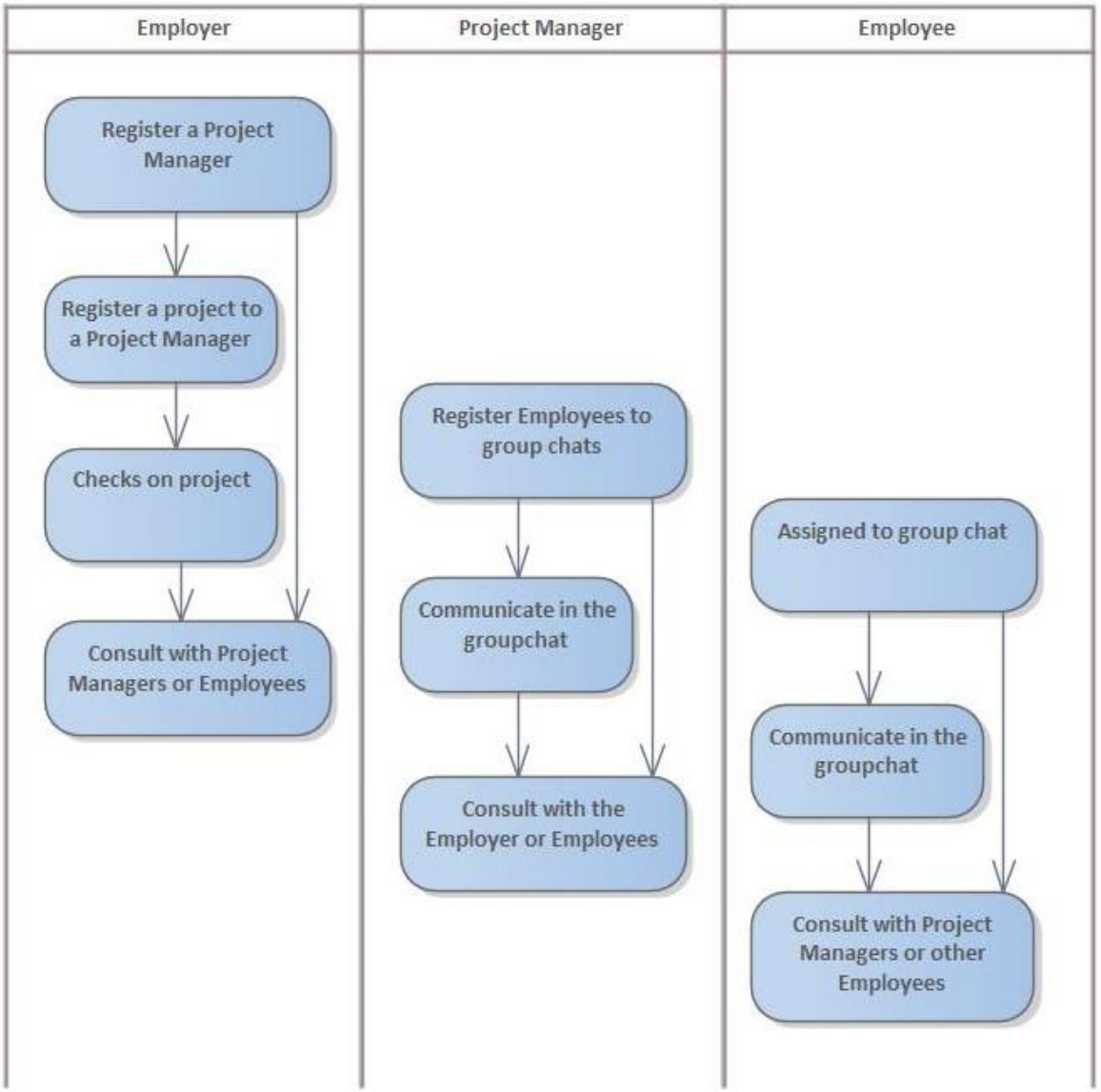
As Project Managers, they want to:

- ★ Constantly monitor the projects they are assigned to
- ★ Add new Employees to group chats
- ★ Report to the Employer privately
- ★ Consult with Employees privately

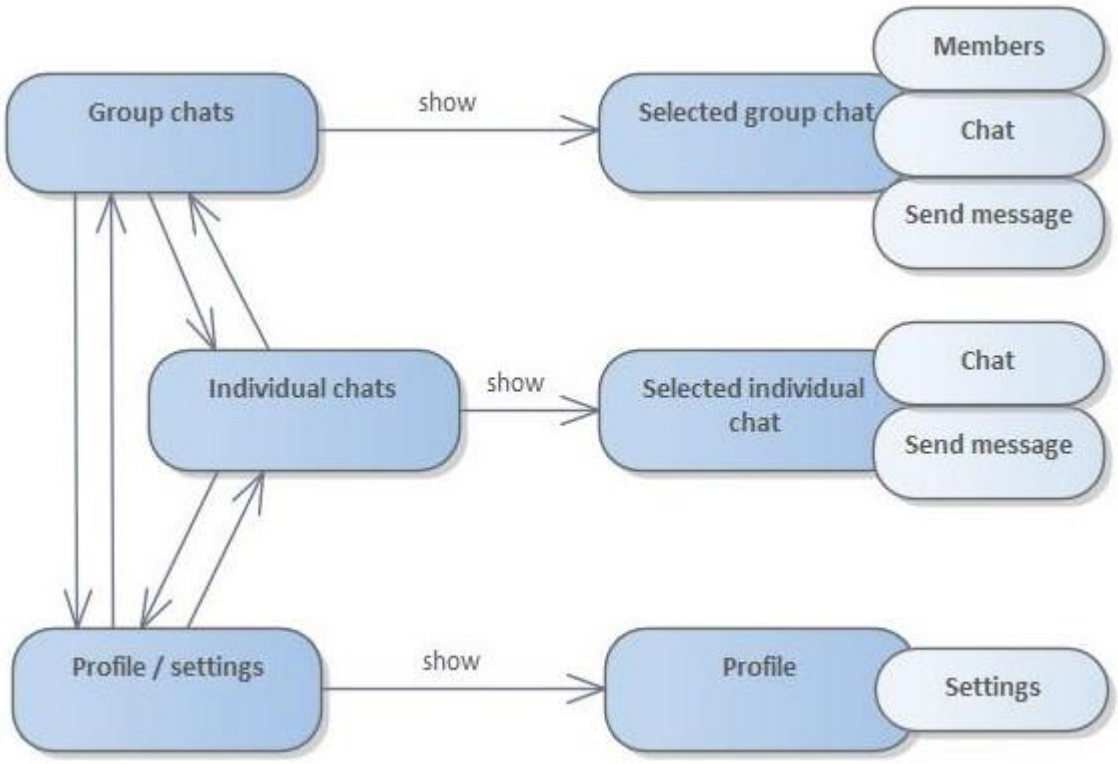
As Employees, the developers want to:

