

Puppr

Dog owner socializing platform

Daria-Maria Popa
293087



Bogdan-Alexandru Mezei
293137



Natali Munk-Jakobsen
293132



Lukas Suslavicius
293717



Supervisors: Henrik Kronborg Pedersen, Joseph Chudwudi Okika



VIA University
College

Number of characters: 86.877

Software engineering

Semester 2

03-06-2020

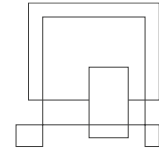
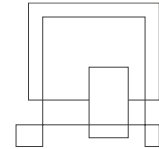


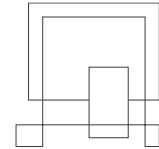
Table of content

Abstract	v
1 Introduction.....	1
2 Analysis	3
2.1 Requirements	6
2.2 Functional Requirements	11
2.3 Non-Functional Requirements	18
3 Design	20
4 Implementation	33
5 Test	43
5.1 Testing Methodology	43
5.2 Test Scenarios.....	43
5.3 Test Cases	44
5.4 Requirement Traceability Matrix	74
6 Results and Discussion.....	79
7 Conclusions	82
8 Project future	83
9 Sources of information	85
10 Appendices	i



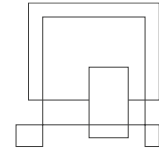
List of tables

Table 2.1. Use Case Description : Manage Post.....	15
Table 5.1. Test Scenario #1 - Checking the signup functionality	44
Table 5.2. Test Scenario #2 - Checking the login functionality	47
Table 5.3. Test Scenario #3 - Checking the posting functionality	48
Table 5.4. Test Scenario #4 - Checking the commenting functionality	53
Table 5.5. Test Scenario #5 - Checking the system behavior while managing dogs ...	57
Table 5.6. Test Scenario #6 - Checking the liking functionality.....	61
Table 5.7. Test Scenario #7 - Checking the system behavior while managing personal details	64
Table 5.8. Test Scenario #8 - Checking the system behavior while moderating with admin privileges.....	67
Table 5.9. Test Scenario #9 - Checking the “Hall of Fame” feature	73
Table 5.10. Requirement Traceability Matrix: Functional requirements	74
Table 5.11. Requirement Traceability Matrix : Non-functional requirements.....	77



List of images

Image 2.1. Domain Model.....	5
Image 2.2. Use Case Diagram.....	12
Image 2.3. Activity Diagram : Manage Post	16
Image 2.4. System Sequence Diagram.....	17
Image 3.1. Layered Architectural Pattern.....	20
Image 3.2. (Remote) Proxy Design Pattern.....	23
Image 3.3. Manage Profile menu with the “X” icon above	26
Image 3.4. Main Feed menu bar with “Add Post” button	26
Image 3.5. The colour scheme.....	27
Image 3.6. Example of the colour scheme in use.....	27
Image 3.7. Conceptual ER-Diagram	29
Image 3.8. Global ER Diagram	30
Image 3.9. Dogs table in the database.....	31
Image 3.10. AddDog sequence diagram.....	32
Image 4.1. DatabaseConnection Class and Singleton Design Pattern	34
Image 4.2. Example Of Adapter Design Pattern-1	35
Image 4.3. Example Of Adapter Design Pattern-2	35
Image 4.4. Verification of post likes.....	38
Image 4.5. blockUser method	40
Image 4.6. ImageToByte method.....	41
Image 4.7. Usage of RemoteSubject and RemoteListener interfaces	42



Abstract

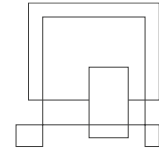
Daria Maria Popa

This paper proposes a novel idea for a socializing application that is specifically catered towards dog lovers, called “Puppr”. The main objective of this project is creating a social environment where the focus is equally split between the owners and their dogs, with features aimed towards aiding the interaction of people and showcasing the relationship between them and their canine companions.

The analysis of this problem domain puts forward a list of domain objects interconnected through a different range of relationships, which help describe and design the main architecture of the program. For the creation of this project extensive use of Java and Git technologies was employed, combined with Structured Query Language (SQL), used for managing the significant volume of data involved.

After implementation numerous system and acceptance tests, as well as unit tests, have concluded that all the requirements posed by the stakeholders have been successfully applied to the app. Based on this, the behaviour of the product as well as its specifications were deemed satisfactory for consumer usage and in tone with what the market needed.

Overall, the “Puppr” app managed to bring a new view on modern social interaction for dog owners, while keeping a simple but secure user experience and effective data handling.



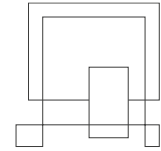
1 Introduction

Daria Maria Popa

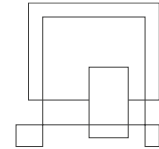
Pets have been one important part of the human life since prehistoric times (Pet | animal, 1998) and, as time progresses, their role in people's lives grows exponentially. With pets being considered members of the modern family, the latest figures show that 80 million European households alone have a pet (The Important Role Pets Have In Society - FEDIAF, 2020). The history of pets is intertwined with the process of animal domestication, and it is likely that the dog, as the first domesticated species, was also the first pet (Pet | animal, 1998). Dogs popularity only grew with time, mostly due to their friendly and helping nature, but also their health and physical benefits (cdc.gov, 2019), a study conducted in 2018 noting that "approximately 900 million people are dog owners" (Sundra Chelsea, 2018).

Due to the amount of pet owners nowadays it is only natural that the pet markets all over the world are growing and evolving at dramatic rates. From pet food, to toys to pet insurance, the American Pet Products Association (APPA) reported that Americans alone spent \$69.5 billion on the pet industry in 2017 (Here's a Look at the Pet Markets Trends Around the World, 2018). Online markets and platforms are the newest and most convenient method of shopping. Dog lovers have access to a wide array of online markets and social platforms such as Instagram or Facebook but few of them are made specifically for them. Because of this, the demand for an online place where the dog- loving community can gather and interact, organize events, showcase their dogs and socialize is on the rise.

Famous platforms such as Facebook and Instagram handle this demand quite poorly and offer no benefits or special features for the dog loving users they have. More specific platforms and applications like Dogster, Chewy and FitBark aid with ordering food, monitoring dogs' health and adoption, but still none offer a place for only dog lovers to



share their pictures, post, comment and interact with other owners. That is why social platforms where the focus is not only on the user but also on the pets are needed.



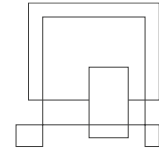
2 Analysis

Daria Maria Popa

Today's social world has been split between massive socializing platforms such as Facebook, whose goals include helping people of diverse backgrounds connect and interact in an easier way, and more niche platforms specialised in catering for a certain group of people. One of these always growing groups of people with a dedicated fan base is represented by none other than dog owners and dog lover alike.

As mentioned before (see chapter 1), the pet industry is broad and constantly evolving. Despite this, there are no specialised platforms created with the intention of helping dog owners interact and socialize better. Having platforms for online pet markets and health monitoring is helpful for any owner, but socialising is something that should also ought not to be forgotten or dismissed. The idea of socialising 'better' includes a certain list of key features that the platform must provide to the user. Based on the stakeholders' description these feature include: being able to interact with other owners by some public means of communication, having a focus on the pets and the ability to showcase them to people in a more unique way that the user can see and engage with easier, having a sense of security and good administration when using the app, and most importantly finding the platform easier to use than the existing ones.

The features wanted for a new social platform for dog lovers help outline the main objects that need to interact and exists in the platform. The first and most important one of these is the user. The user represents the stakeholders, the owners, which come into direct contact with the app and want to utilise it. The user should have the choice to create a profile in order to be identifiable by other users and have the ability to save and link their information to the said profile. The profile should contain not only their personal



information but also their dogs', putting the pets on a more important place like the stakeholder specified.

Because of the specified need of a safe environment and good administration of the new platform, another domain object is represented by the admin. The admin will act as a normal user but also be the one in charge of checking the interactions between users and have the ability to take actions against potentially harmful ones which might deter other users from using the platform.

The need for interaction with other owners described by the problem domain calls for an object catering to these needs. This object is represented by the post a user can make. A post can be viewed by anyone, liked, and commented on, to ensure a means of communication for the stakeholders. The posts can all be added to a public feed, refreshing occasionally, and showing posts chronologically. In the same fashion, the dogs can be liked by users, ranked, and listed on another feed or part of the platform, making for the more 'unique' approach to dogs as an integral part of users' profiles.

As a last part of the list of domain objects, a person, represented by either a dog owner or simply dog lover, should be able to act as a guest to the page, see the main feed and explore the posts before deciding to join the community. This will engage and add to the sense of security that the stakeholders mentioned.

In order to visually represent how these real-situation objects interact with each other a domain model can be constructed (Larman, 2004). The domain model below shows how the seven objects found to be a part of the problem domain based on the stakeholders' description interact and are linked to each other, making for a 'visual dictionary' (Larman, 2004) of the concepts and ideas presented in the background description.

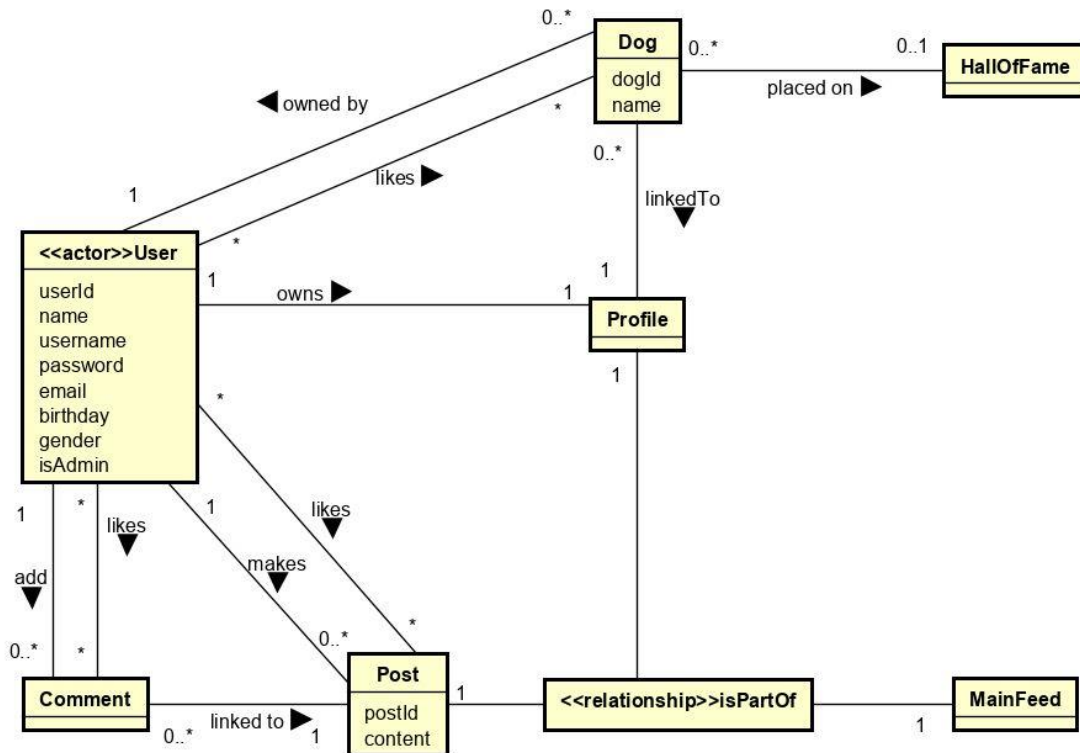
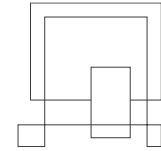
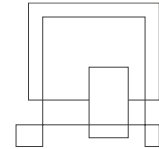


Image 2.1. Domain Model

The user is described by the username and password used for logging. Due to the fact that the admin has all the normal user functions and then extra administrative functions, it is not represented as another actor but an attribute of the user domain object which confers or takes from the user the right to manage a platform. The id is a way of distinguishing between all users. The guest is not directly interacting with any other objects, so it is not visualized in the domain model.

The post is simply described by the id and its content, being linked to the user, the maker of the post, the user's profile, and the main feed where all posts appear. The comment object is too linked to the profile, the user being able to comment on posts or like already existing in order to achieve the wanted interactivity. User has the possibility to like posts too, again helping the users interact on the platform.



A user can also choose to like a dog object, thus showing their appreciation for it. The dog is linked to a user's profile and can possibly be placed on the hall of fame, a part different from the main feed where dogs are ranked based on likes and showcased to everyone.

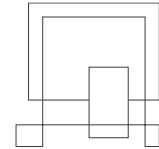
The associations mostly have a multiplicity of one to many, showing that, at a particular moment and in the context given by the stakeholders, zero or more instances can be associated to one of the users or dog instances. The exceptions consist of the user – profile and post – profile - main feed associations with multiplicity of one to one showing that a user or admin must have one and only one profile and every post will be linked to the one main feed and the profile of the post's author. The dog – Hall of Fame associations also shows that a dog may or may not be part of the one Hall of Fame.

The domain model is the end result of the whole analysis of the project and represents the basis for further designing and then implementing the system.

2.1 Requirements

Every subject system is driven by the people, entities, triggers and supporting systems interacting with it constantly to make it work and fulfil its intended purpose. These are also known as the actors of a system, the ones initiating the use case (Types of Actor in a Use Case Model, 2018). Based on Larman's definition, an actor is anything with behaviour, including the system itself when it calls upon the services of other systems (Larman, 2004) and any other external hardware or subjects.

Actors are of different types based on their roles and goals regarding the system. The two main types are represented by the primary and supporting actors. The primary actor is a user whose user goals are fulfilled by the system. They are often the ones to also trigger the use case (Cockburn, n.d.).



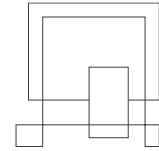
The primary actor for this system is the user of the app. A user is a person who uses the app for its intended purposes of engaging with other people and posting their own content. Their goals are represented by the ability to post, to interact by liking and commenting, to edit their posted content and to feel safe while using the platform. The expected outcome from the system is letting the user perform the actions they are allowed to, saving and updating the information after each action. The users are the ones who trigger the use case by needing a new platform modelled to their needs.

The admins are also primary actors because, even though they have a type of supporting role for the system, they also have all the functions of the normal user and can call on the system to deliver its services in the same way. Their goals include making the platform a safe space for the users by managing all the content posted to the platform and monitoring the users' activity. The system should let the admin make changes to posts and block or unblock users freely.

Supporting actors are the second type of actors, offering a service to the system, for example offering information (Cockburn, n.d.). Because of the data related to the users and dogs needing to be saved, stored and retrieved constantly for this type of platform, the secondary actor will be represented by the database that will handle and provide the information needed to the primary actors, while also saving the information they provide.

The actors are the ones involved in the user stories, which depict the different goals and scenarios the system must be able to handle in the way specified by the user or stakeholder. The user stories can be seen as the functional requirements of the system. As a whole, requirements represent the capabilities to which the system must conform (Larman, 2004).

The requirements are split into functional, non-functional, and sometimes domain requirements. The functional requirements are, as aforementioned, the ones demanded by the user and seen as basic facilities that the system should offer (Software Engineering | Classification of Software Requirements - GeeksforGeeks, n.d.). Non-functional



requirements are related to the system's portability, security, maintainability, and other non-behavioural elements that impose a quality constraint. The last type, domain requirements are the requirements that a part of the system related to the specific domain must exhibit (Software Engineering | Classification of Software Requirements - GeeksforGeeks, n.d.). This project does not have any particular domain requirements although like and comment are the same no matter is the actor is a user or admin.

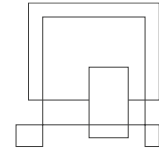
For the proposed idea of the Puppr platform a set of functional and non-functional requirements were identified based on the user stories related to the users, admins, and guests. This list of functional requirements is prioritised from most important to least, and split into critical, high, and low priority to the system as well.

Both functional and non-functional requirements were prioritised using the MoSCoW requirements prioritization Technique (Business Analyst Learnings 2013). Its four categories: must, should, could, will not or would, were used to decide the requirements' importance for the current project and if they were or not relevant.

The list of functional requirements is as follows:

Critical priority

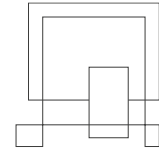
1. As a user, I want to be able to create a new profile so that I can use all the systems functionalities and be part of the community.
2. As a user, I want to be able to add pictures and information about my pet(s) when modifying a profile so that I can have a special part of the profile dedicated to my dog(s).
3. As a user I want to remove or add a pet's information to my profile freely, so that the other users can see an up to date list of my pets.
4. As a user, I want to be able to add posts as a text description with or without an image on the platform, so that I can share my pet-related experiences and announcements with other people.



5. As a user, I want to be able to like a post, so that I can show the user my appreciation for their posted content.
6. As a user, I want to be able to comment on other users' posts as a means of contacting and interacting with other users.
7. As a user, I want to be able to see the comments on every selected post so that I can know the other users remarks and thoughts about it.
8. As an admin, I want to be able to use all user-level functions (adding posts or comments, liking other users' posts and comments and adding dog information) so that I can be an active member of the community and moderate the platform better.
9. As a user, I want to be able to see other users' posts in real time so that I can follow the latest news in the community.

High priority

10. As a user, I want to be able to update selected fields in my profile, so that I can have full control over the personal information available on it.
11. As an admin, I want to be able to delete user posts and comments when needed so that I can manage unwanted content and provide a well-maintained main feed.
12. As a user, I want to be able to edit my comments or posts after they have been posted so that I can easily correct any mistakes without having to delete the post/comment.
13. As a user, I want to be able to see other users' profiles, so that I can interact easier with their posts and dog posts.
14. As an admin, I want to be able to block/unblock users when needed so that I can provide a safe and friendly environment on the platform for the users.

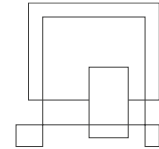


Low priority

15. As a guest I want to be able to see the main feed of the application whenever, without needing to be signed into an account, so that I have easy access to the platform's main content and source of entertainment.
16. As a user, I want to be able to like other users' comments so that I can show my interest and support for the comment.
17. As a user, I want to be able to like other users' dogs so that I can help my favourite dogs get a place on the Hall of Fame.
18. As a user, I want to see the top 10 most liked dogs on the Hall of Fame with their owner information, so that I can view their profile and see more posts from them.

The non-functional requirements are as follows:

1. The system must support Windows 7, 8 and 10 OS.
2. The system will not be compatible with any mobile or tablet interfaces.
3. The system must refresh the page after a change has been committed or new page opened in 0.5 seconds 90% of the time and less than 2 seconds the rest of the time.
4. Adding high quality images or other complex information to the system must provide a maximum of 5 seconds response time for the user.
5. All changes processing during a system crash on either server or client side must not be committed to the database and not saved in the system to ensure no faulty data entering the system.
6. No information already existing in the system should be affected during a system crash.
7. No normal user or guest should have access to any admin rights or the option to give themselves admin rights.
8. Passwords must be stored as hashed passwords in the database.



9. Users trying to add information or manage their accounts should result in an error rate of less than 5 percent.
10. Time stamps and other date formats must be as follows: day/month/year.

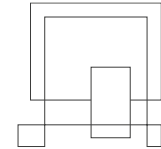
2.2 Functional Requirements

Functional requirements are, as mentioned before, based on user stories, and captures in the use case. They describe what the system must or should fulfil, based on their priority, and result in a functional product feature of the system.

For the current project, the functional requirements are based on the user, admin and guest requirements, with admin and user being the primary actors which interact with the system and trigger the user case. Based on these actors and their roles and requirements, taken especially from the actor descriptions (see chapter 2.1), the use case diagram can be constructed, in order to have a visual representation of the expected behaviour of the system (What is Use Case Diagram?, n.d.), a contract of how the system will behave (Cockburn, n.d.).

In the use case diagram bellow (Image 2.2.) the relationships between the admin and user actors, the use cases and the system are shown, in order to create a clear representation of who is using the system, what are their goals and what scenarios do they use to get to their goals.

From the diagram there are a number of cases that both primary actors have access to and can trigger individually. These represent the main functions of a user that the admin can do too, for example liking, commenting, and posting. This will ensure a more rounded admin actor and add transparency and accountability for the admins which will make the platform feel safer from the user perspective.



For security reasons, the admin will not be able to be created using the app's create profile use case. In the same way only the admin will be able to manage users (block or unblock) and manage all the posted content (posts, comments and dogs) by having the freedom to edit or delete any hateful or otherwise hurtful content.