- ★ Access group chats for their group projects
- ★ Report to the Employer or their Project Managers privately
- ★ Consult tasks with colleagues

1.6 Main Processes

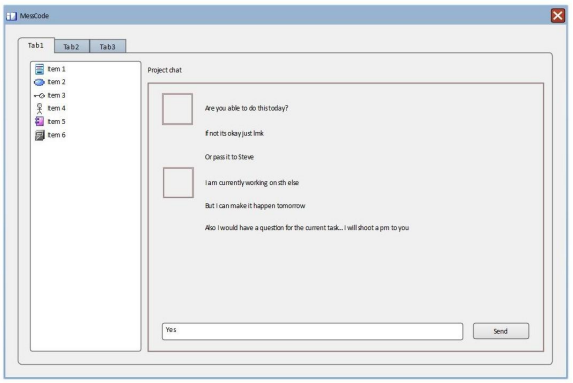
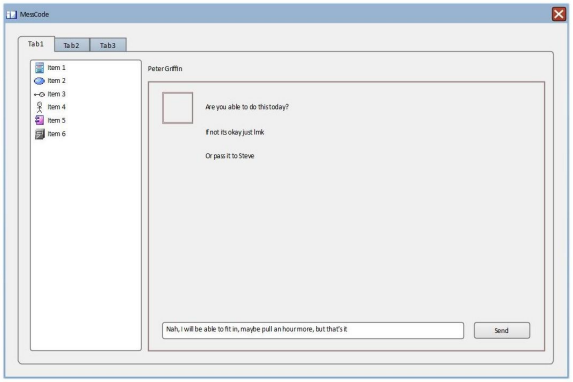
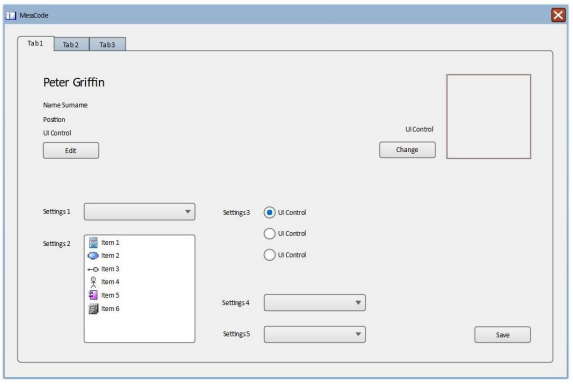


1.7 Navigation



At the start of the app, the customer would see the group chats. The group chats, individual chats, and the profile/settings are available as tabs. After clicking on respective tabs, they see possibilities associated with the current tab.

1.8 Mockups

<table>
<tr><td colspan="2">Group chats tab</td></tr>
<tr><td> 
</td><td>- List of group chats on the left
- Selected group chat of project
- Message sending</td></tr>
<tr><td colspan="2">Individual chats tab</td></tr>
<tr><td> 
</td><td>- List of chats with individuals
- Selected individual chat
- Message sending</td></tr>
<tr><td colspan="2">Profile / Settings tab</td></tr>
<tr><td> 
</td><td>- Profile data
- Profile settings</td></tr>
</table>

1.9 Non-functional requirements

1. The system will be implemented in Java.
2. The database will be implemented in PostgreSQL, with JDBC to connect to the system.
3. The system will be a client-server system.
4. Messages sent by users must have a latency below 500ms.
5. All messages will be encrypted.
6. The application will have localization for both English and Slovak-speaking workers.
7. Database will be SQL-Injection proof
8. The application will have an easy and understandable interface.
9. Any errors will be logged.
10. The program will take up a maximum of 1 GB of memory (Including logs and locally saved configurations).
11. The project will be developed in IntelliJ IDEA, in Maven project using Java version 11.0.

2 Project Documentation

On the basis of the created vision document all the system and its functionalities were implemented. Further functionalities were added to the original design, which will be described in this section.

The new parts of the system have been implemented in order to meet the requirements of the system. The added functionalities of the system include, first of all, the possibility to change the user's password and create new group chats in the system. Admins can now register a new user account, setting the employee's name and email accordingly. Also, a dark theme was implemented to increase the user's comfort using the application interface.

This section summarises all the functionality of the system, with the more complex ones being briefly described.

2.1 Superuser's functionalities

The superuser role was created for the system administrator. He can create new user accounts, remove user accounts. He can also create projects, change group leaders of existing groups or delete groups. He has the privilege to reset someone's password which he has to send to the owner of that account outside of this application.

List of superuser's key functionalities:

- Adding new users such as superuser, employers or employees to the system.
- Adding new group chats, when starting new projects.
- Adding all kinds of users to group chats.
- Read and monitor all chats in the system, except for private messages and group.
- Communicate with anyone by private messages.
- To edit their account information, change the personal password.
- Deleting all kinds of users from the system.
- Resetting the password for any user.

2.2 Employer's functionalities

For the employer, compared to the vision document, the possibility has been added to change his own password in the settings tab interface. Also, the administrator can now create any group chats by adding any employees to the application.

When adding new users to the system, the employer has to enter their credentials, which are the user's email and first & last name. A random password is generated for the added user

through the system, which is displayed and properly hashed in-app. These passwords are randomly generated and the generated password can be read by users who need to change it.

List of employer's key functionalities:

- Adding new users as superuser, employers or employees to the system.
- Adding new group chats, when starting new projects.
- Adding all kinds of users to group chats.
- Read and monitor all chats in the system, except for private messages and group.
- Communicate with project managers or workers by private messages.
- To edit their account information, change the personal password.
- Deleting all kinds of users from the system.

2.3 Project Leader's functionalities

An employee becomes a Project Leader when he is assigned to a group by an employee or superuser. Project Leaders can add and remove users from the group. The project leader also has the ability to create group chats for effective management of the current project and to add people working on the project to those group chats.

List of project leader's key functionalities:

- Adding new users with role employees to the system.
- Adding new project group chats.
- Add employees to group chats.
- Read and reply to all chats related to the assigned project.
- Communicate with users using private messages.
- To edit their account information, change the personal password.
- Remove users with role employees from the group..

2.4 Employee's functionalities

The ability to change the password has been added to the user account settings options user account settings, as it exists in standard systems. This is mainly due to the fact that when new users are added, the system generates a password that is initially also available to the administrator, whose job is to deliver the password securely to the users.

List of employee’s key functionalities:

- Read and reply to all chats related to the assigned project.
- Communicate with users using private messages.
- To edit their account information, change the personal password.
- Work, work, work, work, work :)

2.4 Project Requirements

2.4.1 GUI

JavaFX technology was used for the project graphical interface. The main architectural pattern used was the Model–View–ViewModel.

Model-View-ViewModel (MVVM) is a software architectural pattern that separates the development of the graphical user interface (the *view*) from the development of the business logic or back-end logic (the *model*). The viewmodel acts as the element between the view and the model, it is a value converter, meaning it is responsible for converting data objects from the model in such a way that objects are easily managed and presented. The viewmodel handles most of the view's display logic.

Some say that the Model-View-ViewModel is a variation of the Model-View-Controller architectural pattern that is tailored for modern UI development, where the View is the responsibility of a designer rather than the developer's. However, MVVM and MVC have a couple differences. While MVC facilitates the separation of the application into three logical components: Model, View and Controller, MVVM on the other hand strives for the separation of the GUI from the business logic. In MVC, the controller is the entry point to the application, while in MVVM, the view is the entry point. The MVC model component can be tested separately from the user, while MVVM is easy for separate unit testing, and the code is event-driven. MVC architecture has "one to many" relationships between Controller & View while in MVVM architecture, "one to many" relationships are found between the View & ViewModel.

2.4.2 Security

Password encryption:

Our passwords need to be stored in our database. Before they are stored they are appended with salt. Our salt is a randomly generated 10 character long string. After they are appended they are hashed using the SHA-512 hashing algorithm. This way even with access to our database there is no way of retrieving the original password from our database. Because we are appending our passwords with salt even when multiple users are using the same password they have different hash values. Unless their salts are also the same which is very unlikely. When a user tries to log into an account their entered password is appended with this account's salt retrieved from the database and then it is hashed and compared to the hashed password stored in the database. If they are the same the user entered the correct password and login is successful, otherwise login is unsuccessful.

SQL injection defense:

To prevent users from injecting sql queries when providing input we always use parameterized queries when interacting with the database in our project. This automatically

turns all special symbols into regular characters. For example, input “peter‘ OR 1=1;” gets turned into “peter\’ OR 1=1;”.

2.4.3 Logging

For the logging we used log4j2 which is configured with an xml file. We are logging information and errors both server and client side. Also after 30 days the logs are automatically deleted.

2.4.4 Database

Since we have a client-server application, the structure of the application needed to be divided into two parts. The client side deals with all the information coming from the GUI. From then on, the client forwards these to the server, where the server communicates the respective transactions down with the database. In the case of both successful and unsuccessful transactions (correct / incorrect login) the server responds to the client, from where it's the client’s responsibility to forward the response to the GUI for processing and filtration (model and viewmodel), and presenting it (view).

Handling the database is done through JDBC which is an API for Java for database interaction. The JDBC queries are split into two classes, the ImportData which deals with insert and updates, and the ExportData which deals with selects.

2.4.5 Localization

Localization has been done through ResourceBundles. In this bundle we can see two resources: *bundle_en.properties* and *bundle_sk.properties.* These two files are responsible for storing the two languages’ locales - our chosen ones were English and Slovak. From there on, the strings are read from these two files for the GUI.

When the application starts for the first time, the user is asked to select a language, after that user can change it in the application settings.

2.4.6 XML

We are using an xml file for the logging configuration and also for saving the local application settings such as localization language, chat message custom colour, dark/light mode locally in XML file. The application can change XML file variables if the user configures it in the settings.