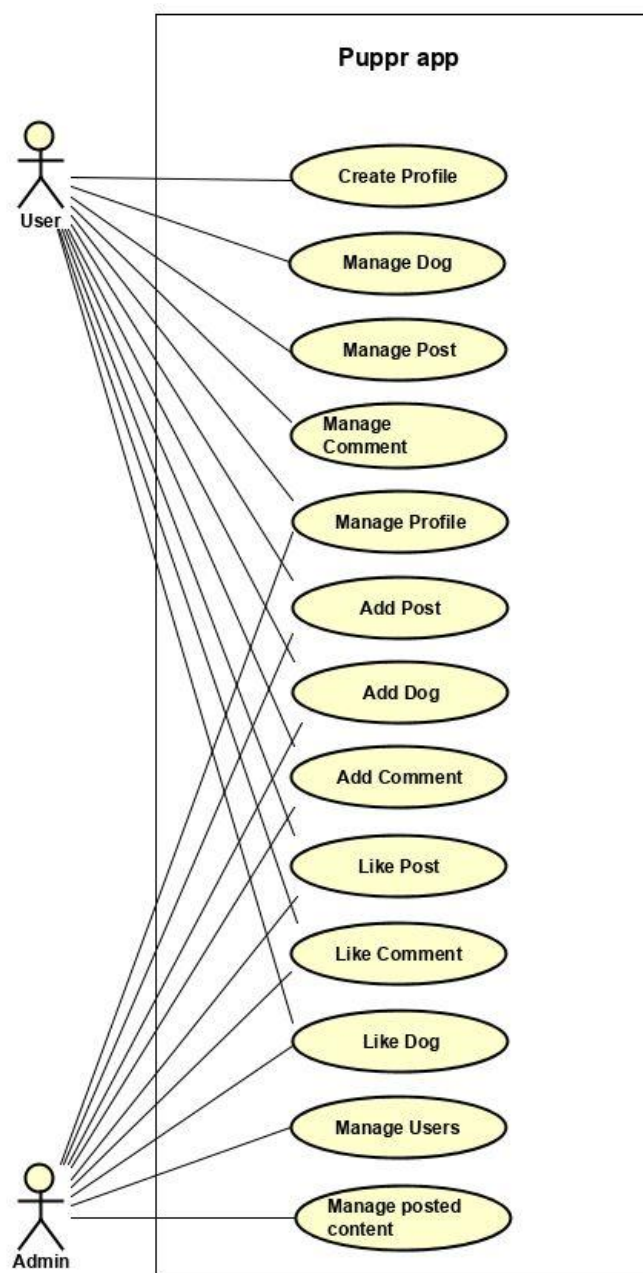
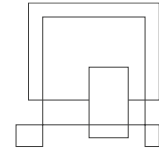


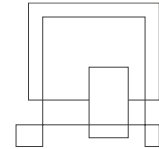
Image 2.2. Use Case Diagram



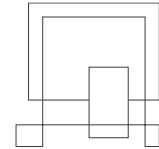
The use case diagram can be further explained with the help of fully dressed (Larman, 2004) use case descriptions, by explicitly detailing all the variations of each use case and their interests, goals and scenarios.

Manage post use case, is an example of a use case that is on user and admin level, can be triggered by both at different times, but does not differ based on these two user types. The success guarantee section describes how the system should react on a normal add post activity, being described in more detail using the main success scenario. The alternative flows describe other options the actors might pick during the scenario, or potential fail scenarios that might happen. Last four sections handle related non-functional requirements, technology involved, miscellaneous and the frequency at which the use case might occur in the system. The rest of the use case descriptions can be found in the Appendix.

Use case uc12: Manage Post	
Scope	Puppr application
Level	User goal
Primary actor	User
Stakeholder and interests	<ul style="list-style-type: none"> - User: Wants to have the ability to remove or edit their posts after posting them to the app - Admin: Wants the users to have the possibility to fix any mistakes or delete any of their posts freely
Preconditions	User has already a set up account
Success guarantee	The selected post data is modified and updated accordingly and immediately, so other users can see the new information in real time.



Main success scenario	<ol style="list-style-type: none"> 1. User logs in using their credentials (username and password) 2. User opens their profile page and decides to delete or edit of one their posts 3. User clicks the related "delete" or "edit" buttons for one of the posts 4. The data is deleted or edited accordingly if the changes are saved 5. All changes are committed and updated for all users
Extensions (or alternative flows)	<ol style="list-style-type: none"> 1a. Login information is incorrect <ol style="list-style-type: none"> a. The user/admin will be prompted to check the information and try again b. If the login succeeds, they can proceed to the next step, otherwise they can only see the main feed as a guest 2a. User has no posts <ol style="list-style-type: none"> 1. The system will not display any edit or delete buttons until the user completes an "add post" operation in the app 3a. User selects the delete option <ol style="list-style-type: none"> 1. The post data will be deleted immediately, as well as the comments and likes linked to it 2. Both the profile page and the main feed will be updated for everyone 3b. User selects the edit option <ol style="list-style-type: none"> 1. A new window showing the current information related to the respective post will pop up 2. The user can select what to change and what to leave the same way <ol style="list-style-type: none"> a. User chooses to save the changes:



	<ul style="list-style-type: none"> ▪ The new old information will be replaced with the new one and changes committed b. User chooses to exit the edit menu: <ul style="list-style-type: none"> ▪ The old information will remain unedited
Special Requirements	System updates the admin changes and notifies the user within 2 seconds 95% of the time.
Technology and Data Variations List	
Frequency of Occurrence	Often
Miscellaneous	

Table 2.1. Use Case Description : Manage Post

Because of the significant number of scenarios or paths the actor can take in the system, there is a need for further scenario clarification. This can be done with the help of multiple types of diagrams.

Activity diagrams are often used to visualize workflows and use cases. In the case of a user managing an already existing post, there are many scenarios variations that can be easily understood with the help of an activity diagram (Image 2.3) .

The two partitions show the parts involved in the use case. One is the user, the actor triggering the use case, and the other is the system itself, responding to user input, loading new windows, and committing changes to the database based on the user's decisions.

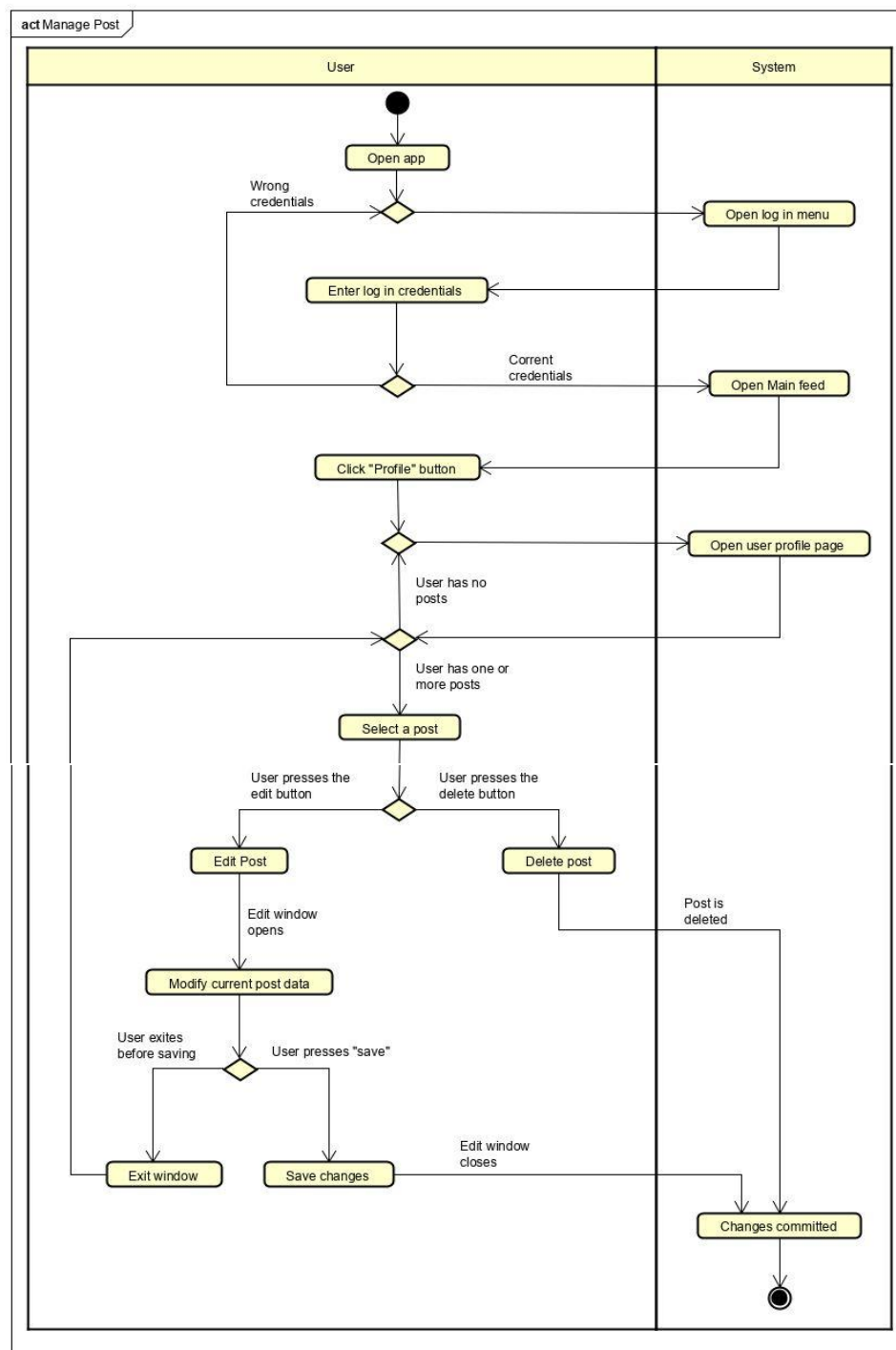
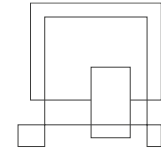
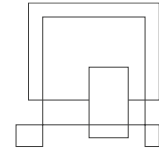


Image 2.3. Activity Diagram : Manage Post



The alternative flows are indicated by the decision nodes and the different paths by the arrows connecting the steps of managing a post depicted in the use case description.

For the blocking and unblocking of users by the admin, a system sequence diagram can be used to show not only the steps but, more importantly, the time element of the use case.

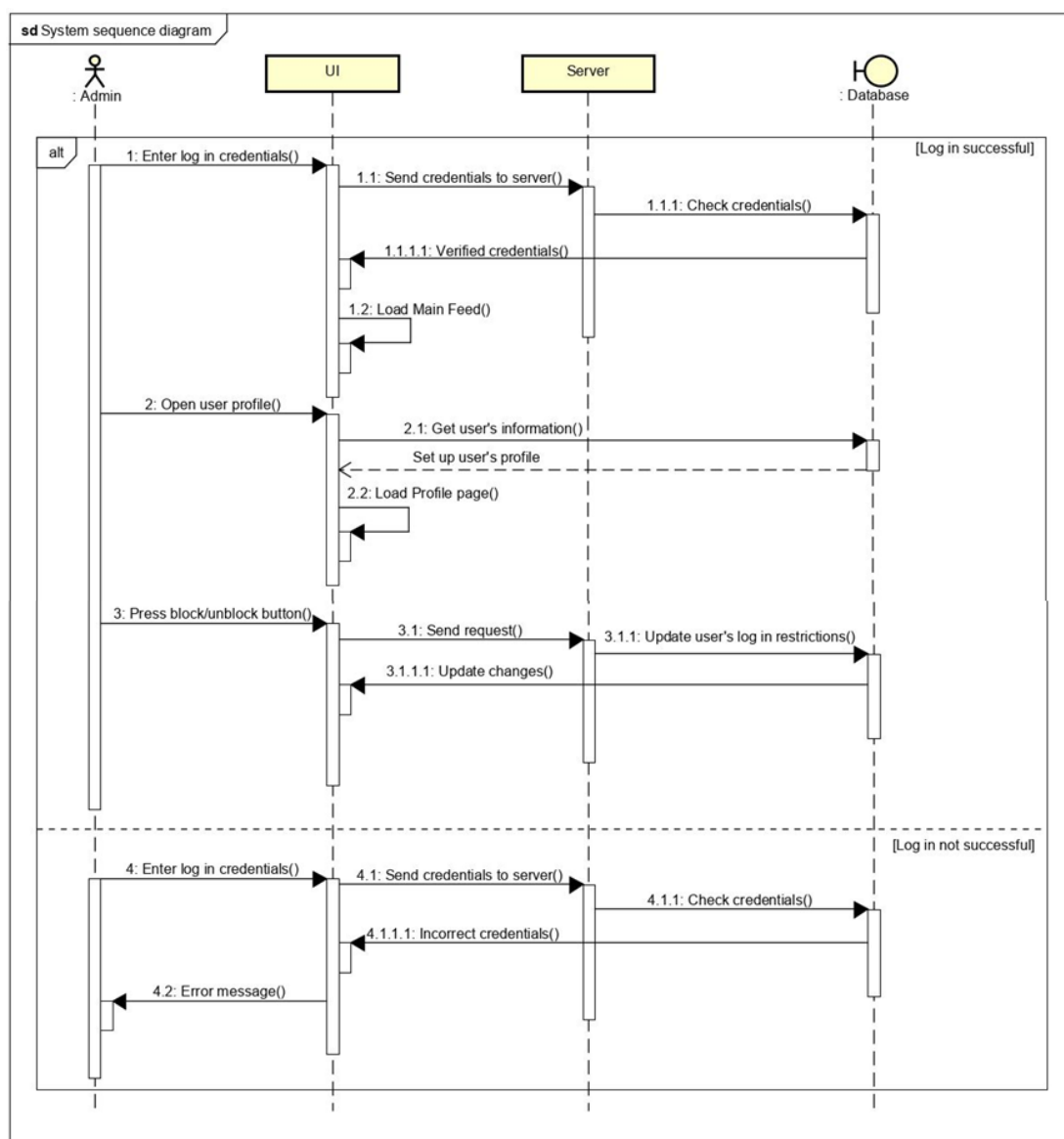
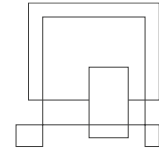


Image 2.4. System Sequence Diagram



All in all, the use case diagrams, analysis and system sequence diagrams are a helpful tool for explaining and understanding the user stories and the functional requirements that come with them.

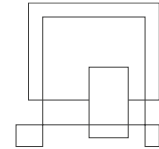
2.3 Non-Functional Requirements

Non-functional requirements (NFR) refer to qualities or specifications of the product instead of behaviour, which is described by the functional requirements. They can be of different types, tackling different subjects of matter accordingly.

One of the first and most important types of NFR handle the performance and scalability of the program. This is important for both the immediate users and the future maintenance work for the system, and it simply tells how much time does the system usually take to return a result and how that changes based on higher workloads (Non-functional Requirements: Examples, Types, How to Approach, 2019).

For this current program, the performance and scalability requirements decided on are: "The system must refresh the page after a change has been committed in 0.5 seconds 90% of the time and less than 2 seconds the rest of the time" and "Adding high quality images or other complex information to the system must provide a maximum of 5 seconds response time for the user". The response time was chose based on the metric of 3 important limits for response time (Nielsen, 1993).

Portability and compatibility are also important topics for a software program, handling the system's response in different environments and how well can it co-exist with other systems. In this case there are two main portability/compatibility related requirements: "The system must support Windows 7, 8 and 10 OS" and "The system will not be compatible with any mobile or tablet interfaces". Both are based on researching the preferred environment for the target audience.



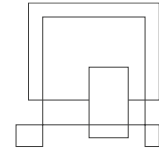
The recoverability, reliability and maintainability are a focus for this system based on the stakeholders' wishes. They deal with the system's response in case of a crash or failure and are as follows: "All changes processing during a system crash on either server or client side must not be committed to the database and not saved in the system to ensure no faulty data entering the system" and "No information already existing in the system should be affected during a system crash".

Another important type of NFR are security-oriented ones and they assure protection against unauthorized access. The security concern for this system is mostly related to admin status and the rights that come with it: "No normal user or guest should have access to any admin rights or the option to give themselves admin rights". Sensitive data safety is also a major point ensured by this requirement: "Passwords must be stored as hashed passwords in the database".

Usability and Localization are the last two attributes defining non-functional requirements. Usability measures the grade of difficulty that the user can encounter while using the system. For the current system that is defined by "Users trying to add information or manage their accounts should result in an error rate of less than 5 percent", conforming to the wish of the stakeholders for an easy to use platform.

Localization defines how well a system falls in line with the context of the local market-to-be (Non-functional Requirements: Examples, Types, How to Approach, 2019). For the current platform, the localization NFR is "Time stamps and other date formats must be as follows: day/month/year".

Overall, the non-functional requirements have been set by taking into consideration the stakeholders' interests and the testability of the system, as well as any architectural or other limitations that might appear in this particular system.



3 Design

Daria Maria Popa and Natali Munk-Jakobsen

Natali Munk-Jakobsen (Architecture and Design Patterns)

The structure of the system is based on **Client-Server architecture** combined with the **Layered architecture**. Client-server architecture is an architecture of a computer network in which many remote processors request and receive service from a centralized server. Client computers provide a user interface to allow the user to request services of the server and to display the results the server returns. The server waits for requests to arrive from clients and then responds to them.

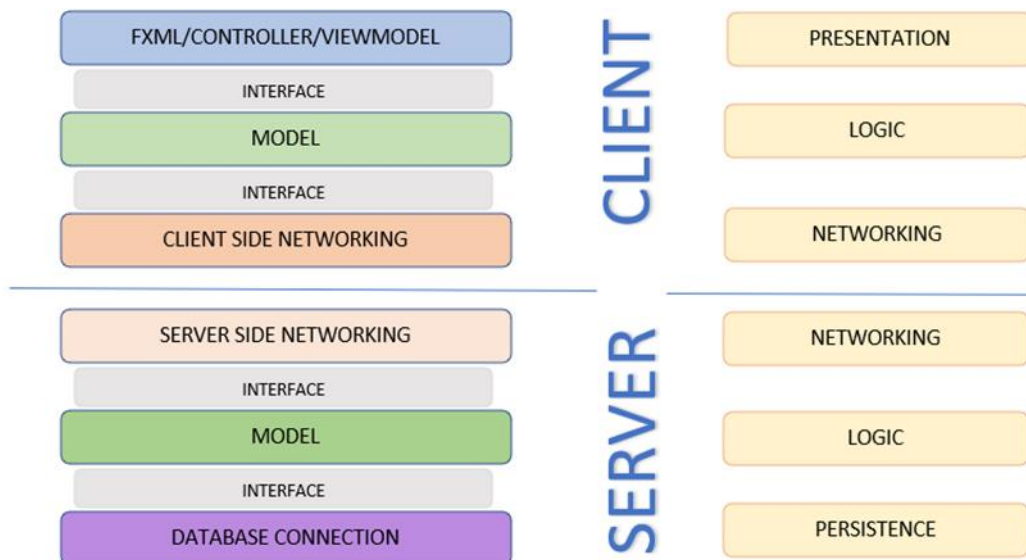
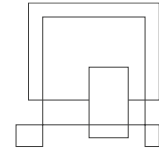


Image 3.1. Layered Architectural Pattern

The system consists of two main parts: server side and client side. Both server and client side consist of three layers (Image 3.1). The client side includes presentation, logic, and networking layers. The presentation layer contains FXML files, controller classes and



ViewModel classes which are responsible for presenting the user interface and getting input from the user. The logic layer contains model classes and the networking layer includes client side networking classes which have a part in client-server connection.

The server side has a similar structure and contains logic and networking layers for the same purposes. In addition, it includes persistence layer containing all database related classes and establishing the database connection.

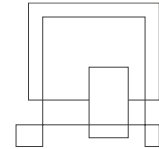
There are interfaces in between each layer to be compatible with the Dependency Inversion Principle (SOLID), which is based on the idea of "High-level modules should not depend on low-level modules. Both should depend on abstractions".

The Adapter Design Pattern has been used several times for delegating the work to another layer. Therefore, this design pattern has an important role in constructing the layered architecture in the whole system.

The layered architecture and usage of interfaces have been preferred in this project because they enable the programmer to change higher-level and lower-level components without affecting any other layers and provide maintainable and extendable code.