2.4.7 Regex

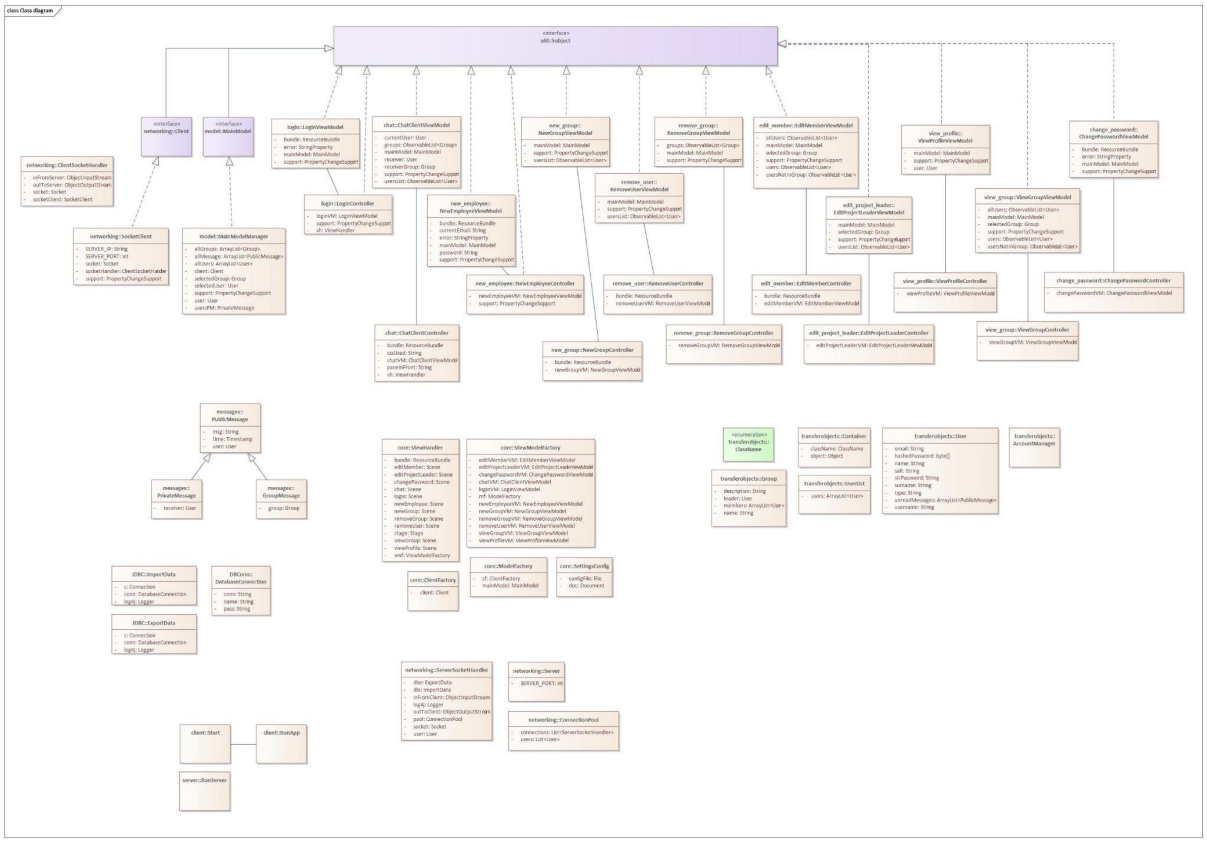
For checking the correct input for email, name, password, and the group name we used several regexes during our project. Also the other inputs are limited so these were all the places we could implement them.

2.4.8 Encapsulation

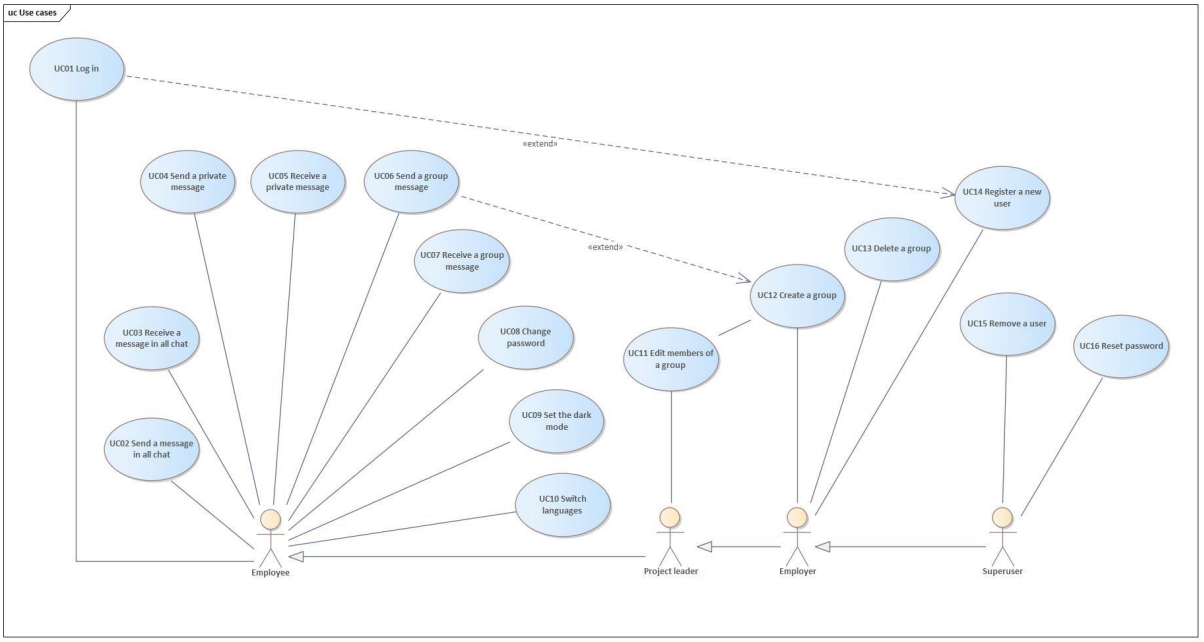
We strictly followed encapsulation and made sure that all the variables and methods are carefully encapsulated as they are supposed to.

3 UML Diagrams

3.1 Class diagram



3.2 Use case diagram and use cases



Disclaimer: Since the employee acts as a basic user, the terms “employee” and “user” will be used interchangeably. Other types of users, like the project leader, the employer and the superuser inherit all the functionalities of their lower rank counterparts.

UC01 Log in

Name: Log in

ID: UC01

Brief description: The users are able to log in and authenticate themselves using their email and password. Through logging in, they are able to use other functionalities.

Actor: Employee

Input conditions: The user has been registered by the superuser (see UC14)

Output conditions: The user has successfully logged in

Main scenario:

1. The employee opens the application
2. The system presents the employee with the login screen
3. The employee enters their information (email and password)
4. The system checks the provided information against the database, sees that the user is authorised to step in and opens the main chat screen for the employee

UC02 Send a message in all chat

Name: Send a message in all chat

ID: UC02

Brief description: The users are able to send and receive various types of messages, like messages in the all chat. Anybody can send a message here, messages sent here can be viewed by anyone who logs into the application.

Actor: Employee

Input conditions: The user has logged in

Output conditions: The user sends a message for everyone in the all chat

Main scenario:

1. The system presents the user with the main chat window
2. The user types in their message and sends it
3. The system processes the message and forwards it to the database

UC03 Receive a message in all chat

Name: Receive a message in all chat

ID: UC03

Brief description: The users are able to send and receive various types of messages, like messages in the all chat. Any message that belongs to the all chat is received by every authorised person.

Actor: Employee

Input conditions: The user has logged in

Output conditions: The user sees the message appear in the all chat

Main scenario:

1. The system presents the user with the main chat window
2. The user sees the message in the all chat

UC04 Send a private message

Name: Send a private message

ID: UC04

Brief description: The users are able to send and receive various types of messages, like private messages. Private messages are only displayed to the two users (sender and receiver) who belong to them.

Actor: Employee

Input conditions: The user has logged in

Output conditions: The user sends a private message

Main scenario:

1. The system presents the user with the main chat window
2. The user selects the private message tab
3. The system presents the user with the private message tab
4. The user selects the person who they wish to write a PM for
5. The system presents the user with the private messages between those two people
6. The user writes the message
7. The system processes the message and forwards it

UC05 Receive a private message

Name: Receive a private message

ID: UC05

Brief description: The users are able to send and receive various types of messages, like private messages. Private messages are only displayed to the two users (sender and receiver) who belong to them.

Actor: Employee

Input conditions: The user has logged in

Output conditions: The user sees the private message

Main scenario:

1. The system presents the user with the main chat window
2. The user selects the private message tab
3. The system presents the user with the private message tab
4. The user selects the person who they wish to read PMs from
5. The system presents the user with the private messages between those two people
6. The user views the messages

UC06 Send a group message

Name: Send a group message

ID: UC06

Brief description: The users are able to send and receive various types of messages, like group messages. Group messages are only displayed to the users who are part of the group.

Actor: Employee

Input conditions: The user has logged in and belongs to a group

Output conditions: The user sends a group message

Main scenario:

1. The system presents the user with the main chat window
2. The user selects the group message tab
3. The system presents the user with the group message tab
4. The user selects the group who they wish to send GMs to
5. The system presents the user with the group messages of that group
6. The user writes and sends the message
7. The system processes the message and forwards it

UC07 Receive a group message

Name: Receive a group message

ID: UC07

Brief description: The users are able to send and receive various types of messages, like group messages. Group messages are only displayed to the users who are part of the group.

Actor: Employee

Input conditions: The user has logged in and belongs to a group

Output conditions: The user sees the group message

Main scenario:

1. The system presents the user with the main chat window
2. The user selects the group message tab
3. The system presents the user with the group message tab
4. The user selects the group who they wish to read GMs from
5. The system presents the user with the group messages of that group
6. The user sees the messages of the group

UC08 Change password

Name: Change password

ID: UC08

Brief description: The users are able to change their passwords.

Actor: Employee

Input conditions: The user has logged in

Output conditions: The user has successfully changed their password

Main scenario:

1. The system presents the user with the main chat window
2. The user selects the settings tab
3. The system presents the user with the settings
4. The user chooses the change password option
5. The system presents the user with the change password window
6. The user enters the required information and sends it
7. The system processes the request and changes the account's password

UC09 Set the dark mode

Name: Set the dark mode

ID: UC09

Brief description: The users are able to set their preferred theme of the GUI. The basic interface is the light mode, therefore the user has to switch into the dark mode.

Actor: Employee

Input conditions: The user has logged in

Output conditions: The user has successfully set the dark theme for the application

Main scenario:

1. The system presents the user with the main chat window
2. The user selects the settings tab
3. The system presents the user with the settings
4. The user chooses the toggle GUI theme option
5. The system changes the theme to dark mode

UC10 Switch languages

Name: Switch languages

ID: UC10

Brief description: The users are able to set their preferred language. On the first run, the users are prompted to choose a language, but after that they have to go into the settings to switch languages.

Actor: Employee

Input conditions: The user has logged in

Output conditions: The user has successfully switched the app's language

Main scenario:

1. The system presents the user with the main chat window
2. The user selects the settings tab
3. The system presents the user with the settings
4. The user chooses the toggle language option
5. The system changes the language of the application

UC11 Edit members of a group

Name: Edit members of a group

ID: UC11

Brief description: The project leader is able to edit the members of the group.