MVVM (Model-View-ViewModel) is an architectural pattern that is used across many platforms and languages. In the client side of the system, classes in presentation and logic layers and their interactions have been established based on MVVM in order to ensure a clean and well-structured program. Since Model, ViewModel and View parts are separated and responsibilities are divided in MVVM, it enables the programmers to test and modify the code in an easy way.

The View part contains FXML files and controller classes and it is responsible for visual representation of data and controlling all user interactions with the system. It displays data from the system and gets input from the user by using JavaFX elements, such as TextBox, RadioButton, Button.



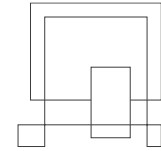
ViewModel part consists of 15 ViewModel classes in addition to ViewModelFactory class. ViewModelFactory is responsible for creating and holding all ViewModels. View controllers in the view part have a reference to corresponding ViewModel class and all ViewModel classes have a reference to LocalModel interface which defines a collection of methods accessing, modifying, and getting objects from the Model package. LocalModel interface is implemented by LocalModelManager which delegates LocalClientModel and it has a role in sending the information further to the server.

Although most of the updates in View and ViewModel are handled by bindings, the **Observer Design Pattern** is still needed in these parts. Since these parts are responsible for user interactions, they have to control this transaction dynamically. A view can notify another view automatically in case of any changes through the observer design pattern.

The connection between the client and the server has been established in networking layers by using **RMI (Remote Method Invocation)**. In RMI mechanism, a remote object is created and a reference of that object is made available for the client by using the registry on the server side. The client requests the remote objects on the server and invokes its methods.

RMI provides remote communication between Java programs. Since Java is the only programming language used in the system, RMI could be used for client-server connection without problem. It has been preferred because of its simple usage and although it can be less efficient than socket connection for some cases, in this project it provided a sufficient result for the intended system.

RmiServer and RmiClient classes are responsible for managing this mechanism. The stub plays the role of a proxy server for the remote objects. The server creates an object and registers this object with the RMIregistry by using a unique name known as bind name in the RmiServer class. In the RmiClient class, the client communicates with



the registry to obtain a remote reference to the server and the remote object is used for invoking the methods from the remote object.

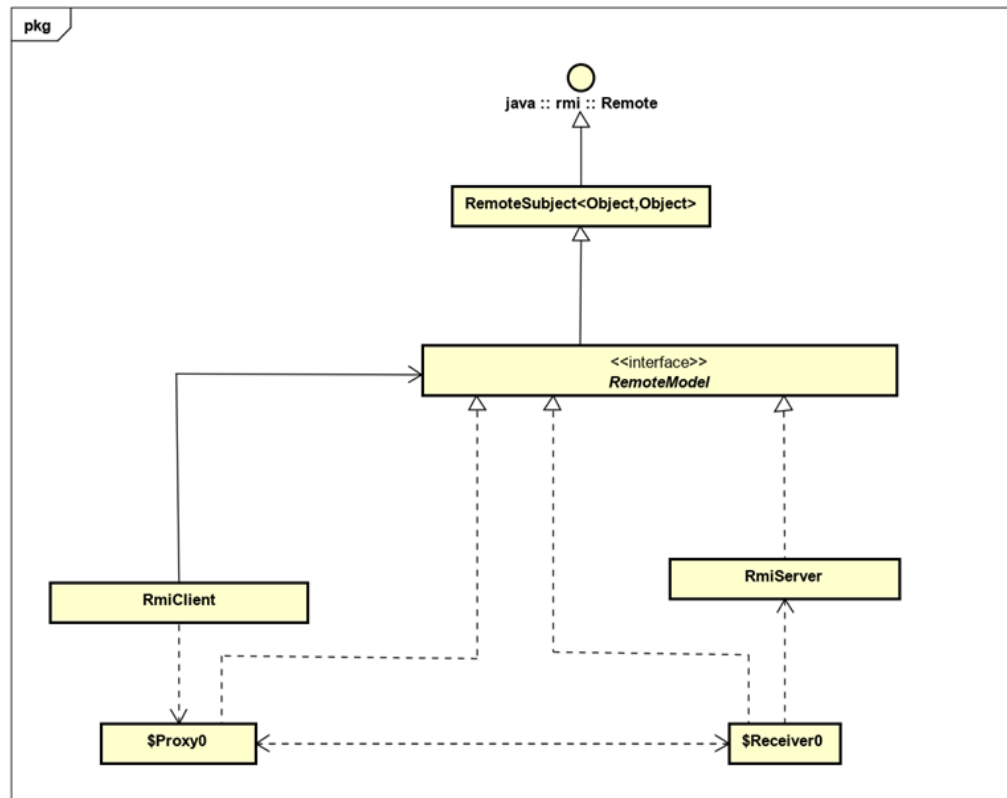
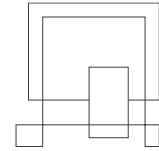


Image 3.2. (Remote) Proxy Design Pattern

The hidden part of the RMI mechanism makes extensive use of the Remote Proxy Design Pattern. A remote proxy provides a local representative for an object in a different address space.

As seen in Image 3.2, RmiClient has an association with the RemoteModel interface and the RemoteModel interface is implemented by RmiServer class. In addition, \$Proxy0 and \$Receiver0 classes implement RemoteModel interface. \$Proxy0 class deals with TCP sockets and sending data to the server side. In this pattern, RemoteModel interface is subject, \$Proxy0 class is proxy and it delegates the work to \$Receiver0 class and \$Receiver0 delegates RmiServer class which is the real subject.



Another design pattern that has been used in the networking layers is the Observer Design Pattern. This pattern is used for observing the server side and notifying the client in case of any changes. Since observable objects and observers are not in the same computer Remote interface takes part in the implementation of this pattern in RMI.

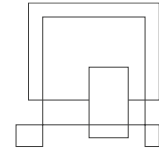
Singleton Design Pattern has been used in the persistence layer to assure that the DatabaseConnection class has only one instance and provides a global point of access to it. This pattern has been used to establish the database connection in an easy and safe way.

Daria Maria Popa (Technologies, UI design, Database design, Interaction Diagram)

In order to further implement the system, a number of technologies are needed. The main technology used for this project is Java, as the main programming language. The reason why it was chosen as the main language is because it provides great stability and scalability, while also ensuring cross-platform compatibility, which the use of database makes necessary.

Java is an Object-Oriented Programming language (OOP), making the transition between analysis and design objects to java objects easier to implement and represent in diagrams, while also aiding in the iterative approach chosen for this project. The rich API and source libraries offer extra help when creating an app that has a server-side, which is one of the main aims of this project. The JavaFX library is also useful for creating the UI needed for the client side of the platform. All Java related work was possible with the help of the IntelliJ IDEA IDE (Integrated development environment).

Another helpful technology is represented by SQL (Structured Query Language), which is used in the communication of the server to the database and creating the database itself. The use of SQL is not only needed for the queries, which help manage data in the table and return it to the server, but also with data modelling and data integrity. The use of SQL helps design a database with fast response speed, assured data quality and good



organization over large amounts of data. For this project, the database system used was PostgreSQL.

The last technology used is Git. Git provides many benefits in relation to code management and workflow management in general. It handles the release versions, with many branching or merging options, which are helpful for the iterative aim of the project. An added benefit comes from Java's cross-platform compatibility, allowing Git to and Java to work together as one in both implementing and branching the code, making for a much safer and better organized workflow.

UI design is a very important part of designing the application, having a big impact on how the stakeholders and users see the platform and how likely they are to use it for long periods of time. Based on the analysis and understanding of the users' wants, the UI needed to be simple and consistent, while having distinct differences from the other social platforms.

To ensure the simplicity of the interface, the number of menu bars and buttons was reduced to the bare minimum, with only two menus, in the "Main Feed" and the "Manage Profile" part of the system. The buttons with no importance for guest type users, such as "My Profile" or "Manage Profile" were hidden for an even more simplistic view.

The simple and fast flow between different windows of the app was ensured by button going to a specified page and the "x" icon going back to the most recent page the user came from (Image 3.4).

The size and placement of certain elements was also made with the purposeful intent of easy user navigation. A good example for this is the button for making a new post being significantly larger and of different size and colour than the other menu buttons (Image 3.3), drawing the attention of the user.

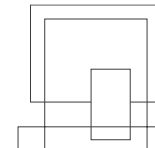


Image 3.4. Main Feed menu bar with "Add Post" button

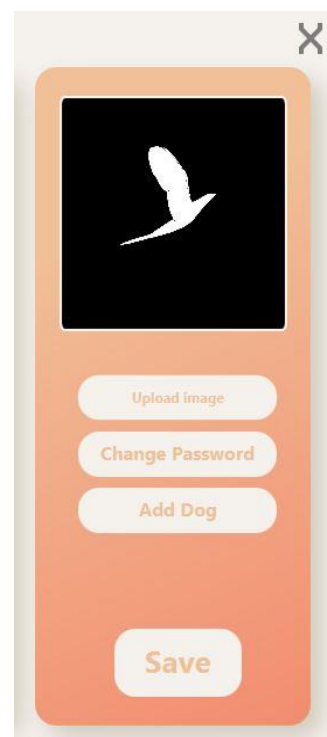
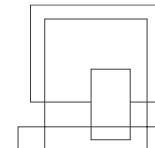


Image 3.3. Manage Profile menu with the "X" icon above

The consistency of the user interface is given by a number of design decisions. One of the first consistent designs throughout the system is the column arrangement of the containers with informational components and input controls. Everything from the first form, the log in form, to the profile page layout respects this design choice of vertical column separation where separation between different data sets is needed.

The column design is also related to the second consistent part of the UI, which is represented by the colour scheme. The colour scheme was chosen to be natural looking while still offering contrast and being different from the mostly blue-based tones of many social platforms today (Image 3.5). The off-white background with orange containers,



represented by the columns, and green highlights for buttons, text or image borders is kept the same on all the pages, menus, and buttons of the application. Red borders or text colour are also used to inform users of errors or mistakes when completing certain fields (Image 3.6).

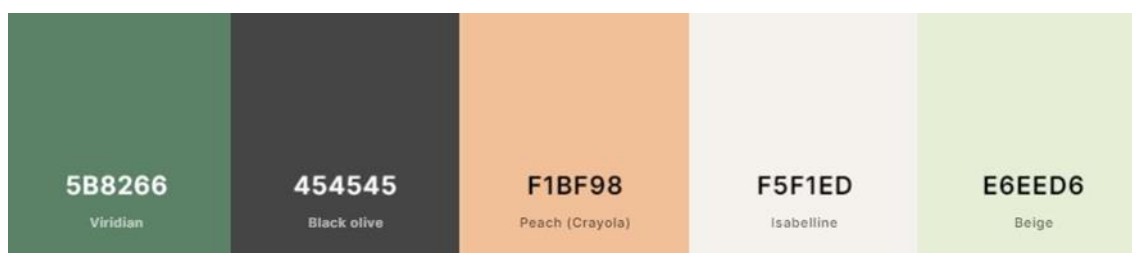
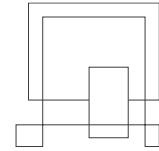


Image 3.5. The colour scheme

Image 3.6. Example of the colour scheme in use

The colour scheme also represents a useful tool of getting the user's attention by contrasting the pale warm colours of the background with the bright buttons, green usernames or handles and green textual information about the number of likes and comments a post has.



The last common element of the UI is represented by the rounded corners of all the containers and buttons, which also make for a more unique look of the app, contrasting and bringing the users' attention to the rectangular text fields or text areas and square image formats of the user posts.

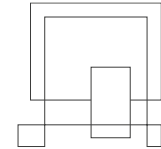
Communication between the system and the users is made possible with alerts and error messages. Alerts are used for errors such as incorrect login information or a guest type user trying to like, comment or add a post and are usually characterized by an option to simply close the alert and one taking the user to another page.

The error messages are often accompanied by red borders on the empty or otherwise incorrect fields they are referencing (Image 3.5). This UI design is guaranteeing the even better visibility of errors for users, making the system easier to navigate and use.

Overall, beside the design and aesthetic related choices, a crucial part of UI was optimizing it to better display the needed data, bring focus on people and dogs equally, and aid all types of users in reaching their goals with the system efficiently.

This system is related to and handles a significant amount of data both private and public, which is constantly being updated, deleted, or added. Therefore, having a database handling and sorting this data was necessary to ensure the persistence of the system. The Puppr database has a total of seven tables, all in a single schema. As mentioned before, the technology used for the database side for this project was PostgreSQL, which helped create and hold the database information, while also being compatible with the Java IDE and aiding the connection between those two.

The first step in creating the database is making the conceptual ER diagram for it. This diagram results forms the domain model and shows the information collected from the business requirements (Image 3.7). There are four entities, each describing the most important parts of the "Puppr" system: users, with their three different types, dogs, comments, and posts. The liking principle is modelled as many-to-many relationships between users and posts, comments, or dogs.



The principle of adding a comment is modelled by the ternary relationship between Users, Posts and Comments, due to the fact that a comment is linked not only to its author but to the post it was added to. The last two relationships, “made by” and “owned by” describe the bond between posts and users, respectively users and dogs.

Users entity has two special attributes made to distinguish between the three types of users (normal users, admins, and guests), and to take note of the blocked or unblocked status a user might have.

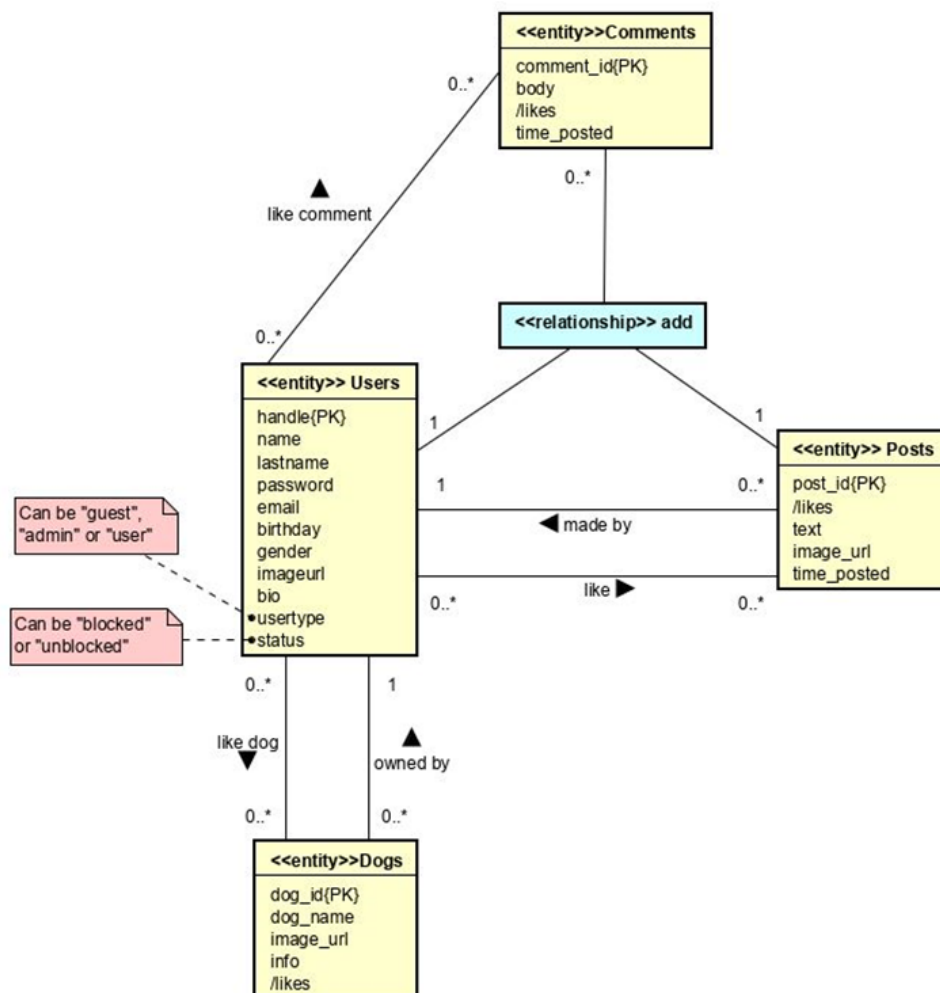
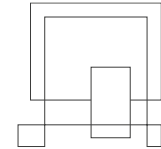


Image 3.7. Conceptual ER-Diagram



The second part of creating the database consisted of modelling and normalizing it in order to have as final product a structurally solid database, which respects all the business requirements. The visual representation of the normalization and mapping is the Global ER Diagram (Image 3.8).

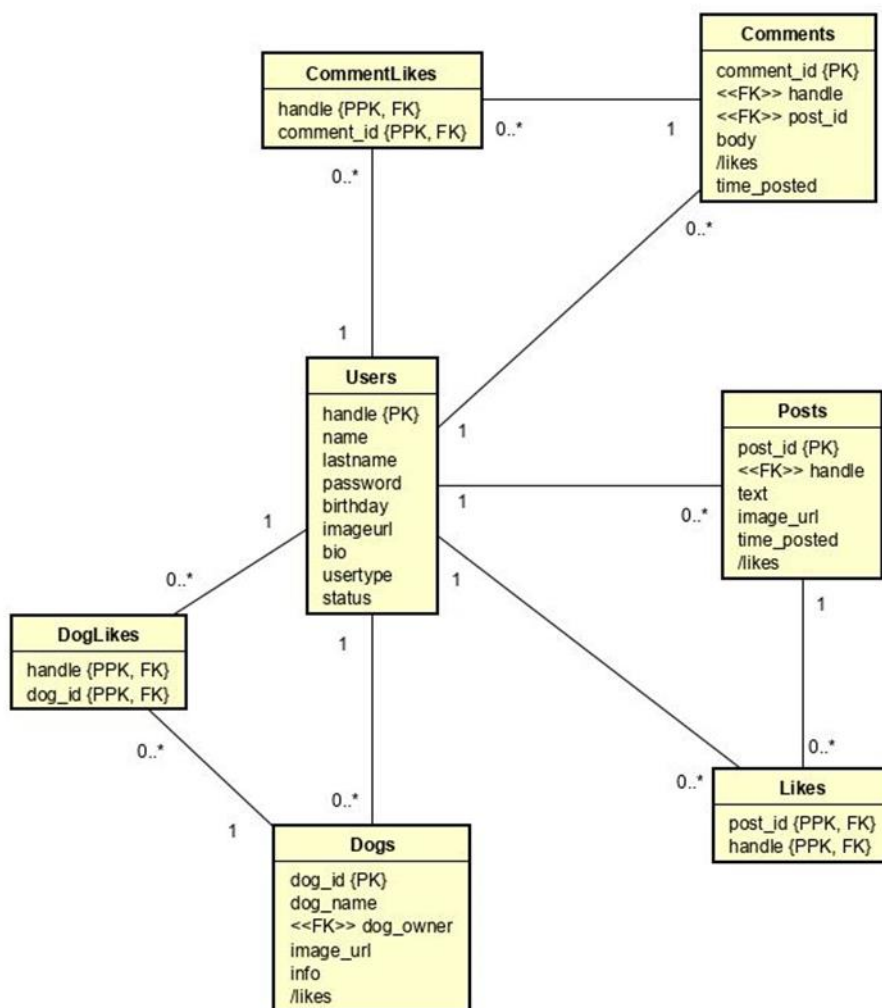
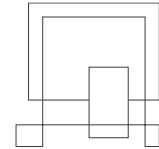


Image 3.8. Global ER Diagram

The four entities from the conceptual diagram were all strong entities and mapped directly into the relations named accordingly. After resolving the many-to-many relationship, the three like relationships became three different relationship relations, “Likes”, “DogLikes” and “CommentLikes”, with composed primary keys resulted from



the two primary keys of the main relations they link, also acting as foreign keys. The rest of the one-to-many relationships were resolved and foreign keys added in the Dogs, Posts and Comments relations.

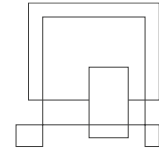
Before creating the final physical database for the system, the database model was normalized in order to remove any existing anomalies like partial and transitive dependencies. The resulted tables are all in third normal form because the data is in the table format, each table has a primary key, and all the other non-prime attributes in a table depend on the whole primary key and on nothing else.