Actor: Project Leader

Input conditions: The user has the privileges of a project leader and is part of a group

Output conditions: The project leader has successfully edited the members of the group

Main scenario:

1. The system presents the project leader with the main chat window
2. The project leader selects the group message tab
3. The system presents the project leader with the group message tab
4. The project leader selects the group who they wish to edit
5. The system presents the project leader with the edit member window
6. The project leader edits the members of the group
7. The system processes the request and presents the project leader with the changed list of users in the group

UC12 Create a group

Name: Create a group

ID: UC12

Brief description: The employer is able to create groups for new projects.

Actor: Employer

Input conditions: The user has the privileges of an employer and has logged in

Output conditions: The employer has successfully created a group

Main scenario:

1. The system presents the employer with the main chat window
2. The employer selects the settings tab
3. The system presents the employer with the settings
4. The employer chooses the create group option
5. The system presents the employer with the create group
6. The employer fills the required information and sends the request
7. The system processes the information and creates the group

UC13 Delete a group

Name: Delete a group

ID: UC13

Brief description: The employer is able to remove already created groups.

Actor: Employer

Input conditions: The user has the privileges of an employer and a group already exists

Output conditions: The employe has successfully removed a group

Main scenario:

1. The system presents the employer with the main chat window
2. The employer selects the settings tab
3. The system presents the employer with the settings
4. The employer chooses the remove group option
5. The system presents the employer with the remove group window
6. The employer chooses the group and sends the request
7. The system processes the information and removes the group

UC14 Register a user

Name: Register a user

ID: UC14

Brief description: The employer is able to create (register) users into the application

Actor: Employer

Input conditions: The user has the privileges of an employer and has logged in

Output conditions: The employer successfully registered a new user

Main scenario:

1. The system presents the employer with the main chat window
2. The employer selects the settings tab
3. The system presents the employer with the settings
4. The employer chooses the register user option
5. The system presents the employer with the register user window
6. The employer fills in the required information and sends the request
7. The system processes the request and registers the new user

UC15 Remove a user

Name: Remove a user

ID: UC15

Brief description: The superuser is able to remove users from the application

Actor: Superuser

Input conditions: The user has the privileges of a superuser and has logged in

Output conditions: The superuser successfully removed a user from the application

Main scenario:

1. The system presents the superuser with the main chat window
2. The superuser selects the settings tab
3. The system presents the superuser with the settings
4. The superuser chooses the remove user option
5. The system presents the superuser with the remove user window
6. The superuser selects the user to remove and sends the request
7. The system processes the request and removes the user from the application

UC16 Reset password

Name: Reset Password

ID: UC16

Brief description: The superuser is able to reset passwords of the users

Actor: Superuser

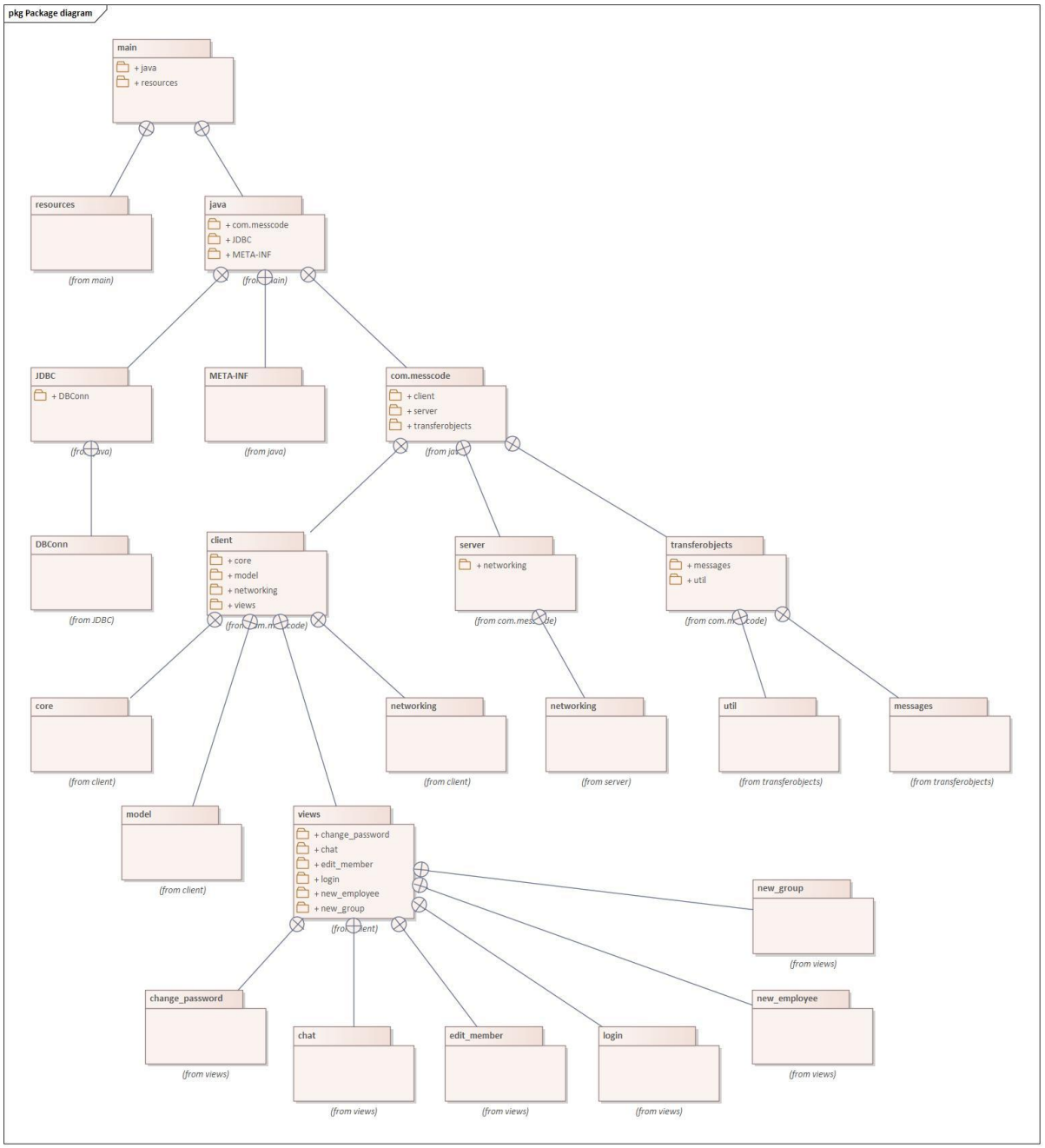
Input conditions: The user has the privileges of a superuser and has logged in

Output conditions: The superuser successfully reset the password of a user

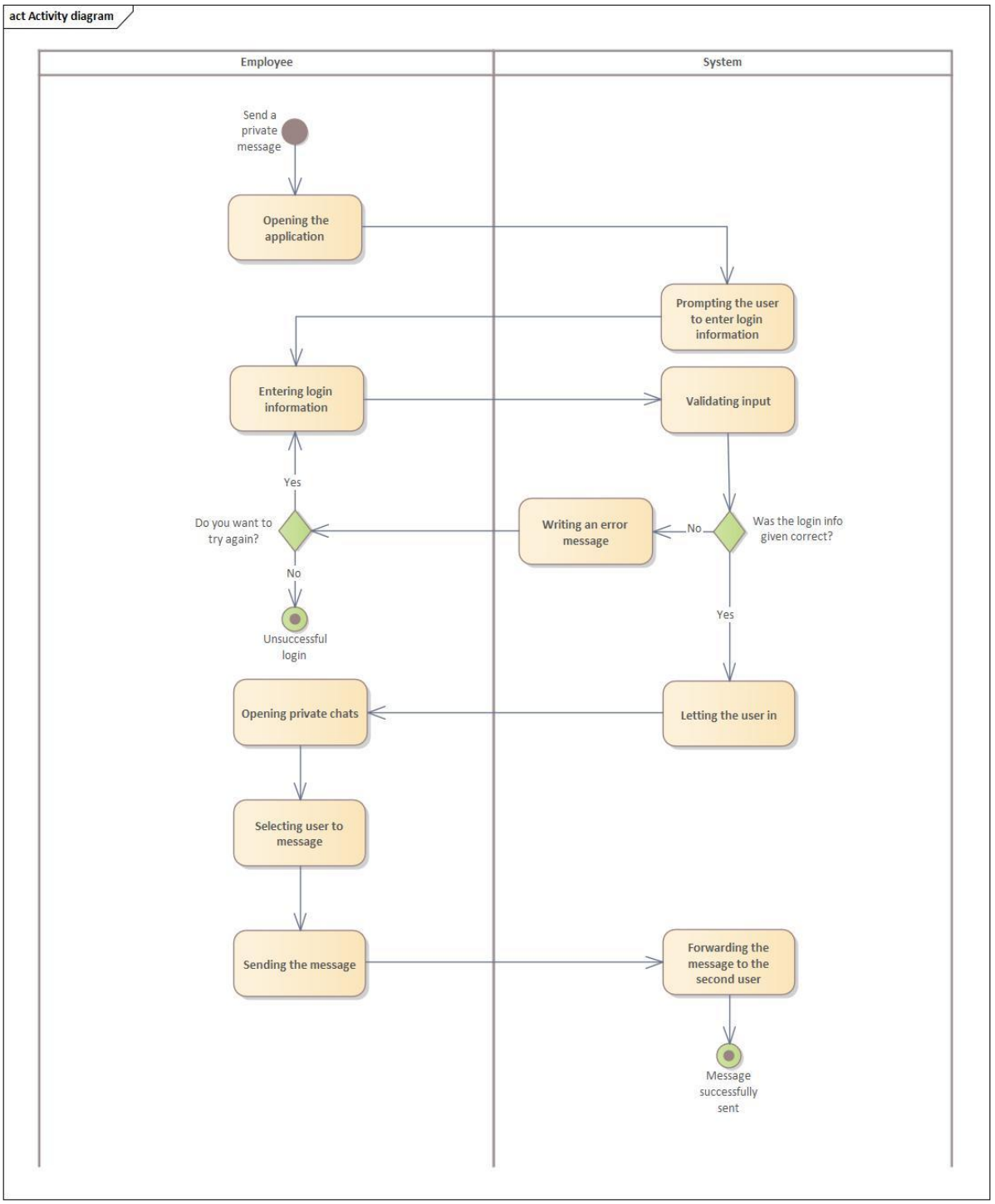
Main scenario:

1. The system presents the superuser with the main chat window
2. The superuser selects the private chat tab
3. The system presents the superuser with the private chats
4. The superuser chooses the user to reset the password of
5. The system presents the superuser with the selected user and its PMs
6. The superuser chooses the reset password option
7. The system presents the superuser with a popup window with the newly generated password
8. The superuser copies the new password and forwards it to the user