This is most visible in the tables (Image 3.9) but also in the Global ER Diagram because all the attributes of the relations depend and describe only their own relation, for example in "Users" the name (first name of the user), last name, birthday, etc. are only about that specific user, hence no partial or transitive dependencies.

	dog_id [PK] bigint	dog_name character varying (50)	image_url bytea	info character varying (100)	dog_owner character varying (50)	likes numeric (10)
1	5	Doggo c:	[binary data]	i hab a doggooo	sunny	1
2	8	Max2	[null]	best puppr!!!	bbb	1
3	7	Max	[binary data]	My fluffy boy	admin	2
4	6	hund	[binary data]	helloooooooooooooooooooooo...	bbb	2

Image 3.9. Dogs table in the database

The result of the designing of the system can be visually interpreted as the Design Class Diagram which models all the objects and how they interact in a way that can be then directly implemented or translated into code. For this system, all the java objects, classes or interfaces, are represented in their own packages with all the different types of relationships such as associations, aggregations, etc., being mapped as well. The separation between the server and client side is also clearly made by splitting it into two diagrams, both containing the common part, the RemoteModel interface, which is the one that handles the communication between the two parts and ensures communication even when the parts are on separate computers.



In order to see the clearer flow of methods in the system for a particular activity, sequence diagrams can be created. These help with especially more complex paths, explaining them in greater detail and showing the timing between different method calls. Through the sequence diagrams the layer architecture of the system can also be seen by the numerous layers the method passes through before it reaches the database, the usual border of the system.

For example, when adding a new dog to their profile, the user's information is checked and sent to the local interface classes before reaching the remote interface and reaching the special class with a database connection which saves the information in the table (Image 3.10).

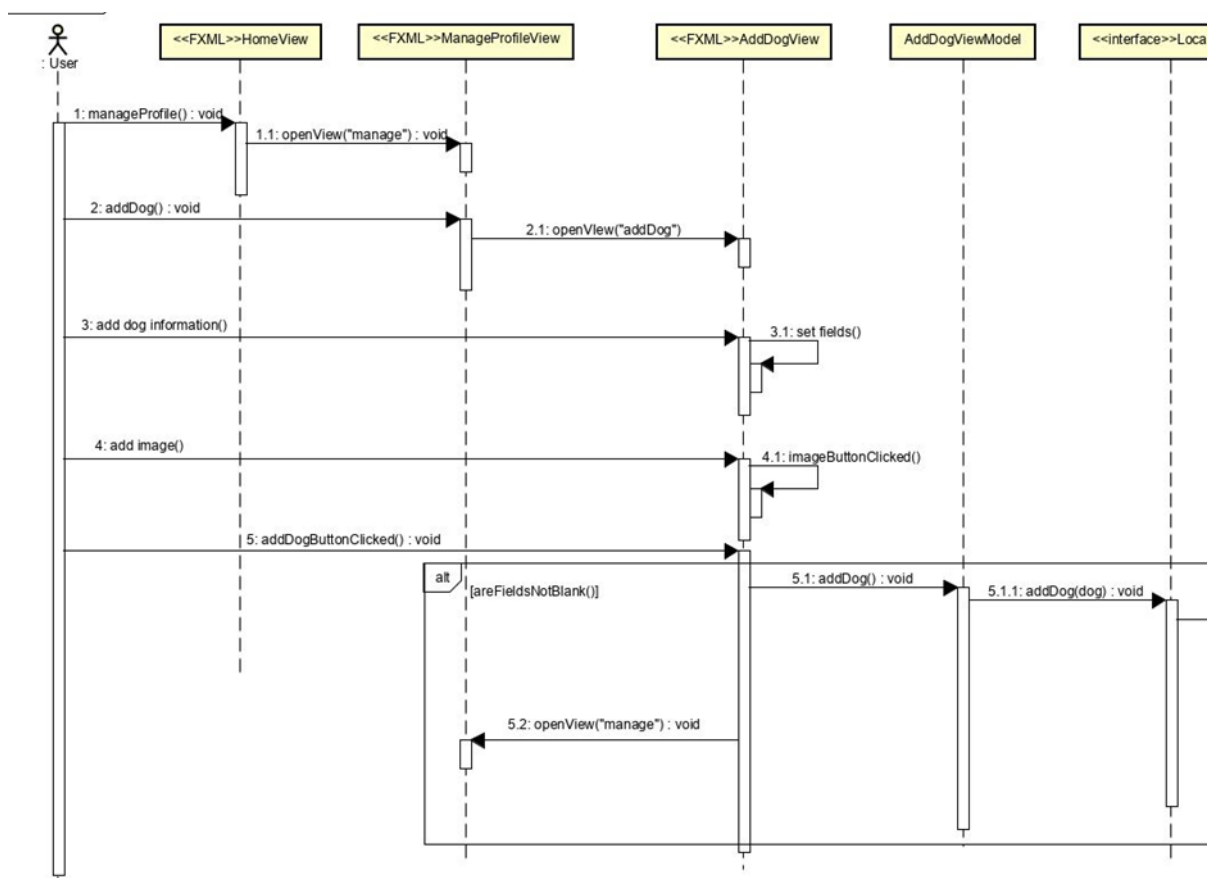
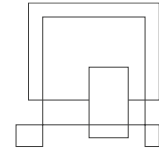


Image 3.10. AddDog sequence diagram



4 Implementation

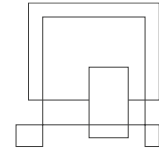
Natali Munk-Jakobsen

The most important part of the server side is the database connection. In the beginning of the database implementation, "postgresql-42.2.12.jar" was added as an external library, since POSTGRESQL is used for this project. The database package includes DatabaseConnection class and seven additional database classes. Each one of these classes contains a set of methods for adding, getting, updating and deleting related data.

DatabaseConnection class holds the values required to maintain a connection to the database and establishes the connection. *getConnection* method of the DriverManager class is called to create a connection to the PostgreSQL database server and JDBC URL, username and password values are used as method parameters.

The Singleton Pattern is used in databaseConnection class (Image 4.1). This pattern ensures that a class has only one instance and provides a global point of access to it. Thus, an instance is created only one time and the same instance is used every time needed in different parts of the system. A private constructor loads the driver and gets a connection. The database is called over this connection with *getInstance* method when it is needed. It provides easy and safe connection management.

Database classes use the same instance of databaseConnection and connect to the database. Each class includes several methods to link user input from the client side with the database by preparing and executing statements. These methods have similar structures in all classes. First, statements are prepared for deleting, inserting or updating table rows in the database or getting selected rows from the database tables. Prepared



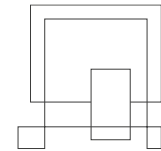
statements are then used as a parameter for calling *operation* method in the `DatabaseConnection` class.

```
7
8 public class DatabaseConnection
9 {
10
11     private static DatabaseConnection instance;
12     private Connection connection;
13     private String url = "jdbc:postgresql://localhost:5432/SEPDB";
14     private String username = "postgres";
15     private String password = "123456";
16
17     private DatabaseConnection() throws SQLException
18     {
19         try
20         {
21             Class.forName("org.postgresql.Driver");
22             this.connection = DriverManager.getConnection(url, username, password);
23             connection.setAutoCommit(false);
24             System.out.println("Database connected");
25         }
26         catch (ClassNotFoundException ex)
27         {
28             System.out.println("Database Connection Creation Failed : " + ex.getMessage());
29         }
30     }
31
32     public Connection getConnection() { return connection; }
33
34
35
36
37     public static DatabaseConnection getInstance() throws SQLException
38     {
39         if (instance == null)
40         {
41             instance = new DatabaseConnection();
42         }
43         else if (instance.getConnection().isClosed())
44         {
45             instance = new DatabaseConnection();
46         }
47
48         return instance;
49     }
50 }
```

Image 4.1. *DatabaseConnection Class and Singleton Design Pattern*

The `ModelManager` class is an implementation of the `Model` interface and has relations with all database classes. Therefore, it establishes connections between `Model` methods and database methods and helps the `RemoteModel` to perform changes to the database.

`RemoteModel` allows the client and the server to contact by using RMI connection. Both `RmiServer` and `RmiClient` classes implement `RemoteModel` interface that extends `RemoteSubject` interface. `RemoteSubject` contains *`addListener`* and *`removeListener`*



methods for observer design pattern and extends Remote interface. For this reason, all methods in RemoteModel interface declare RemoteException.

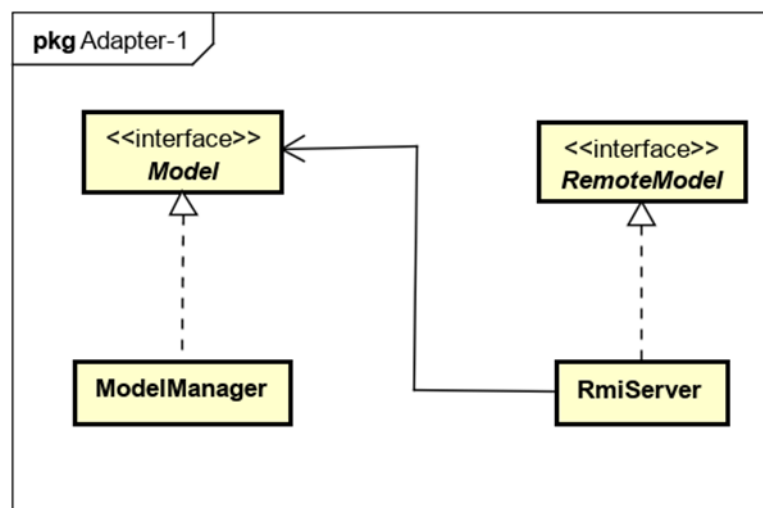


Image 4.2. Example Of Adapter Design Pattern-1

RmiServer class starts the registry, creates and uploads the stub to the registry and binds it to a name by means of its constructor and *startRegistry* method. Besides, it is the server implementation of the RemoteModel interface. As seen in Image 4.2, adapter design pattern is used for this implementation. In this pattern, RmiServer is an adapter and delegates adaptee part that consist of Model interface and ModelManager class.

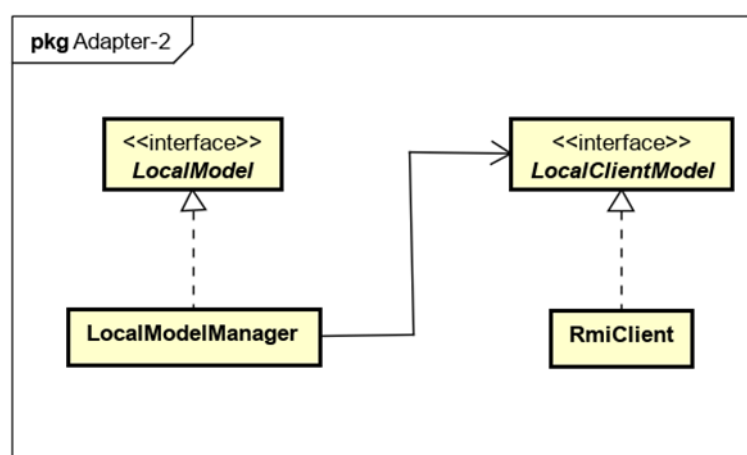
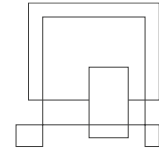


Image 4.3. Example Of Adapter Design Pattern-2



The adapter design pattern is also used for the implementation of LocalModel interface on the client side. (Image 4.3). LocalModel interface is target implementing by adapter class LocalModelManager. Adapter delegates LocalClientModel and RmiClient in the network package.

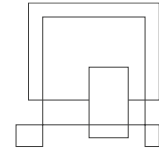
RmiClient class is responsible for establishing the connection with the server. This class contacts the registry by looking up the name of the server stub and then downloads it in order to call methods on server object remotely.

The model package on the client side includes LocalModel interface and LocalModelManager in addition to Comment, Dog, User and Post classes holding related data. These classes implement the java.io.Serializable interface because an object has to be serialized in order to be transmitted over RMI network connection. LocalModel interface is implemented by LocalModelManager class and establishes the connection between ViewModel and Model classes.

Model–View–ViewModel (MVVM) software architectural pattern is used for the separation of the model and the graphical user interface (the view). The ViewModel part consists of the ViewModelFactory class and 15 ViewModel class. The ViewModelFactory class is responsible for creating and returning instances of the ViewModel classes. The ViewModel classes indirectly notify the view via binding.

View package consists of ViewHandler class and 15 view controllers. ViewHandler is responsible for creating all views, and switching between them. When the user clicks a button to open a new view, *openView* method is called from ViewHandler class. For each ViewModel class, there is a View. GUI elements are bound to ViewModel properties.

Puppr application has 15 views. In order to be able to login to the system, users need to sign up in the system. After entering all data in the signupView, the user can log in and start to use all functions. In case of entering an incorrect username or password, the application shows an alert and does not allow the user to log in.



HomeView is the main view showing the latest posts and having “My profile”, “Manage Profile”, “The Team” and “Hall Of Fame” buttons. ManageProfileView enables the user to edit all user data except username (handle) since handle is planned as an unchangeable and unique identifier for each user. “Save” button calls *editUserInfo* method as long as form is complete. “Add Dog” opens AddDogView that is used for adding a new dog to the ProfileView.

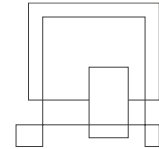
My Profile button directs the user to ProfileView presenting logged in user’s posts and dogs and allows the user to edit and delete them. HallOfFameView shows the top ten dogs based on the number of likes they have.

One of the most important functions of the application is adding posts to the main feed in HomeView. All logged-in users are allowed to use this function. An event handler is added to the post image in HomeView, therefore it can open a new view in case of mouse clicking. Post image directs the user to the CreatePostView.

CreatePostView is a simple view with a TextArea and a button related to *addPost* method which is responsible for adding posts. This method calls related methods from CreatePostViewModel, LocalModel and LocalClientModel in succession, transfers the post data to the server side through RMI connection and adds the data to the database.

The visual presentation of the user post is handled by *makeCard* method that connects several JavaFX elements with the given post object and returns a JavaFX Pane object. *AddLastPosts* method is responsible for placing the latest posts to HomeView. A similar method is used in ProfileView for showing the posts of profile owner.

Post cards allow the users to comment and like to the related post. Each post card contains two images associated with a event handler.

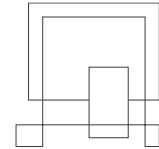


Comment image directs the user to `CreateCommentView` which contains a `TextArea` and `Send` button. `Send` button calls `addComment` method from the `viewmodel` and starts a procedure to add the comment to the database.

```
245
246
247 bone.addEventHandler(MouseEvent.MOUSE_CLICKED, event -> {
248
249     String verify = post.getId() + "/" + handleLabel.getText().replace(target: "@", replacement: "");
250
251     boolean found = false;
252     if(likedPosts!=null) {
253         for (String like : likedPosts) {
254             if (like.equals(verify)) {
255                 found = true;
256                 break;
257             }
258         }
259     }
260     if (!found) {
261         viewModel.likePost(post.getId(), handleLabel.getText().replace(target: "@", replacement: ""));
262         vbox.getChildren().clear();
263         addLastPosts();
264         likedPosts = viewModel.getAllLikes();
265         System.out.println(likedPosts);
266     }
267 });
268 }
```

Image 4.4. Verification of post likes

The user clicks the bone image to like a post. An event handler is added to this image and in case of mouse clicking, it calls `likePost` method if the necessary condition is fulfilled. Each user is allowed to like a post only once and this condition is checked through a boolean variable (Image 4.4). Liked post and logged in user relation data is held in the database and used for checking if a user already liked the given post. `likedPosts` is an `ArrayList` of strings and it is initialized in the constructor of `HomeViewController` by calling `getAllLikes` method from `ViewModel`. `getAllLikes` method returns an `ArrayList` of `String` that includes post and user data. When the user likes a post, a verification string is created and if this string is an element of the `ArrayList`, then `likePost` method is called. `likePost` method is responsible for adding data about 'liked post' and 'logged in user' relation to the database and also increasing the number of likes for the post.



The users can see comments through ViewPostView that is opened by show button in the post card. Comment cards are shown under the main post in the ViewPostView . Comment cards are very similar with post cards except having a comment button since the users are not able to write comments to other users' comments. In a similar way as post like, a verification string is used to check user likes for the comment before adding a like for it.

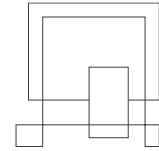
Beside commenting, liking and viewing related post, post cards also allow the user to visit the post owner's profile. User photo and name in the post card directs the users to the related profile.

ProfileView is reachable from two different parts of the application. Firstly, each user can see their own user profiles by clicking "My Profile" button. This button opens profileView and also fires an event. ProfileViewModel listens to this event, updates properties in case of having "user" property name and as a consequence bound JavaFX components in ProfileView are formed according to logged in user data.

Secondly, all users can reach other users' profiles by clicking their user photos or names. In these cases, profileView is opened and an event with "otherUser" property name is fired. This event ensures that bound properties in ProfileViewModel and JavaFx components in ProfileView are updated according to intended user data.

When ProfileView is opened as logged in user, special parts in post cards become visible and available. These parts allow the user to delete or edit selected posts. This distinction handled by Boolean mainUserPage field which gets different values based on the property name when listening to fired events.

Puppr application can be used as a guest without logging in. Guests have limited access to some Puppr functions, such as viewing posts, comments and user profiles. Nevertheless, they do not have permission to write posts or comments and use like buttons. The presence of guest users is controlled with Boolean isGuest variable and



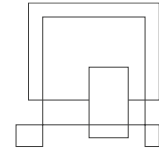
isGuest method with Boolean return type in the code. In case of a guest visits the application, restricted parts become invisible or an alert is shown when clicking nonadmitted part.

Admin is a privileged user with additional access authorization. In a similar way as managing guests, *isAdmin* boolean variable and *isAdmin* method are used for handling admin users. In addition to all basic user functions, admin can edit or delete other users' posts, comments or dogs and block/unblock the users.

```
548 public void blockUser() {  
549     User user = viewModel.getUserByHandle(viewModel.getHandle().replace( target: "@", replacement: ""));  
550     if (user.getUserType().equals("user")) {  
551         if (user.getStatus().equalsIgnoreCase( anotherString: "unblocked")) {  
552             viewModel.blockUser();  
553             block.setText("UNBLOCK USER");  
554         } else {  
555             viewModel.blockUser();  
556             block.setText("BLOCK USER");  
557         }  
558     }  
559 }  
560
```

Image 4.5. *blockUser* method

When an admin visits the profileView, “Block” button becomes visible in the view and can be used for calling *blockUser* method (Image 4.5) . This method blocks the owner of the visited profile and sets button text as “unblock” in order to allow the admin to unblock the user when it is needed. *BlockUser* method changes user status to “blocked” and a blocked user cannot log in to the application until being unblocked by an admin.



```
22 public static byte[] ImageToByte(File file) throws FileNotFoundException {
23     FileInputStream fis = new FileInputStream(file);
24     ByteArrayOutputStream bos = new ByteArrayOutputStream();
25     byte[] buf = new byte[1024];
26     try{
27         for(int x; (x=fis.read(buf)) != -1 ;){
28             bos.write(buf, off: 0, x);
29         }
30     } catch (IOException e) {
31         e.printStackTrace();
32     }
33     return bos.toByteArray();
34 }
```

Image 4.6. ImageToByte method

The users can add their own photos and their dogs' photos to their profile. They can also add images to the posts. Images are held as byte arrays (byte[]) in the database. Therefore image files have to be converted to byte array in order to be added in the database. *ImageToByte* method is used for this conversion (Image 4.6). Since this method is needed in different parts of the system, *ImageConverted* class is created to hold it in order to avoid code duplication. The method is static to ensure easier usage in other classes.

Since Puppr is designed as "a dog owner socialising platform", dogs have an important part in the user profiles. Dogs are shown through dog cards that are formed by *makePupprCard* method. This method creates a JavaFX Pane for given dog data. Dog cards present dog info and logged in users can edit or delete their dogs with the buttons on the card. Dog cards also enable the user to like the selected dog through the like button. Number of likes for dogs is used for forming *HallOfFameView*.

Observer design pattern is often used for handling updates in view and viewmodel parts. *LocalListener* and *LocalSubject* interfaces from "MyObserver.jar" are used for notifying other classes in case of any update. When a user makes a change in a view, an event is fired and a *View* or *ViewModel* class listens to this event.