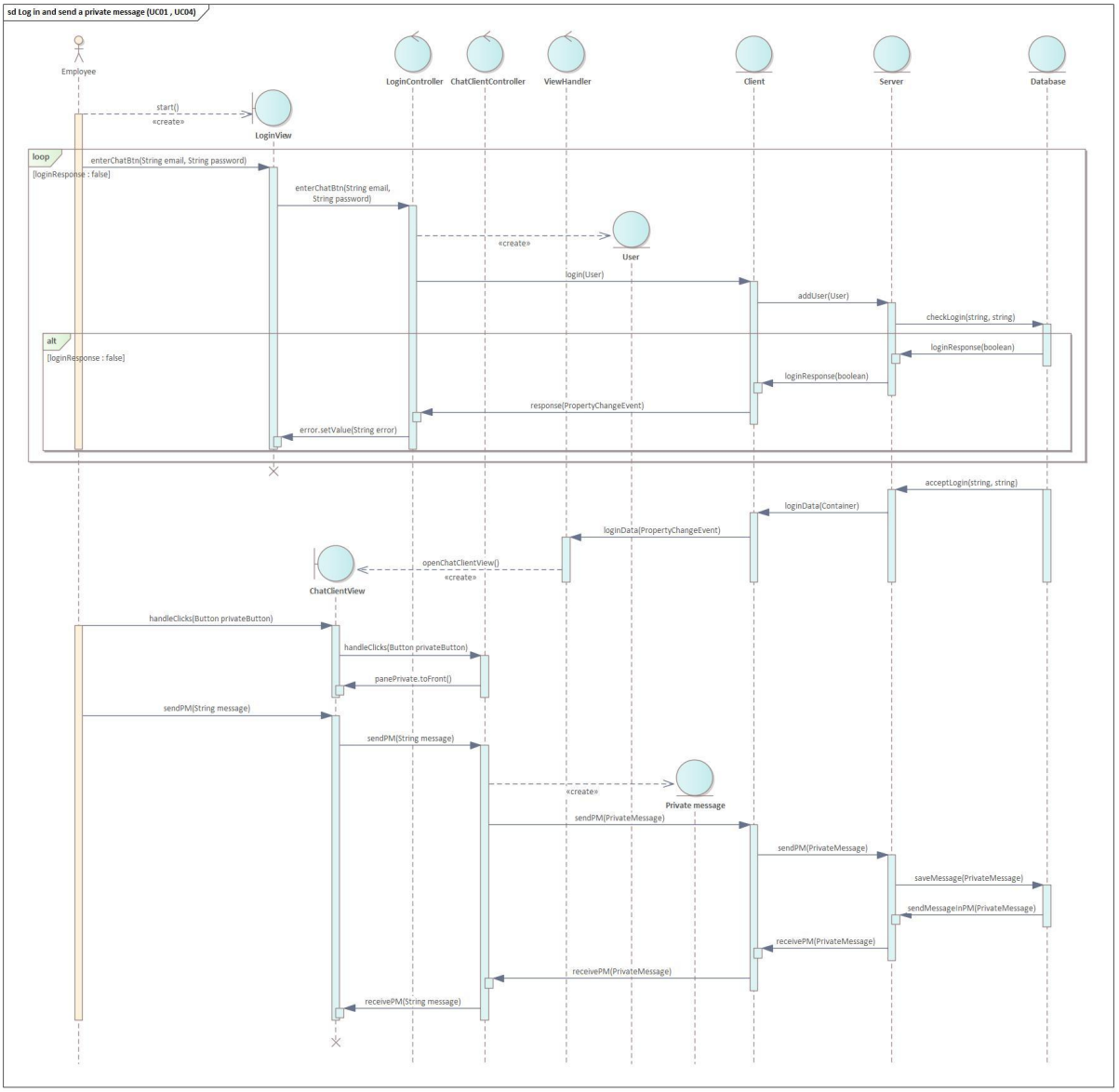
3.3 Package diagram



3.4 Activity diagram



3.5 Sequence diagram



4 User's Manual

In order to run the program, Java version 11 or higher must be installed on the device.

It is also necessary to locally create the database with the following parameters:

name: MessCode

username: postgres

password: chickenattack777

And restore it from the database backup file "UN-defined-messcode.tz".

The program can be run using the two included JAR files, a server-side and a client program:

Firstly start-up the server:

your_11_java_version\bin\java.exe -jar MessCodeServer.jar

Then run client-side application:

your_11_java_version\bin\java.exe -jar MessCodeClient.jar

To be able to effectively test the messenger it is recommended to run three client instances, with the employer, project manager, and worker accounts.

5 Conclusion

The goal of this project was to create a work-friendly messenger that is intended to improve communication between employers, managers, and employees.

The system is designed primarily for IT companies, for which the constant coordination of project activities is very important. It allows the programmers to get feedback from the managers faster, while the managers get a way to distribute and monitor the work tasks faster.

It is also worth noting that such an application is also an advantage for the employer himself who can check the progress of his employees at any time.

We wrote the project using a Model-View-ViewModel pattern and also used things like collections, localization, logging, XML, I/O, and regular expressions. Over the course of the project, we refined the graphical interface of the application so that it was user-friendly and pleasing to the eyes. We have managed to make it easy to use and efficient at the same time, especially for simple employees who always need to work efficiently.

6 Sources

Wikipedia Contributors (2022). *Model–view–viewmodel*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel [Accessed 11 Apr. 2022].

kexugit (2005). *Introduction to Model/View/ViewModel pattern for building WPF apps*. [online] Microsoft.com. Available at: https://docs.microsoft.com/en-us/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps [Accessed 11 Apr. 2022].

Guru99. (2020). *MVC vs MVVM: Key Differences with Examples*. [online] Available at: https://www.guru99.com/mvc-vs-mvvm.html [Accessed 11 Apr. 2022].