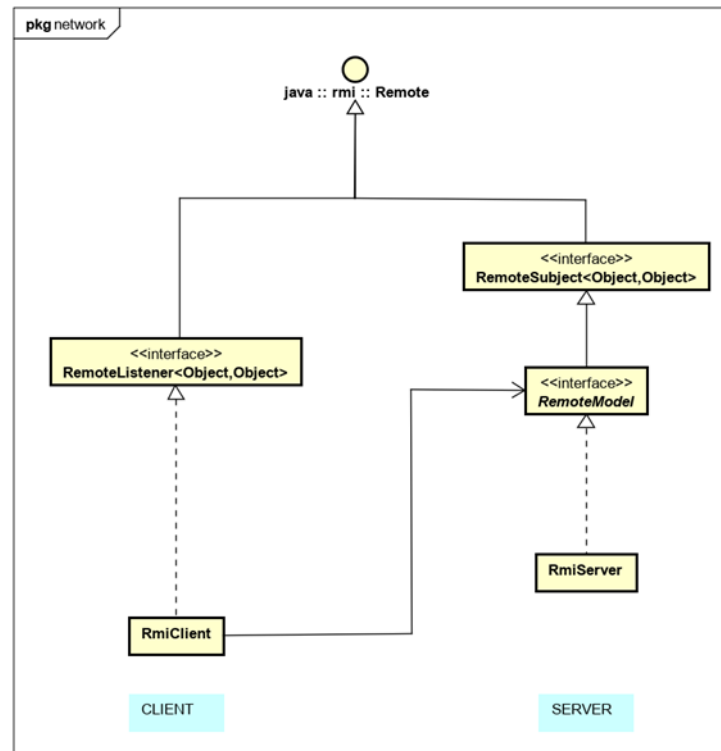
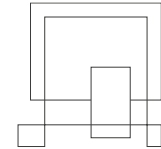
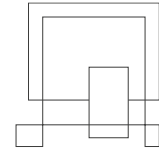


Image 4.7. Usage of RemoteSubject and RemoteListener interfaces

In addition, RMI connection between server and client is built with observer design pattern. RemoteListener and RemoteSubject interfaces from “MyObserver.jar” have a part in RMI connection. (Image 4.7)



5 Test

Bogdan-Alexandru Mezei

5.1 Testing Methodology

When starting to test the software, nine main test scenarios were created which ensured complete test coverage. After that, for each test scenario, multiple test cases (each with their own unique assigned test case ID) were created in accordance with the ZOMBIES framework. Lastly, in order to ensure that all the necessary requirements have been met, all the information gained from before has been neatly packed into a requirement traceability matrix.

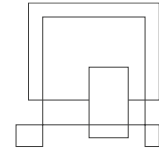
For the “Puppr” project the following testing methods have been picked:

- Black Box Testing
 - System Tests
 - Acceptance Tests
- White Box Testing
 - Unit Tests using the ZOMBIES framework.

The application has been tested on three different clients at the same time.

5.2 Test Scenarios

1. Checking the signup functionality.
2. Checking the login functionality.
3. Checking the posting functionality.
4. Checking the commenting functionality.
5. Checking the system behaviour while managing dogs.
6. Checking the liking functionality.

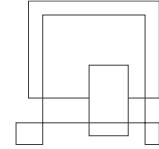


7. Checking the system behaviour while managing personal details.
8. Checking the system behaviour while moderating with admin privileges.
9. Checking the "Hall of Fame" feature.

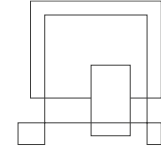
5.3 Test Cases

Table 5.1. Test Scenario #1 - Checking the signup functionality

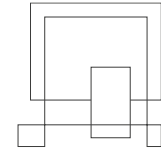
Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Grade
TC11	Checking the creation of an account with valid data.	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Sign up". 3. Enter all the necessary data. 4. Click "Sign up". 	<p>First Name = Jane</p> <p>Last Name = Doe</p> <p>Handle = @janedoe</p> <p>Email = janedoe@gmail.com</p> <p>Password = keyboard</p> <p>Personal Description = Hi! My name is Jane.</p> <p>Date of Birth = 14-05-1998</p>	A new account is created and all the inputted data remains the same.	As expected.	Pass



			Gender = Female			
			Profile Picture = Uploaded			
TC12	Checking the creation of an account with incomplete data.	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Sign up". 3. Enter all the necessary data. 4. Click "Sign up". 	First Name = Last Name = Handle = @janedoe Email = Password = puppy123 Personal Description = Hi! My name is Jane. Date of Birth = 14-05-1998 Gender = Female Profile Picture = 	The application throws an error and prompts the user to fill in the obligatory fields which get highlighted in red	As expected	Pass



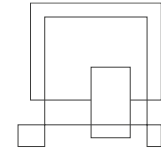
			Uploaded			
TC13	Checking the creation of an account when the user does not provide a profile picture.	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Sign up". 3. Enter all the necessary data. 4. Click "Sign up". 	<p>First Name = Jane</p> <p>Last Name = Doe</p> <p>Handle = @janedoe</p> <p>Email = janedoe@gmail.com</p> <p>Password = keyboard</p> <p>Personal Description = Hi! My name is Jane.</p> <p>Date of Birth = 14-05-1998</p> <p>Gender = Female</p> <p>Profile Picture = Not Uploaded</p>	The account is created and a default profile picture is assigned.	As expected.	Pass
TC14	Attempting to	1. Open the	Name	The	As	Pass



	fill in a text field with more characters than the maximum allowed.	application. 2. Click "Sign up". 3. Attempt entering a very long string as one of your personal details.	= aaa...aaa (over 50 characters long)	application automatically cuts the string to the maximum size and does not allow going past that point.	expected.	
--	---	--	---	---	-----------	--

Table 5.2. Test Scenario #2 - Checking the login functionality

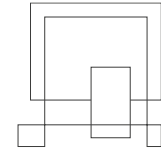
Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Grade
TC21	Checking if an existing user can log in with valid credentials	1. Open the application. 2. Enter username and password 3. Click "Log in".	Username = TrashPanda Password = rac00n	The user is logged in.	As expected.	Pass
TC22	Checking system behavior when a user inputs a valid username but an invalid password.	1. Open the application. 2. Enter username and password 3. Click "Log in".	Username = TrashPanda Password = wr0ng	The user is not logged in and an error message is displayed.	As expected.	Pass
TC23	Checking system behavior when a user inputs an invalid username but a valid password.	1. Open the application. 2. Enter username and password 3. Click "Log in".	Username = idontexist Password = rac00n	The user is not logged in and an error message is displayed.	As expected.	Pass



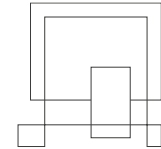
TC24	Checking system behavior when a user inputs an invalid email and password.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 	Username = idontexist Password = wr0ng	The user is not logged in and an error message is displayed.	As expected.	Pass
TC25	Checking system behavior when a user logs in as a guest	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Continue as a guest" 		The user is taken to the home page as a guest.	As expected.	Pass

Table 5.3. Test Scenario #3 - Checking the posting functionality

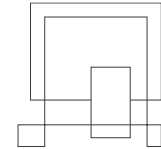
Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Grade
TC31	Checking if a logged-in user can create a text post.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click the create post button. 5. Type the text you want to post. 6. Click "Post". 	Username = TrashPanda Password = rac00n Text = This is my first post! Photo = Not uploaded	The user is taken back to the home screen and the just created text post is visible.	As expected.	Pass
TC32	Checking if a	1. Open the	Username	The user is	As	Pass



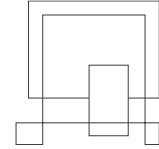
	logged-in user can create a photo post.	<ol style="list-style-type: none"> 1. application. 2. Enter username and password 3. Click "Log in". 4. Click the create post button. 5. Type the text you want to post. 6. Upload an image. 7. Click "Post". 	<p>= TrashPanda</p> <p>Password = rac00n</p> <p>Text = This is my first post!</p> <p>Photo = Uploaded</p>	taken back to the home screen and the just created photo post is visible.	expected.	
TC33	Checking if the scroll pane displays correctly while there are no posts.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p>	The scroll pane appears empty.	As expected.	Pass
TC34	Checking if the scroll pane displays correctly with a single post.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>One previously created post.</p>	The scroll pane displays the post but the user cannot scroll.	As expected.	Pass
TC35	Checking if the scroll pane displays correctly with multiple posts.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p>	The scroll pane displays all the posts at once and the user is able to scroll through all of them.	As expected.	Pass



			Multiple previously created posts.			
TC36	Checking if a user can create a text post with more characters than the maximum allowed.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click the create post button. 5. Type the text you want to post. 6. Click "Post". 	Username = TrashPanda Password = rac00n Text = aaaa...aaa (over 140 characters) Photo = Not uploaded	The application automatically cuts the string to the maximum size and does not allow going past that point.	As expected.	Pass
TC37	Checking if a user can create a photo post with a description that has more characters than the maximum allowed.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click the create post button. 5. Type the text you want to post. 6. Upload an image. 7. Click "Post". 	Username = TrashPanda Password = rac00n Text = aaaa...aaa (over 140 characters) Photo = Uploaded	The application automatically cuts the string to the maximum size and does not allow going past that point.	As expected.	Pass
TC38	Checking if a text post is displayed correctly when clicking	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and 	Username = TrashPanda Password	The text post is opened and displayed correctly on the left side of	As expected.	Pass



	the "show" button.	3. password Click "Log in". 4. Click "Show".	= rac00n One previously created text post.	a new window.		
TC39	Checking if a photo post is displayed correctly when clicking the "show" button.	1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Show".	Username = TrashPanda Password = rac00n One previously created photo post.	The photo post is opened and displayed correctly on the left side of a new window.	As expected.	Pass
TC310	Checking if a text post is displayed correctly when clicking the "show" button while logged in as guest.	1. Open the application. 2. Click "Continue as a guest" 3. Click "Show".	One previously created text post.	The text post is opened and displayed correctly on the left side of a new window.	As expected.	Pass
TC311	Checking if a photo post is displayed correctly when clicking the "show" button while logged in as guest.	1. Open the application. 2. Click "Continue as a guest" 3. Click "Show".	One previously created photo post.	The photo post is opened and displayed correctly on the left side of a new window.	As expected.	Pass
TC312	Checking system behavior when	1. Open the application. 2. Enter username	Multiple posts of any kind previously created by the	All of the user's posts are displayed on their	As expected.	Pass



	displaying posts on a profile page. (personal feed)	<ol style="list-style-type: none"> 3. Click "Log in". 4. Click on someone's name to be taken to their personal page. 	user.	personal feed on their profile page in the center.		
TC313	Checking if a logged-in user can edit his own text post.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Edit" next to one of your posts. 5. Type in the new post. 6. Click "Post". 	Username = TrashPanda Password = rac00n Old post = I love cats! New post = I love dogs!	The post is edited.	As expected.	Pass
TC314	Checking if a logged-in user can edit his own photo post.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Edit" next to one of your posts. 5. Click "Edit Photo". 6. Upload the new picture. 7. Click "Post". 	Username = TrashPanda Password = rac00n Old photo = cat.png New photo = dog.png	The photo in the post is edited.	As expected.	Pass

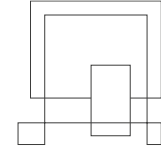
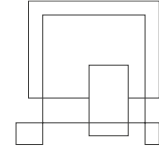
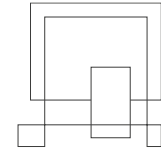


Table 5.4. Test Scenario #4 - Checking the commenting functionality

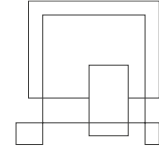
Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Grade
TC41	Checking if a logged-in user can create a comment on an existing post from the home screen.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click the comment button on an existing post. 5. Type the comment you want to post. 6. Click "Send". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>Comment = This is my first comment!</p> <p>A previously created post.</p>	The user is taken back to the home screen and the comment is successfully created and added to the right post.	As expected.	Pass
TC42	Checking if a logged-in user can create a comment on an existing post from the "show post" screen.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Show". 5. Click the comment button. 6. Type the comment you want to post. 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>Comment = This is my first comment!</p> <p>A previously created post.</p>	The user is taken back to "show post" screen and the comment is successfully created and added to the right post.	As expected.	Pass



		7. Click "Send".				
TC43	Checking if a user logged in as guest can create a comment from the home screen.	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Continue as a guest" 3. Click the comment button. 	A previously created post.	An alert is shown to the user giving him the option to log in or continue as a guest. In either case, no comment is created.	As expected.	Pass
TC44	Checking if a user logged in as guest can create a comment from the "show post" screen.	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Continue as a guest" 3. Click "Show". 4. Click the comment button. 	A previously created post.	An alert is shown to the user giving him the option to log in or continue as a guest. In either case, no comment is created.	As expected.	Pass
TC45	Checking if a logged-in user can create a comment that has more characters than the maximum allowed.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click the comment button. 5. Type the comment you want to post. 6. Click "Send". 	Username = TrashPanda Password = rac00n Comment = aaaa...aaa (over 140 characters)	The application automatically cuts the string to the maximum size and does not allow going past that point.	As expected.	Pass



TC46	Checking if it is possible for a logged-in user to edit their own comment.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Show" next to a post you have previously commented on. 5. Click "Edit" next to the comment you want to edit. 6. Type in the new comment. 7. Click "Send". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>Old comment = Hi Karen!</p> <p>New comment = Hi Felicia!</p> <p>A previously created comment on an existing post.</p>	The comment is edited.	As expected.	Pass
TC47	Checking if it is possible for a logged-in user to delete their own comment.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Show" next to a post you have previously commented on. 5. Click "Delete" next to the 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>A previously created comment on an existing post.</p>	The comment is deleted.	As expected.	Pass



		comment you want to delete.				
TC48	Checking if the comments section is displayed correctly on a post without comments.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Show". 	A previously created post without comments.	The comment section is correctly displayed as empty.	As expected.	Pass
TC49	Checking if the comments section is displayed correctly on a post with one comment.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Show". 	A previously created post with one comment.	The comment section is displayed correctly along with the sole comment.	As expected.	Pass
TC410	Checking if the comments section is displayed correctly on a post with multiple comments.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Show". 	A previously created post with multiple comments.	The comment section is displayed correctly and the user is able to scroll in order to view all the comments.	As expected.	Pass

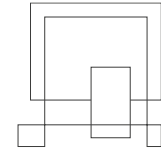
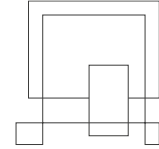
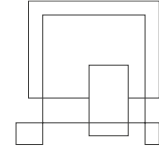


Table 5.5. Test Scenario #5 - Checking the system behavior while managing dogs

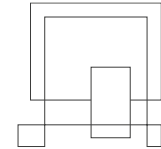
Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Grade
TC51	Checking if the dog list in the profile page displays correctly while there are no dogs.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "My Profile". 	Username = TrashPanda Password = rac00n	The scroll pane is shown as empty and no error is thrown.	As expected.	Pass
TC52	Checking if the dog list in the profile page displays correctly with one dog.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "My Profile". 	Username = TrashPanda Password = rac00n A previously added dog.	The scroll pane displays the dog but scrolling is not possible.	As expected.	Pass
TC53	Checking if the dog list in the profile page displays correctly with multiple dogs.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "My Profile". 	Username = TrashPanda Password = rac00n Multiple previously added dogs.	The scroll pane displays the dogs and the user is able to scroll in order to view all of them.	As expected.	Pass
TC54	Checking if deleting a previously added dog is possible.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 	Username = TrashPanda Password =	The dog is deleted and removed from the scroll pane.	As expected.	Pass



		<ol style="list-style-type: none"> Click "Log in". Click "My Profile". Click "Delete". 	<p>rac00n</p> <p>A previously added dog.</p>			
TC55	Checking if adding a new dog with a picture to your profile is possible.	<ol style="list-style-type: none"> Open the application. Enter username and password Click "Log in". Click "Manage Profile". Click "Add Dog". Enter the necessary information about the dog. Click "Set Image". Select the image you want to upload. Click "Add". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>Dog Name = Rex</p> <p>Dog Description = He's the best boy!</p> <p>Dog Picture = Uploaded</p>	The dog is added and can be seen in your profile.	As expected.	Pass
TC56	Checking if adding a new dog without a picture to your profile is possible.	<ol style="list-style-type: none"> Open the application. Enter username and password Click "Log in". Click "Manage Profile". Click "Add Dog". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>Dog Name = Rex</p> <p>Dog</p>	The dog is added with a default picture and can be seen in your profile.	As expected.	Pass



		6. Enter the necessary information about the dog. 7. Click "Add".	Description = He's the best boy! Dog Picture = Not uploaded			
TC57	Checking system behavior when trying to input a string that is longer than the maximum allowed for the name of the dog or the description.	1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Manage Profile". 5. Click "Add Dog". 6. Enter the necessary information about the dog. 7. Click "Add".	Username = TrashPanda Password = rac00n Dog Name = aaa...aaa (over 12 characters) Dog Description = aaa...aaa (over 100 characters) Dog Picture = Uploaded	The application automatically cuts the strings to the maximum size and does not allow going past that point.	As expected.	Pass
TC58	Checking system behavior when trying to edit the name of an already existing dog.	1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "View Profile". 5. Click "Edit" next to the	Username = TrashPanda Password = rac00n New dog name = Cupcake	The dog's name is changed to the new one.	As expected.	Pass



		<p>dog you want to edit.</p> <p>6. Type in the new name.</p> <p>7. Click "Edit".</p>	A previously added dog.			
TC59	Checking system behavior when trying to edit the description of an already existing dog	<p>1. Open the application.</p> <p>2. Enter username and password</p> <p>3. Click "Log in".</p> <p>4. Click "View Profile".</p> <p>5. Click "Edit" next to the dog you want to edit.</p> <p>6. Type in the new description.</p> <p>7. Click "Edit".</p>	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>New dog description = I love him so much!</p> <p>A previously added dog.</p>	The dog's description is changed to the new one.	As expected.	Pass
TC510	Checking system behavior when trying to edit the photo of an already existing dog	<p>1. Open the application.</p> <p>2. Enter username and password</p> <p>3. Click "Log in".</p> <p>4. Click "View Profile".</p> <p>5. Click "Edit" next to the dog you want to edit.</p> <p>6. Click "Set Image".</p> <p>7. Upload the new image.</p> <p>8. Click "Edit".</p>	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>New dog image = newImage.png</p> <p>A previously added dog.</p>	The dog's photo is changed to the new one.	As expected.	Pass

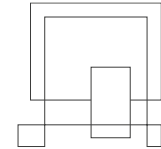
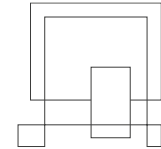
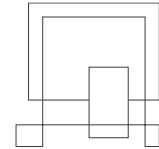


Table 5.6. Test Scenario #6 - Checking the liking functionality

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Grade
TC61	Checking if a logged-in user can like a post from the home screen.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click the like button next to the post you want to like. 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>A previously created post.</p>	The post is liked and the bone icon fills in to notify the user.	As expected.	Pass
TC62	Checking if a logged-in user can like a post from the "show post" screen.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Show". 5. Click the like button next to the post you want to like. 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>A previously created post.</p>	The post is liked and the bone icon fills in to notify the user.	As expected.	Pass
TC63	Checking system behavior when a guest tries to like a post from the home screen.	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Continue as a guest" 3. Click the like button next to the post you want to like. 	A previously created post.	The post is not liked and an alert is shown to the user with the options of logging in or continuing as guest.	As expected.	Pass



TC64	Checking system behavior when a guest tries to like a post from the home screen.	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Continue as a guest" 3. Click "Show". 4. Click the like button next to the post you want to like. 	A previously created post.	The post is not liked and an alert is shown to the user with the options of logging in or continuing as guest.	As expected.	Pass
TC65	Checking if a logged-in user can like a dog on a profile.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click on someone's name in order to be taken to their personal profile. 5. Click the like button next to the dog you want to like. 	A previously added dog on any profile.	The dog is liked and the bone icon fills in to notify the user.	As expected.	Pass
TC66	Checking system behavior when a guest tries to like a dog on any profile.	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Continue as a guest" 3. Click "Show". 4. Click the like button next to the post you 	Username = TrashPanda Password = rac00n A previously added dog on any profile.	The post is not liked and an alert is shown to the user with the options of logging in or continuing as guest.	As expected.	Pass



		want to like.				
TC67	Checking if a logged-in user can like a comment.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Show". 5. Click the like button next to the comment you want to like. 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>A previously added comment to an already existing post.</p>	The comment is liked and the bone icon fills in to notify the user.	As expected.	Pass
TC68	Checking system behavior when a guest tries to like a comment.	<ol style="list-style-type: none"> 1. Open the application. 2. Click "Continue as a guest" 3. Click "Show". 4. Click the like button next to the comment you want to like. 	A previously created post with at least one comment.	The comment is not liked and an alert is shown to the user with the options of logging in or continuing as guest.	As expected.	Pass

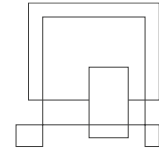
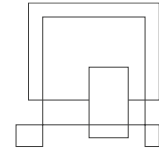
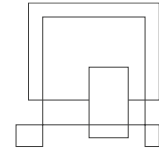


Table 5.7. Test Scenario #7 - Checking the system behavior while managing personal details

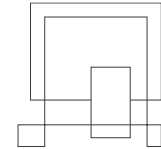
Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Grade
TC71	Checking if changing the first name of a logged-in user is possible.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Manage Profile". 5. Change the first name to the new one. 6. Click "Save". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>First Name = Bogdan</p> <p>New First Name = Alexandru</p>	The first name of the user is changed.	As expected.	Pass
TC72	Checking if changing the last name of a logged-in user is possible.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Manage Profile". 5. Change the last name to the new one. 6. Click "Save". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p> <p>Last Name = Mezei</p> <p>New Last Name = Ivascu</p>	The last name of the user is changed.	As expected.	Pass
TC73	Checking if	1. Open the	Username	The email of	As	Pass



	changing the email of a logged-in user is possible.	application. 2. Enter username and password 3. Click "Log in". 4. Click "Manage Profile". 5. Change the email to the new one. 6. Click "Save".	= TrashPanda Password = rac00n Email = bogdan.mezei@gmail.com New Email = bogdan335@hotmail.com	the user is changed.	expected.	
TC74	Checking if changing the description of a logged-in user is possible.	1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Manage Profile". 5. Change the description to the new one. 6. Click "Save".	Username = TrashPanda Password = rac00n Description = This is my first description! New Description = This is my new and improved description!	The description of the user is changed.	As expected.	Pass
TC75	Checking if changing the date of birth	1. Open the application. Enter	Username = TrashPanda	The birthdate of the user is changed.	As expected.	Pass



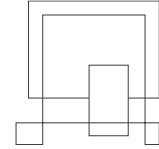
	of a logged-in user is possible.	username and password 3. Click "Log in". 4. Click "Manage Profile". 5. Change the birthdate to the new one using the date picker. 6. Click "Save".	Password = rac00n Birthdate = 10.01.2000 New Birthdate = 20.04.2001			
TC76	Checking if changing the gender of a logged-in user is possible.	1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Manage Profile". 5. Select the new gender. 6. Click "Save".	Username = TrashPanda Password = rac00n Gender = Male New Gender = Female	The gender of the user is changed.	As expected.	Pass
TC77	Checking if changing the profile picture of a logged-in user is possible.	1. Open the application. 2. Enter username and password 3. Click "Log in". 4. Click "Manage Profile". 5. Click "Upload Image".	Username = TrashPanda Password = rac00n Profile Picture = firstPic.png New Profile Picture	The profile picture of the user is changed.	As expected.	Pass



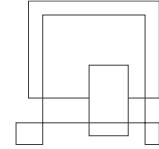
		6. Upload the new picture 7. Click "Save".	= finalPic.jpg			
--	--	---	-------------------	--	--	--

Table 5.8. Test Scenario #8 - Checking the system behavior while moderating with admin privileges

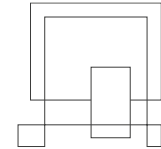
Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Grade
TC81	Checking if it is possible for an admin to block an account.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter admin username and password 3. Click "Log in". 4. Click on someone's name in order to be taken to their profile. 5. Click "Block User". 	Username = root Password = toor A previously created account.	The user is blocked. When trying to log in as a blocked user an alert is shown informing the user that the account is blocked.	As expected.	Pass
TC82	Checking if it is possible for an admin to delete a post from the home screen.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter admin username and password 3. Click "Log in". 4. Click "Delete" next to the post you want to delete. 	Username = root Password = toor A previously created post.	The post is deleted.	As expected.	Pass



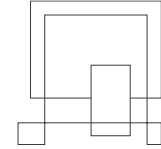
TC83	Checking if it is possible for an admin to delete a post from the profile page.	<ol style="list-style-type: none"> Open the application. Enter admin username and password Click "Log in". Click on someone's name in order to be taken to their profile. Click "Delete" next to the post you want to delete. 	<p>Username = root</p> <p>Password = toor</p> <p>A previously created post.</p>	The post is deleted.	As expected.	Pass
TC84	Checking if it is possible for an admin to edit a post from the home screen.	<ol style="list-style-type: none"> Open the application. Enter admin username and password Click "Log in". Click "Edit" next to the post you want to edit. 	<p>Username = root</p> <p>Password = toor</p> <p>A previously created post.</p>	The post is edited.	As expected.	Pass
TC85	Checking if it is possible for an admin to edit a post from the profile page.	<ol style="list-style-type: none"> Open the application. Enter admin username and password. Click "Log in". Click on someone's name in order to be 	<p>Username = root</p> <p>Password = toor</p> <p>A previously created post.</p>	The post is edited.	As expected.	Pass



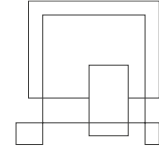
		taken to their profile. 5. Click "Edit" next to the post you want to Edit.				
TC86	Checking if it is possible for an admin to edit a comment.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter admin username and password. 3. Click "Log in". 4. Click "Show". 5. Click "Edit" next to the comment you want to edit. 6. Type in the new comment. 7. Click "Send". 	<p>Username = root</p> <p>Password = toor</p> <p>Old comment = Hi Karen!</p> <p>New comment = Hi Felicia!</p> <p>A previously created comment on an existing post.</p>	The comment is edited.	As expected.	Pass
TC87	Checking if it is possible for an admin user to delete a comment.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter admin username and password 3. Click "Log in". 4. Click "Show". 5. Click "Delete" next to the comment you want to delete. 	<p>Username = root</p> <p>Password = toor</p> <p>A previously created comment on an existing post.</p>	The comment is deleted.	As expected.	Pass



TC88	Checking if it is possible for an admin to unblock a previously blocked user.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter admin username and password 3. Click "Log in". 4. Click on someone's name in order to be taken to their profile. 5. Click "Unblock User". 	<p>Username = root</p> <p>Password = toor</p> <p>A previously blocked account.</p>	The account is unblocked and can now log back in.	As expected.	Pass
TC89	Checking if an admin can create a text post.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter admin username and password 3. Click "Log in". 4. Click the create post button. 5. Type the text you want to post. 6. Click "Post". 	<p>Username = root</p> <p>Password = toor</p> <p>Text = This is my first admin post!</p> <p>Photo = Not uploaded</p>	The admin is taken back to the home screen and the just created text post is visible.	As expected.	Pass
TC810	Checking if an admin can comment.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter admin username and password 3. Click "Log 	<p>Username = root</p> <p>Password = toor</p>	The admin is taken back to the home screen and the comment is successfully	As expected.	Pass



		<p>in".</p> <p>4. Click the comment button on an existing post.</p> <p>5. Type the comment you want to post.</p> <p>6. Click "Send".</p>	<p>Comment =</p> <p>This is my first admin comment!</p> <p>A previously created post.</p>	created and added to the right post.		
TC811	Checking if an admin can like a post.	<p>1. Open the application.</p> <p>2. Enter admin username and password</p> <p>3. Click "Log in".</p> <p>4. Click the like button next to the post you want to like.</p>	<p>Username = root</p> <p>Password = toor</p> <p>A previously created post.</p>	The post is liked and the bone icon fills in to notify the admin.	As expected.	Pass
TC812	Checking if an admin can like a comment.	<p>1. Open the application.</p> <p>2. Enter admin username and password</p> <p>3. Click "Log in".</p> <p>4. Click "Show".</p> <p>5. Click the like button next to the comment you want to like.</p>	<p>Username = root</p> <p>Password = toor</p> <p>A previously added comment to an already existing post.</p>	The comment is liked and the bone icon fills in to notify the admin.	As expected.	Pass
TC813	Checking if an admin can	1. Open the application.	Username =	The dog is added and	As expected.	Pass



	add a dog to their profile.	<ol style="list-style-type: none"> 2. Enter admin username and password 3. Click "Log in". 4. Click "Manage Profile". 5. Click "Add Dog". 6. Enter the necessary information about the dog. 7. Click "Set Image". 8. Select the image you want to upload. 9. Click "Add". 	<p>root</p> <p>Password = toor</p> <p>Dog Name = Rex</p> <p>Dog Description = He's the best boy!</p> <p>Dog Picture = Uploaded</p>	can be seen in your profile.		
TC814	Checking if an admin can create a photo post.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter admin username and password 3. Click "Log in". 4. Click the create post button. 5. Type the text you want to post. 6. Upload an image. 7. Click "Post". 	<p>Username = root</p> <p>Password = toor</p> <p>Text = This is my first post!</p> <p>Photo = Uploaded</p>	The admin is taken back to the home screen and the just created photo post is visible.	As expected.	Pass

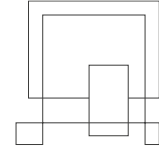
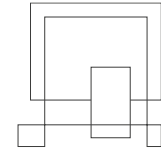


Table 5.9. Test Scenario #9 - Checking the "Hall of Fame" feature

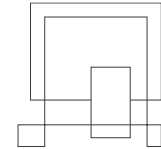
Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Grade
TC91	Checking if the top 10 most liked dogs on the platform are shown correctly on the "Hall of Fame" page.	<ol style="list-style-type: none"> 1. Open the application. 2. Enter admin username and password 3. Click "Log in". 4. Click "Hall of Fame". 	<p>Username = TrashPanda</p> <p>Password = rac00n</p>	<p>The top 3 most liked dogs are displayed on a podium.</p> <p>The runners-up are displayed in a scroll pane on the right side.</p>	As expected.	Pass



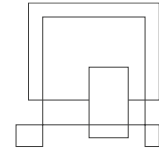
5.4 Requirement Traceability Matrix

Table 5.10. Requirement Traceability Matrix: Functional requirements

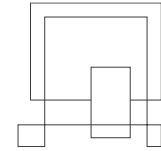
Requirement #	Requirement Description	Test case ID + Status	Final Grade
1	As a user, I want to be able to create a new profile so that I can use all the systems functionalities and be part of the community.	TC11 - Pass TC12 - Pass TC13 - Pass TC14 - Pass	Pass
2	As a user, I want to be able to add pictures and information about my pet(s) when modifying a profile so that I can have a special part of the profile dedicated to my dog(s).	TC51 - Pass TC52 - Pass TC53 - Pass TC54 - Pass TC55 - Pass TC56 - Pass TC57 - Pass	Pass
3	As a user I want to remove or add a pet's information to my profile freely, so that the other users can see an up to date list of my pets.	TC58 - Pass TC59 - Pass TC510 - Pass	Pass
4	As a user, I want to be able to add posts as a text description with or without an image on the platform, so that I can share my pet-related experiences and announcements with other people.	TC31 - Pass TC32 - Pass TC33 - Pass TC34 - Pass TC35 - Pass TC36 - Pass TC37 - Pass TC38 - Pass TC39 - Pass TC310 - Pass TC311 - Pass TC312 - Pass	Pass
5	As a user, I want to be able to like a post, so that I can show the user my appreciation	TC61 - Pass TC62 - Pass TC63 - Pass TC64 - Pass	Pass



	for their posted content.		
6	As a user, I want to be able to comment on other users' posts as a means of contacting and interacting with other users.	TC41 - Pass TC42 - Pass TC43 - Pass TC44 - Pass TC45 - Pass TC46 - Pass TC47 - Pass	Pass
7	As a user, I want to be able to see the comments on every selected post so that I can know the other users remarks and thoughts about it.	TC48 - Pass TC49 - Pass TC410 - Pass	Pass
8	As an admin, I want to be able to use all user-level functions (adding posts or comments, liking other users' posts and comments and adding dog information) so that I can be an active member of the community and moderate the platform better.	TC89 - Pass TC810 - Pass TC811 - Pass TC812 - Pass TC813 - Pass TC814 - Pass	Pass
9	As a user, I want to be able to see other users' posts in real time so that I can follow the latest news in the community.	TC31 - Pass TC32 - Pass TC33 - Pass TC34 - Pass TC35 - Pass	Pass
10	As a user, I want to be able to update selected fields in my profile, so that I can have full control over the personal information available on it.	TC71 - Pass TC72 - Pass TC73 - Pass TC74 - Pass TC75 - Pass TC76 - Pass TC77 - Pass	Pass



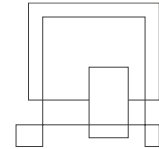
11	As an admin, I want to be able to delete user posts and comments when needed so that I can manage unwanted content and provide a well-maintained main feed.	TC82 - Pass TC83 - Pass TC84 - Pass TC85 - Pass TC86 - Pass TC87 - Pass	Pass
12	As a user, I want to be able to edit my comments or posts after they have been posted so that I can easily correct any mistakes without having to delete the post/comment.	TC46 - Pass TC313 - Pass TC314 - Pass	Pass
13	As a user, I want to be able to see other users' profiles, so that I can interact easier with their posts and dog posts.	TC312 - Pass	Pass
14	As an admin, I want to be able to block/unblock users when needed so that I can provide a safe and friendly environment on the platform for the users.	TC81 - Pass TC88 - Pass	Pass
15	As a guest I want to be able to see the main feed of the application whenever, without needing to be signed into an account, so that I have easy access to the platform's main content and source of entertainment.	TC25 - Pass	Pass
16	As a user, I want to be able to like other users' comments so that I can show my interest and support for the	TC67 - Pass TC68 - Pass	Pass



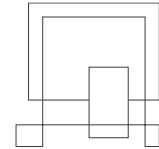
	comment.		
17	As a user, I want to be able to like other users' dogs so that I can help my favourite dogs get a place on the Hall of Fame.	TC65 - Pass TC66 - Pass	Pass
18	As a user, I want to see the top 10 most liked dogs on the Hall of Fame with their owner information, so that I can view their profile and see more posts from them.	TC91 - Pass	Pass

Table 5.11. Requirement Traceability Matrix : Non-functional requirements

Requirement #	Requirement Description	Notes	Final Grade
1	The system must support Windows 7, 8 and 10 OS.	-	Pass
2	The system will not be compatible with any mobile or tablet interfaces.	-	Pass
3	The system must refresh the page after a change has been committed or new page opened in 0.5 seconds 90% of the time and less than 2 seconds the rest of the time.	During testing the page refreshed in less than 0.4 seconds in every case.	Pass
4	Adding high quality images or other complex information to the system must provide a maximum of 5 seconds response time for the user.	-	Pass



5	All changes processing during a system crash on either server or client side must not be committed to the database and not saved in the system to ensure no faulty data entering the system.	No crash has ever been recorded during testing	Pass
6	No information already existing in the system should be affected during a system crash.	-	Pass
7	No normal user or guest should have access to any admin rights or the option to give themselves admin rights.	Admin rights are given manually in the database.	Pass
8	Passwords must be stored as hashed passwords in the database.	All passwords are saved in the database as SHA-256 hashes.	Pass
9	Users trying to add information or manage their accounts should result in an error rate of less than 5 percent.	No errors have been recorded in testing while trying to add or manage account information.	Pass
10	Time stamps and other date formats must be as follows: day/month/year.	All time stamps have the same format of dd/mm/yyyy.	Pass

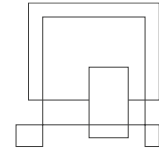


6 Results and Discussion

Natali Munk-Jakobsen

The Puppr application has been designed and implemented based on 18 functional requirements and these requirements structured 13 use cases. All these use cases successfully implemented as stated in the list below:

- The user can create a profile by clicking “Sign up” button and adding user info including user photo. “Manage profile” button opens a new view and user info can be managed and saved successfully. “Change password” button and function work as it is planned.
- “Add Dog” button opens a new view to add dog info and dog info is shown in a special section in the user profile. The user can easily edit or delete info about his dogs in this section.
- Adding post function works successfully and the user can add posts as a text with or without an image. The user also can edit or delete his posts.
- The user can see other users’ posts on the main feed and comment on these posts by clicking “Comment” button. The user also can edit or delete his comment.
- The user can see other users’ comments for a selected post by clicking “Show” button and opening a new view.
- “Like” button works as intended in all views and the user can like the selected post, comment or dog by clicking the bone image. In case of clicking this button, the number of likes for the item increases and bone image changes color.



- Admin can edit and delete all users' posts and comments.
- "Block" button works for admin users as planned and admins can block or unblock users.

The application has been designed to be used by the users, admins, and guests. All views are correctly shaped according to user type. Additional admin functions are only shown for admins and these functions cannot be used by other users. The guests can only reach a limited version of the application. An alert is shown when a guest attempts to use one of the restricted functions.

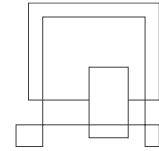
The non-functional requirements have been fulfilled to a large extent. Nevertheless, there is a need for improvement of the response time, since it is slower than it is aimed.

The Layered architecture corresponds to the analysis stage of the system and provides an efficient and useful structure to fulfill all requirements. Interconnections between layers work properly, therefore layers do not depend on each other but depend on abstraction.

The MVVM architecture could be improved by forming a better separation between Model, View and ViewModel and avoiding code repetition. However, it performs without any problem and covers all the expectations, providing a sufficient user interface and reliably communication between Model, View and ViewModel.

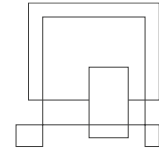
JavaFX elements in the views are responsive and work properly and they provide an easy and practical user experience as well as presenting a pleasant outlook. The input fields have been formed to check user inputs and warn the user in case of an invalid input or no input.

The connection between client and server can be established using the Remote Method Invocation mechanism and the client can invoke remote objects without any problem.



The database connection can be established successfully and queries for inserting, updating, and deleting data in Database package classes work as planned. The database can store the data and transfer the data further to the client when it is requested. After all, the database can perform as it is needed.

In conclusion, all the functionalities described in the requirements have been successfully implemented and a sufficient architectural structure has been established in the Puppr application.



7 Conclusions

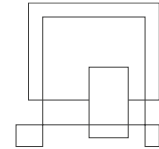
Bogdan-Alexandru Mezei

In conclusion, this project aimed at creating a social platform specifically designed for dog lovers in order to provide them with a safe and friendly place in which they could socialize, organize events and post updates about their pets.

Most notably, the “Puppr” app allows users to be part of the community either as a logged-in user or a guest, post publicly text and/or pictures and have a personal page dedicated to them and their dog companions.

Even though, through numerous iterations of designing and testing the application has been deemed to be very reliable, there is some room for improvement, mainly through the optimization of the code and UI that would result in fixing the slower response time that can sometimes be recorded.

Our final design was ultimately successful, managing to fulfill all the functional and non-functional requirements demonstrated by testing without sacrificing security while storing sensitive information in the database.



8 Project future

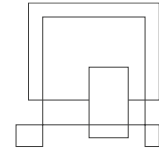
Lukas Suslavicius

The current version of the program fulfills all the functional requirements. From a technical standpoint the project works with some insignificant flaws. Since all the goals are met nothing more needs to be added right now. Having this strong base allows to have a strong idea of how to expand the project for the future.

First major improvement would be adapting the project for phones. Since “Puppr” is similar to “Tinder” and “Facebook” it is very important to be mobile when using it. For customers/users it would be much easier to arrange dates and meetings while having a way to contact the other person on the go. Adapting to a phones interface would bring new challenges. But because there is an established visual theme the hard part would only be implementing the code for phones and making it run alongside with RMI server. Visual implementation for the most part would be quick and easy. In addition being on the mobile market place would introduce “Puppr” to more users that perhaps don’t have computers.

Second major improvement could be expanding the users and admins capabilities. For example for the user side events could be implemented for larger gatherings, google map functionality for dog walking planning. On the other hand admins would get more moderation functionality in tandem with users. The most important addition though would be personal messages this would allow meeting to be personal and not in public. This would give some privacy and also would allow admins to discuss moderation between themselves away from users.

Finally the biggest addition for “Puppr” could be an animal lover application ecosystem that works with each other. While dogs are the obvious choice for this kind of application there are many types of animals. Not all animals can or should be taken outside, but having this ecosystem would allow for animal lovers in general to meet up

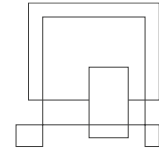


and discuss animals. This ecosystem would probably consist of about 6 branching applications for:

- Dogs
- Cats
- Aquatic animals
- Farm animals
- Birds
- Exotic

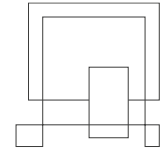
This would make it possible for animal lovers that are looking for specific animal owners to have a separate space for interactions. But this ecosystem would allow to interact with all the other application and this would allow for example cat lovers to see only cats on their application but in order to see other animal they would need to go out of their way to find them by specifically searching for other animals.

In conclusion expansion possibilities for the project are as numerous as there are species of animals. First major changes for the future should be technical and after the project is competitive with other social platform it would be time for spinoffs and expansions.



9 Sources of information

- Banger, D., 2014. A Basic Non-Functional Requirements Checklist « Thoughts from the Systems front line.... Available at: <https://dalbanger.wordpress.com/2014/01/08/a-basic-non-functional-requirements-checklist/> [Accessed January 31, 2017].
- Business Analyst Learnings, 2013. MoSCoW : Requirements Prioritization Technique — Business Analyst Learnings. , pp.1–5. Available at: <https://businessanalystlearnings.com/ba-techniques/2013/3/5/moscow-technique-requirements-prioritization> [Accessed January 31, 2017].
- Dawson, C.W., 2009. *Projects in Computing and Information Systems*, Available at: http://www.sentimentaltoday.net/National_Academy_Press/0321263553.Addison.Wesley.Publishing.Company.Projects.in.Computing.and.Information.Systems.A.Students.Guide.Jun.2005.pdf.
- Gamma, E. et al., 2002. *Design Patterns – Elements of Reusable Object-Oriented Software*, Available at: http://books.google.com/books?id=JPOaP7cyk6wC&pg=PA78&dq=intitle:Design+Patterns+Elements+of+Reusable+Object+Oriented+Software&hl=&cd=3&source=gbp_api%5Cnpapers2://publication/uuid/944613AA-7124-44A4-B86F-C7B2123344F3.
- IEEE Computer Society, 2008. *IEEE Std 829-2008, IEEE Standard for Software and System Test Documentation*,
- Larman, C., 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*,
- Mendeley.com, 2016. Homepage | Mendeley. Available at: <https://www.mendeley.com/> [Accessed February 2, 2017].
- YourCoach, S.M.A.R.T. goal setting | SMART | Coaching tools | YourCoach Gent. Available at: <http://www.yourcoach.be/en/coaching-tools/smart-goal-setting.php> [Accessed August 19, 2017].

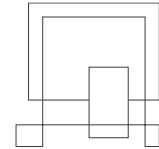


10 Appendices

Appendix A Project Description

Appendix B Use Case Description

Appendix C User Guide



Appendix A Project Description

Project description

Exam Planner

"Puppr"

Group 4

February 2020

Bogdan-Alexandru Mezei – 293137

Lukas Suslavicius – 293717

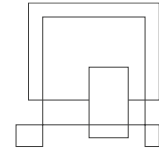
Daria Maria Popa - 293087

Natali Munk-Jakobsen - 293132

Supervisors:

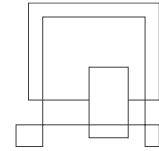
Henrik Kronborg Pedersen

Joseph Chukwudi Okika



Contents:

1. Background description.....	page 3
2. Problem Statement.....	page 4
3. Definition of purpose.....	page 5
4. Delimitation.....	page 6
5. Methodology.....	page 7
6. Time schedule.....	page 8
7. Risk assessment.....	page 9
8. Sources of information.....	page 11

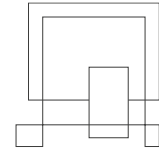


1. Background description

Pets have been one important part of the human life since prehistoric times (Pet | animal, 1998) and, as time progresses, their role in people's lives grows exponentially. With pets being considered members of the modern family, the latest figures show that 80 million European households alone have a pet (The Important Role Pets Have In Society - FEDIAF, 2020). The history of pets is intertwined with the process of animal domestication, and it is likely that the dog, as the first domesticated species, was also the first pet (Pet | animal, 1998). Dogs popularity only grew with time, mostly due to their friendly and helping nature, but also their health and physical benefits (cdc.gov, 2019), a study conducted in 2018 noting that "approximately 900 million people are dog owners" (Sundra Chelsea, 2018).

Due to the amount of pet owners nowadays it is only natural that the pet markets all over the world are growing and evolving at dramatic rates. From pet food, to toys to pet insurance, the American Pet Products Association (APPA) reported that Americans alone spent \$69.5 billion on the pet industry in 2017 (Here's a Look at the Pet Markets Trends Around the World, 2018). Online markets and platforms are the newest and most convenient method of shopping. Dog lovers have access to a wide array of online markets and social platforms such as Instagram or Facebook but few of them are made specifically for them. Because of this, the demand for an online place where the dog-loving community can gather and interact, organize events and socialize is on the rise.

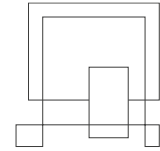
Famous platforms such as Facebook and Instagram handle this demand quite poorly and offer no benefits or special features for the dog loving users they have. More specific platforms and applications like Dogster, Chewy and FitBark aid with ordering food, monitoring dog's health and adoption, but still none offer a place for only dog lovers to share their pictures, post, comment and interact with other owners.



2. Problem statement

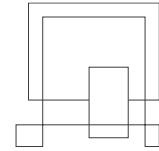
The main problem is dog owners not having their own personal place to socialize.

- a. What features would dog owners like to have on the social platform?
- b. How could owners connect to each other more easily?
- c. What are social platforms today lacking in?
- d. How do people organize events for their pets?



3. Definition of purpose

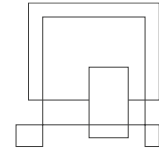
We want to create a social platform for dog owners centered around dogs because the market is lacking and demand is rising among the target audience.



4. Delimitation

The product will not:

- have private messaging between users because the application focuses on the community on a larger scale.
- run on mobile platforms because it is designed for Windows machines.
- support video formats or file sharing because the focus of the program is text and picture formats.
- feature an online marketplace because it is intended only for social use



5. Methodology

Use of SCRUM

The whole team will start each sprint by having a SCRUM meeting on Wednesday and setting the goals. At the end of each sprint, there will be a period of testing and documenting the progress made and what still requires work.

Another important SCRUM meeting will be held at the end of the sprint in order to decide the goals for the next sprint.

Roles

Bogdan Mezei - Product owner

Lukas Suslavicius - SCRUM master

Daria Maria Popa - SCRUM team

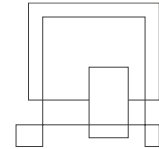
Natali Munk-Jakobsen - SCRUM team

Sprint length

Each sprint will have a total duration of 7 days, starting on Wednesday and ending on the following Wednesday.

Number of sprints

13 sprints starting March 4th ending on June 4th.



6. Time Schedule

The Semester Project has 10 ECTS, which results in approximately 280 hours of work per student.

The iterations will begin starting with the 4th of March and end before the final deadline, the 4th of June. Each will have a span of exactly one week.

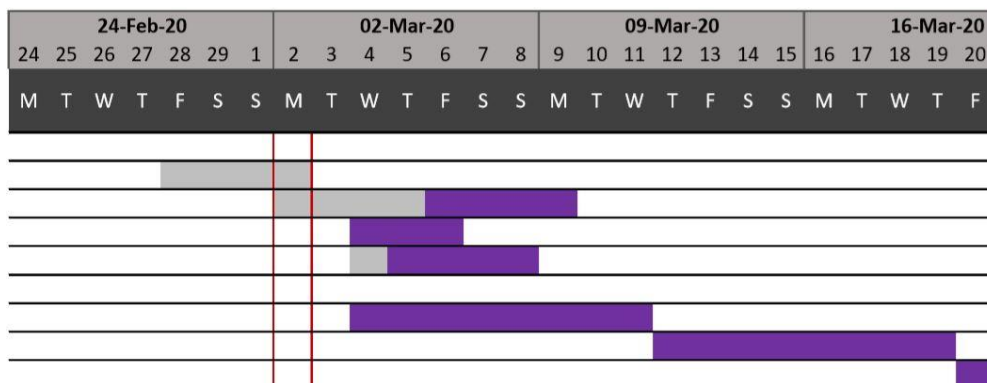
The time schedule, including the major goals set by the team, is split based on the four phases: Inception, Elaboration, Construction and Transition.

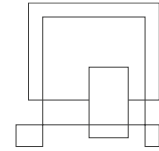
PUPPR

VIA University College

Project Start: Sun, 1-Mar-2020
Display Week: 1

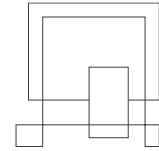
TASK	ASSIGNED TO	PROGRESS	START	END
Inception				
Create mockup project description		100%	28-Feb-20	02-Mar-20
Finalize project description		50%	02-Mar-20	09-Mar-20
Evaluate scope		10%	04-Mar-20	06-Mar-20
Business case modelling		20%	04-Mar-20	08-Mar-20
Elaboration				
Iteration 1		0%	04-Mar-20	11-Mar-20
Iteration 2		0%	12-Mar-20	19-Mar-20
Iteration 3		0%	20-Mar-20	27-Mar-20



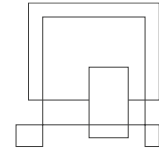


7. Risk assessment

Risks	Description	Likelihood Scale: 1-5 5 = high risk	Severity Scale: 1-5 5 = high risk	Product of likelihood and severity	Risk mitigation e.g Preventive & Responsive actions	Identifiers	Responsible
Risk 1	Lack of time before hand- in	2	5	10	Preventive: Working with SCRUM by following sprint goals strictly. Responsive: We try to take the unnecessary features out and prioritize the main structure	Falling behind on sprint goals. Falling behind on deadlines	Whole team
Risk 2	Client-server communicati on errors	3	5	15	Preventive Work: Testing and maintaining the server Responsive Work: Isolating and solving the problem	Server not responding properly	Whole team



Risk 3	Failing to integrate the database with the system	2	5	10	<p>Preventive Work: Designing the database and connections carefully</p> <p>Responsive: Checking database structure and related codes</p>	Irrelevant/incorrect information	Whole team
Risk 4	Choosing goals poorly	2	3	6	<p>Preventive Work: Checking sprints carefully together as a team</p> <p>Responsive Work: Changing the goals of the sprints</p>	Not being able to finish a sprint	Product Owner



8. Sources of information

Atitwa, Sundra Chelsea 2018, How Many Dogs Are There In The World?, WorldAtlas, viewed 23 February 2020, <<https://www.worldatlas.com/articles/how-many-dogs-are-there-in-the-world.html>>.

2019, cdc.gov, viewed 23 February 2020

<<https://www.cdc.gov/healthypets/health-benefits/index.html>>

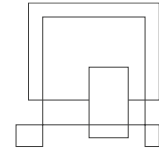
Fediaf.org. 2020. *The Important Role Pets Have In Society - FEDIAF*. [online]

Available at: <<http://www.fediaf.org/pets-in-society/the-important-role-pets-have-in-society.html>> [Accessed 27 February 2020].

Encyclopedia Britannica. 1998. Pet | Animal. [online] Available at:

<<https://www.britannica.com/animal/pet>> [Accessed 27 February 2020].

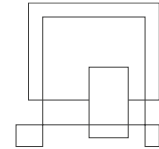
The Balance Careers. 2018. Here's A Look At The Pet Markets Trends Around The World. [online] Available at: <<https://www.thebalancecareers.com/the-world-pet-market-booms-2660629>> [Accessed 27 February 2020].



Appendix B Use Case Description

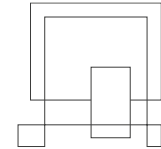
Daria Maria Popa

USE CASE UC1: CREATE PROFILE	
SCOPE	Puppr application
LEVEL	User goal
PRIMARY ACTOR	User
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none">- User: Wants an intuitive creation process, with working features of adding the wanted personal information.- Admin: Wants the profiles to be created correctly and according to the guidelines
PRECONDITIONS	No other profile exists with the same email/username
SUCCESS GUARANTEE	A profile is created with username, password and it can further be managed and altered. All information is correctly added
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none">1. User selects to sign up under a new account.2. Menu opens and the user fills the information needed (email, password, full name, etc.)3. Information is accepted by the user and the system and the new user is created4. New user info is stored in the database5. User can now log in, access his profile, and interact with the feed whenever they want
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ul style="list-style-type: none">*a. User with the same username already exists<ol style="list-style-type: none">1. Error message is shown2. User can check and re-enter a new email or go directly to log in2. No information provided in one or more fields<ol style="list-style-type: none">1. The system will show a warning notifying the user of the need to fill in the obligatory fields



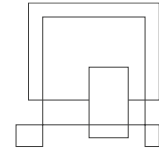
	2. The user can proceed to the next step after providing all the information needed to the system or exit and choose another option from log in page
SPECIAL REQUIREMENTS	- Easy and user-friendly UI for account creation
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Often to very often
MISCELLANEOUS	

USE CASE UC2 : MANAGE PROFILE	
SCOPE	Puppr application
LEVEL	User and admin goal
PRIMARY ACTOR	User, Admin
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none"> - User: Wants to be able to access and modify the private and public information in an easy and user-friendly way - Admin: Wants the users to have a certain degree of freedom in choosing the personal information they want to make public, while being able to manage their own account information
PRECONDITIONS	A working account is already set up for the users
SUCCESS GUARANTEE	The selected profile data is modified and updated accordingly and immediately, so other users can see the new information in real time.
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. User or admin log in with using their credentials 2. User or admin enters the "Manage profile" menu from "Settings" 3. User/Admin select the fields that they want to modify 4. The new data is entered and registered in the system 5. After all changes are done, the user can select to save or cancel them. 6. The "Manage profile" window is closed automatically, and the user/admin can proceed with any other activities, for example adding a post.
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> 1a. Login information is incorrect <ol style="list-style-type: none"> a. The user/admin will be prompted to check the information and try again b. If the login succeeds, they can proceed to the next step, otherwise they can only see the main feed as a guest 2a. The new data entered by the user/admin is not valid



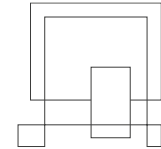
	<ol style="list-style-type: none"> 1. An error will be displayed, notifying the user about any limitations that need to be taken into account (eg: already existing username, any empty field, etc.) 2. After new data that conforms to the regulations of the system is entered, the error will disappear, and system changes will be saved <p>3a. User/Admin wants to add info about a new pet</p> <ol style="list-style-type: none"> 1. The system will show a window with all the information the user can provide about the pet, including a picture 2. After adding all the needed information, they can save the changes 3. A new special part in their profile dedicated to the newly added pet will appear
SPECIAL REQUIREMENTS	
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Often
MISCELLANEOUS	

USE CASE UC3: ADD POST	
SCOPE	Puppr application
LEVEL	User and admin goal
PRIMARY ACTOR	User, Admin
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none"> • User: Wants a fast and intuitive way of adding new post and sharing media on the platform for other users to see. • Admin: Wishes for the system to function as a platform where both user types can easily share media
PRECONDITIONS	
SUCCESS GUARANTEE	Post will be added, saved by the system, and made available for all the users, admin, and guests to see, and for admins and users to also interact with.
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. User/Admin selects the "Add post" option 2. User/Admin then can select from adding text, pictures, or a combination of both 3. After selecting what to post, they click the "Post" button



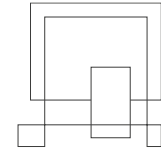
	<ol style="list-style-type: none"> The post is added to the main wall with all the other posts, dated and made available for everyone to see and like or comment depending on their user type.
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> User not logged in <ol style="list-style-type: none"> System will signal to the user the need to log on first and redirect them to that page After successfully logging in, the user can go back to adding a post User/Admin cancels the add post process <ol style="list-style-type: none"> System will show a message asking for confirmation from the user <ol style="list-style-type: none"> If user selects to proceed with not posting the content the post will be saved to drafts for easy later access and the user will be returned to the main page If the user resumes the add process, they will be redirected to their post unfinished post.
SPECIAL REQUIREMENTS	<ul style="list-style-type: none"> Interactive and user-friendly UI for posting - Support of png and jpg formats
TECHNOLOGY AND DATA VARIATIONS LIST	2a. For the future other formats might be wanted by the users, for example mp4
FREQUENCY OF OCCURRENCE	Could be almost continuous
MISCELLANEOUS	

USE CASE UC4 : ADD COMMENT	
SCOPE	Puppr application
LEVEL	User and admin goal
PRIMARY ACTOR	User, Admin
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none"> User: Wants an easy and fast way of interacting with other users through commenting on posts Admin: Wants to further ensure the social functionalities of the platform by allowing users to comment on each other's post and being able to leave their own comments
PRECONDITIONS	
SUCCESS GUARANTEE	Comment is added and saved to the list of comments for the certain post so everyone can see it and interact with it



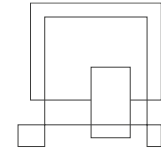
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. User/Admin clicks the comment area under a certain post 2. User/Admin types in the comment and clicks the "Add comment" button 3. After adding the comment, it is added to the list on comments for the certain post 4. User/Admin can continue adding more comments to that post, other posts or continue with other activities on the app
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> 1a. User/Admin is not logged in <ol style="list-style-type: none"> 1. A pop-up window will require them to log in or sign up before being able to add a comment 2. After proceeding with the log in process, the use will be returned to the main flow window 3. They can return to their add comment activity or other activities 2a. "Add comment" is clicked without a comment being entered <ol style="list-style-type: none"> 1. Comment will not be added, and system will notify the user for the need to add text by changing the comment section color
SPECIAL REQUIREMENTS	<ul style="list-style-type: none"> - Comment section updated at most 2 seconds after a new comment has been added - Support for English as the main system language
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Very often or nearly continuous.
MISCELLANEOUS	

USE CASE UC5: ADD DOG	
SCOPE	Puppr application
LEVEL	User and admin goal
PRIMARY ACTOR	User, Admin
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none"> • User: Wants to be able to share their dogs with the other users, this way contributing directly to the dog owner community • Admin: Wants the users to be able to showcase their dogs on their individual profile, while having the right to add their own dog information too.
PRECONDITIONS	
SUCCESS GUARANTEE	Dog is added to the owner's profile and saved in the system.



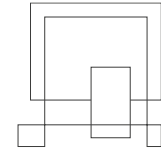
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. User/Admin picks the "Add Dog" option when managing their account. 2. User/Admin fills the name of the dog and can add a short description and a photo. 3. The "save" option is picked 4. New Dog information is saved and linked to the profile of the user or admin for everyone to see 5. The user/admin can continue with other activities or add another dog
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> 1a. User/Admin is not logged in <ol style="list-style-type: none"> 1. A pop-up window will require them to log in or sign up before being able to manage an account and add a dog 2. After proceeding with the adding process, the use will be returned to the managing window where they can continue their activities 2a. "Save" is clicked without a name being entered for the dog <ol style="list-style-type: none"> 1. The system will notify the user/admin of the empty field and will save the new Dog information once the needed information is added 2b. User/Admin selects to exit before saving <ol style="list-style-type: none"> 1. The Dog information will not be saved and window exited
SPECIAL REQUIREMENTS	<ul style="list-style-type: none"> - The system will refresh after a change has been committed in 0.5 seconds 90% of the time.
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Sometimes to often
MISCELLANEOUS	

USE CASE UC6: LIKE POST	
SCOPE	Puppr application
LEVEL	User and admin goal
PRIMARY ACTOR	User, Admin
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none"> - User: Wants to be able to like other users' or their own posts - Admin: Wants the users and admins to be able to like each other's post



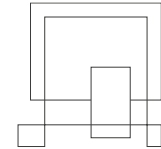
PRECONDITIONS	
SUCCESS GUARANTEE	Number of likes will be increased, and new total number of likes will be shown with the related post.
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. The user or admin selects a post from the wall 2. User/Admin clicks the "Like" button under the post 3. After clicking the button, number of likes will go up and updated number of likes will be shown with the post 4. User can continue interacting with the same post or continue with other activities on the app
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> 1a. User/Admin is not logged in <ol style="list-style-type: none"> 1. A pop-up window will require them to log in or sign up before being able to like a post or continue as a guest <ol style="list-style-type: none"> a. If they proceed with the login process, the user or admin will be returned to the main flow window where they can return to their activities and like the post b. If they choose to continue as a guest, they will be only able to view the posts and profiles 2-3. Post is already liked <ol style="list-style-type: none"> 1. If the post has been already liked by the same user or admin, clicking the button will result in no change in the system
SPECIAL REQUIREMENTS	
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Could be nearly continuous.
MISCELLANEOUS	

USE CASE UC7: LIKE COMMENT	
SCOPE	Puppr application
LEVEL	User and admin goal
PRIMARY ACTOR	User, Admin
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none"> - User: Wants to be able to like other or their own comments - Admin: Wants all users and admins to be able to like each other's comments
PRECONDITIONS	



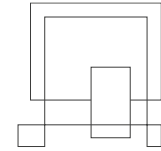
SUCCESS GUARANTEE	The number of likes will be increased and it will be shown together with the related comment
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. User or admin selects a comment from the list related to a post 2. User/Admin clicks the "Like" button for the selected comment 3. After clicking the button, number of likes will go up and the updated number of likes will be shown with the comment 4. They can continue interacting with the same post or continue with other activities on the app
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> 1a. User/Admin is not logged in <ol style="list-style-type: none"> 2. A pop-up window will require them to log in or sign up before being able to like a comment, or continue as a guest <ol style="list-style-type: none"> a. If they proceed with the login/sign up process, the user or admin will be returned to the main flow window where they can return to their activities and like the comment b. If they choose to continue as a guest, they will be only able to view the posts and profiles 2-3. Comment is already liked <ol style="list-style-type: none"> 1. If the comment has been already liked by the same user or admin, clicking the button will result in no change in the system
SPECIAL REQUIREMENTS	
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Could be very often
MISCELLANEOUS	

USE CASE UC8: LIKE DOG	
SCOPE	Puppr application
LEVEL	User and admin goal
PRIMARY ACTOR	User, Admin
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none"> - User: Wants to be able to like other or their own dogs - Admin: Wants all users and admins to be able to like each other's comments
PRECONDITIONS	
SUCCESS GUARANTEE	The number of likes will be increased and it will be shown together with the related dog object

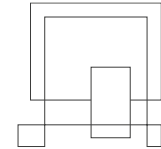


MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. User or admin chooses to view a user profile 2. User/Admin clicks the "Like" button for a dog 3. After clicking the button, number of likes will go up and the updated number of likes will be shown with the dog post 4. They can continue interacting with the same user profile or continue with other activities on the app
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> 1a. User/Admin is not logged in <ol style="list-style-type: none"> 3. A pop-up window will require them to log in or sign up before being able to like a dog post or continue as a guest <ol style="list-style-type: none"> a. If they proceed with the login process, the user or admin will be returned to the main flow window where they can return to their activities and like the dog post b. If they choose to continue as a guest, they will be only able to view the posts and profiles 2-3. Dog is already liked <ol style="list-style-type: none"> 1. If the post has been already liked by the same user or admin, clicking the button will result in no change in the system
SPECIAL REQUIREMENTS	
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Could be very often
MISCELLANEOUS	

USE CASE UC9: MANAGE POSTED CONTENT	
SCOPE	Puppr application
LEVEL	Admin level
PRIMARY ACTOR	Admin
STAKEHOLDER AND INTERESTS	<p>User: - Wants the platform and its posts to be well maintained and managed in order to have a pleasant experience while using it.</p> <p>Admin: - Wants to be able to moderate or remove unwanted content to ensure user safety and experience</p>
PRECONDITIONS	

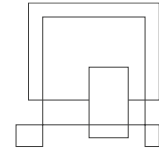


SUCCESS GUARANTEE	The selected posts, comments or dogs will be edited or removed completely, changes saved, and system updated for all the users and admins.
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. Admin logs in with the special credentials 2. System verifies and confirm the admin status 3. Main feed is shown in real time and delete functions are available for every post/comment. 4. An admin can select if they want to edit or delete a post or comment from the main feed or a dog's information from a user profile 5. If the admin decides to proceed, the information will be edited or deleted accordingly from the main feed, profile, and the system's memory 6. The admin can continue deleting more posts/comments or chose to log out and end their session.
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> 1-2. Admin status not recognized <ol style="list-style-type: none"> 1. The Admin will be prompted by the system to check and re-enter the credentials <ol style="list-style-type: none"> a. If the correct credentials are entered the system will be started in admin mode b. If the admin fails to enter to correct login info, the system will not login or give any special admin roles to the user for security reasons 5-6. Admin decides not to proceed with deleting or editing the information <ol style="list-style-type: none"> 1. The selected information will not be affected or modified in any way 2. The admin will be returned to the main feed by the system 3. The admin can the continue his post managing activities or end his session
SPECIAL REQUIREMENTS	<ul style="list-style-type: none"> - System updates the admin changes within 1 second 95% of the time.
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Relatively often.
MISCELLANEOUS	<p>Open issues:</p> <ul style="list-style-type: none"> - Will a user be directly notified when their post has been deleted? - Should users be able to signal for comments or posts that ought to be moderated by the admin?



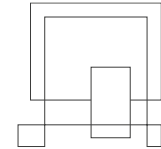
	<ul style="list-style-type: none"> - Can a user disagree with the admin's choice to moderate their comment or post?
--	--

USE CASE UC10: MANAGE USERS	
SCOPE	Puppr application
LEVEL	Admin level
PRIMARY ACTOR	Admin
STAKEHOLDER AND INTERESTS	<p>User: - Wants the platform to be a safe and friendly place for every user or viewer</p> <p>Admin: - Wants to be able to moderate and limit the possibilities of users that have not conformed to the platform's rules.</p>
PRECONDITIONS	
SUCCESS GUARANTEE	The selected user will be either blocked for an undetermined period of time, and so made unable to login with that account, or unblocked and made able to log in, post and comment freely again.
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. Admin logs on and gets identified by the system 2. Admin selects the user's profile who has been not conforming to the platform regulations or a user previously blocked 3. The admin can select the "Block user" or, respectively, the "Unblock user" option 4. After the Admin proceeds with the option the user's capabilities of logging in the app with those credentials will be removed or returned accordingly
EXTENSIONS (OR ALTERNATIVE FLOWS)	<p>1a. Admin status not recognized</p> <ol style="list-style-type: none"> 1. The Admin will be prompted by the system to check and re-enter the credentials <ol style="list-style-type: none"> a. If the correct credentials are entered the system will be started in admin mode b. If the admin fails to enter to correct login info, the system will not login or give any special admin roles to the user for security reasons
SPECIAL REQUIREMENTS	<ul style="list-style-type: none"> - System updates the admin changes and notifies the user within 2 seconds 95% of the time.



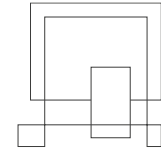
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Rarely
MISCELLANEOUS	<p>Open issues:</p> <ul style="list-style-type: none"> - Will an admin be able to block another admin?

USE CASE UC11: MANAGE DOG	
SCOPE	Puppr application
LEVEL	User goal
PRIMARY ACTOR	User
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none"> - User: Wants to have the ability to remove or edit the information related to their dogs - Admin: Wants the users to have the freedom to modify the information about the dogs they own
PRECONDITIONS	User has already a set up account
SUCCESS GUARANTEE	The selected dog data is modified and updated accordingly and immediately, so other users can see the new information in real time.
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. User logs in using their credentials (username and password) 2. User opens their profile page and decides to delete or edit the information related to one of their dogs 3. User clicks the related "delete" or "edit" buttons 4. The data will be deleted or edited accordingly 5. Changes will be saved, and profile page updated across the app
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> 1a. Login information is incorrect <ol style="list-style-type: none"> a. The user/admin will be prompted to check the information and try again b. If the login succeeds, they can proceed to the next step, otherwise they can only see the main feed as a guest 2a. No dog information is linked to the user's profile <ol style="list-style-type: none"> 1. The system will not display any edit or delete buttons until the user completes an "add dog" operation in the app 3a. User selects the delete option



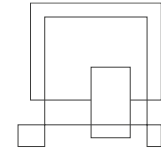
	<ol style="list-style-type: none"> The dog data will be deleted immediately, and profile page updated 3b. User selects the edit option <ol style="list-style-type: none"> A new window showing the current information related to the respective dog will pop up The user can select what to change and what to leave the same way <ol style="list-style-type: none"> User chooses to save the changes: <ul style="list-style-type: none"> The new old information will be replaced with the new one and changes committed User chooses to exit the edit menu: <ul style="list-style-type: none"> The old information will remain unedited
SPECIAL REQUIREMENTS	
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Rarely
MISCELLANEOUS	

USE CASE UC12: MANAGE POST	
SCOPE	Puppr application
LEVEL	User goal
PRIMARY ACTOR	User
STAKEHOLDER AND INTERESTS	<ul style="list-style-type: none"> User: Wants to have the ability to remove or edit their posts after posting them to the app Admin: Wants the users to have the possibility to fix any mistakes or delete any of their posts freely
PRECONDITIONS	User has already a set up account
SUCCESS GUARANTEE	The selected post data is modified and updated accordingly and immediately, so other users can see the new information in real time.
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> User logs in using their credentials (username and password) User opens their profile page and decides to delete or edit of one their posts User clicks the related "delete" or "edit" buttons for one of the posts The data is deleted or edited accordingly if the changes are saved

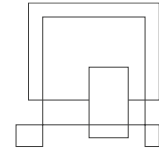


	10. All changes are committed and updated for all users
EXTENSIONS (OR ALTERNATIVE FLOWS)	<p>1a. Login information is incorrect</p> <ul style="list-style-type: none"> c. The user/admin will be prompted to check the information and try again d. If the login succeeds, they can proceed to the next step, otherwise they can only see the main feed as a guest <p>2a. User has no posts</p> <ul style="list-style-type: none"> 2. The system will not display any edit or delete buttons until the user completes an "add post" operation in the app <p>3a. User selects the delete option</p> <ul style="list-style-type: none"> 3. The post data will be deleted immediately, as well as the comments and likes linked to it 4. Both the profile page and the main feed will be updated for everyone <p>3b. User selects the edit option</p> <ul style="list-style-type: none"> 3. A new window showing the current information related to the respective post will pop up 4. The user can select what to change and what to leave the same way <ul style="list-style-type: none"> c. User chooses to save the changes: <ul style="list-style-type: none"> ▪ The new old information will be replaced with the new one and changes committed d. User chooses to exit the edit menu: <ul style="list-style-type: none"> ▪ The old information will remain unedited
SPECIAL REQUIREMENTS	System updates the admin changes and notifies the user within 2 seconds 95% of the time.
TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Often
MISCELLANEOUS	

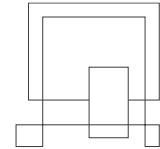
USE CASE UC13: MANAGE COMMENT	
SCOPE	Puppr application
LEVEL	User goal
PRIMARY ACTOR	User
STAKEHOLDER AND INTERESTS	- User: Wants to have the ability to remove or edit their comments after posting them to on the app



	<ul style="list-style-type: none"> - Admin: Wants the users to have the possibility to fix any mistakes or delete any of their comments freely
PRECONDITIONS	User has already a set up account
SUCCESS GUARANTEE	The selected comment data is modified and updated accordingly and immediately, so other users can see the new information in real time.
MAIN SUCCESS SCENARIO	<ol style="list-style-type: none"> 1. User logs in using their credentials (username and password) 2. User opens a post's page and decides to delete or edit of one their comments made on the said post 3. User clicks the related "delete" or "edit" buttons for one of the posts 4. The data is deleted or edited accordingly if the changes are saved 5. All changes are committed and updated for all users
EXTENSIONS (OR ALTERNATIVE FLOWS)	<ol style="list-style-type: none"> 1a. Login information is incorrect <ol style="list-style-type: none"> a. The user/admin will be prompted to check the information and try again b. If the login succeeds, they can proceed to the next step, otherwise they can only see the main feed as a guest 2a. User has no comments on the post <ol style="list-style-type: none"> 1. The system will not display any edit or delete buttons to any other comment until the user adds a comment to the specific post 3a. User selects the delete option <ol style="list-style-type: none"> 1. The comment will be deleted immediately 2. Both the profile page and the main feed will be updated for everyone 3b. User selects the edit option <ol style="list-style-type: none"> 1. A new window showing the current information related to the respective comment will pop up 2. The user can select what to change and what to leave the same way <ol style="list-style-type: none"> a. User chooses to save the changes: <ul style="list-style-type: none"> ▪ The new old information will be replaced with the new one and changes committed b. User chooses to exit the edit menu: <ul style="list-style-type: none"> ▪ The old information will remain unedited
SPECIAL REQUIREMENTS	System updates the admin changes and notifies the user within 2 seconds 95% of the time.



TECHNOLOGY AND DATA VARIATIONS LIST	
FREQUENCY OF OCCURRENCE	Rarely
MISCELLANEOUS	



Appendix C User Guide

Natali Munk-Jakobsen

Create New Profile

1. Open application and click SIGN UP button.
2. Enter First name, Last name, Handle, Email, Password and Personal description.
3. Select Date of birth and Gender.
4. Click UPLOAD IMAGE button and select an image.
5. Click SIGN UP button.

Edit Profile

1. Open application, enter username and password and click LOGIN button.
2. Click MANAGE PROFILE button.
3. Edit First name, Last name, Email, Personal description, Date of birth or Gender.
4. (Optionally) Click UPLOAD IMAGE button and select an image.
5. (Optionally) Click CHANGE PASSWORD button, enter current password and new password, repeat new password and click SAVE button.
6. Click SAVE button to save all changes.

Add Post

1. Open application, enter username and password and click LOGIN button.
2. Click the green post button which is located in the bottom right corner (see Figure 2).
3. Enter a text in the text area.
4. (Optionally) Click ADD PHOTO button and select an image.
5. Click POST button to save the post.

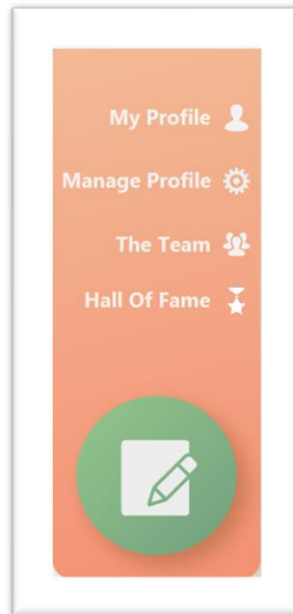
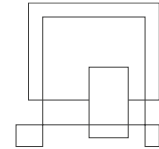


Figure 1. Post button

Add Comment

Adding comment in the main feed

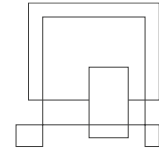
1. Open application, enter username and password and click LOGIN button.
2. Click comment button (see Figure 2) for selected post in the main feed.
3. Enter a text in the text area.
4. Click SEND button to save the comment.

Adding comment in post view

1. Open application, enter username and password and click LOGIN button.
2. Click SHOW button for selected post.
2. Click comment button for the post in the post view.
3. Enter a text in the text area.
4. Click SEND button to save the comment.

Add Dog

1. Open application, enter username and password and click LOGIN button.
2. Click MANAGE PROFILE button.
3. Click ADD DOG button.
4. Enter dog name and dog description.
5. (Optionally) Click SET IMAGE button and select a dog image.
6. Click ADD button to save the dog.



Like Post

1. Open application, enter username and password and click LOGIN button.
2. Click like button (see Figure 2) for selected post in the main feed.
3. (Alternatively) Click SHOW button for selected post and click like button for the post in the post view.

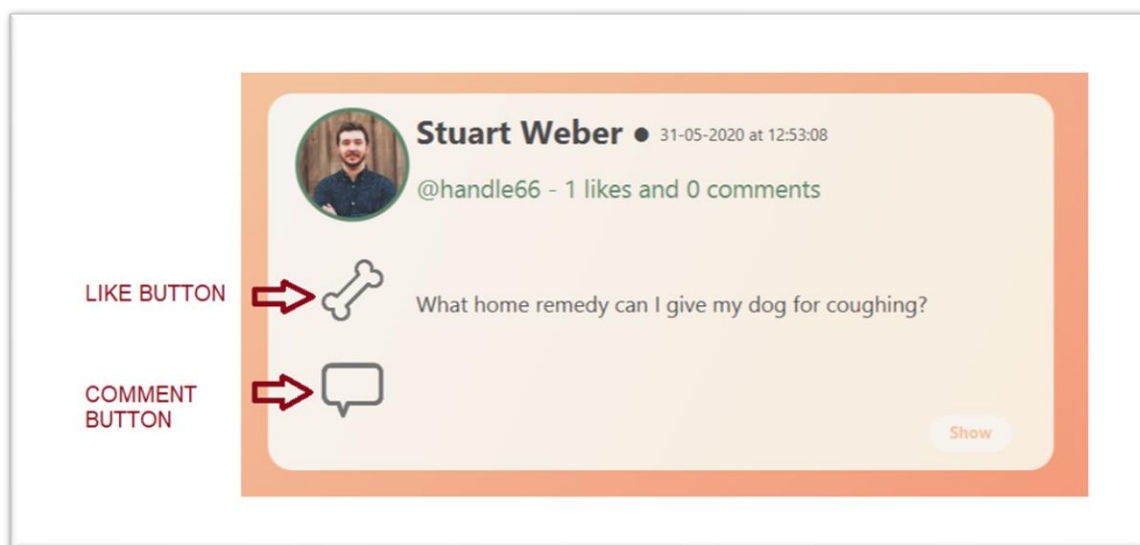


Figure 2. Like Button and Comment Button

Like Comment

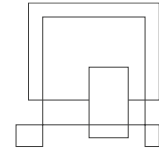
1. Open application, enter username and password and click LOGIN button.
2. Click SHOW button for selected post in the main feed.
2. Click like button (see Figure 2) for the selected comment.

Like Dog

1. Open application, enter username and password and click LOGIN button.
2. Open a user profile by clicking username or user photo.
3. Click like button (see Figure 2) for selected dog in the Puppies section.

Edit Dog

1. Open application, enter username and password and click LOGIN button.
2. Click MY PROFILE button to open the user profile.



3. Click EDIT button for selected dog in the Puppies section.
4. Edit dog name and dog description.
5. (Optionally) Click SET IMAGE button and select a new dog image.
6. Click EDIT button to save all changes.

Edit Post

1. Open application, enter username and password and click LOGIN button.
2. Click MY PROFILE button to open the user profile.
3. Click EDIT button for selected post in the Posts section.
4. Edit the text in the text area.
5. (Optionally) Click EDIT PHOTO button and select a new image.
6. Click POST button to save the edited post.

Edit Comment

1. Open application, enter username and password and click LOGIN button.
2. Click SHOW button for the post related to the comment.
3. Click EDIT button for selected comment.
4. Edit the text in the text area.
5. Click SEND button to save the edited comment.

Delete Dog

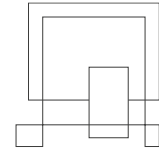
1. Open application, enter username and password and click LOGIN button.
2. Click MY PROFILE button to open the user profile.
3. Click DELETE button to delete selected dog in the Puppies section.

Delete Post

1. Open application, enter username and password and click LOGIN button.
2. Click MY PROFILE button to open the user profile.
3. Click DELETE button to delete selected post in the Posts section.

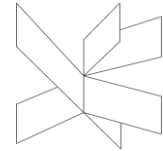
Delete Comment

1. Open application, enter username and password and click LOGIN button.
2. Click SHOW button for the post related to the comment.
3. Click DELETE button to delete selected comment.



Manage user (Only for admin)

1. Open application, enter username and password and click LOGIN button.
2. Open a user profile by clicking username or user photo.
3. Click BLOCK button to block the user or click UNBLOCK button to unblock the user.



Puppr

Dog owner socializing platform

Daria-Maria Popa
293087



Bogdan-Alexandru
Mezei 293137



Natali Munk-
Jakobsen 293132



Lukas Suslavicius
293717



Supervisors: Henrik Kronborg Pedersen, Joseph Chudwudi Okika



**VIA University
College**

Number of characters: 29.339

Software engineering

Semester 2

03-06-2020

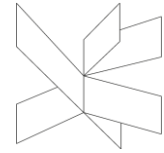
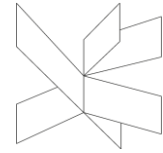


Table of content

1	Introduction	1
2	Group Description	3
3	Project Initiation	7
4	Project Description	9
5	Project Execution	11
6	Personal Reflections	14
7	Supervision	23
8	Conclusions	25

Appendices



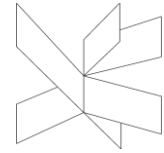
1 Introduction

Lukas Suslavicius

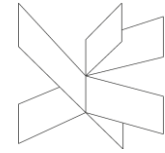
At the beginning of the semester we were tasked to create a project vision, outline our timeframe and decide on the SCRUM sprint length. These were all made in order to make sure that we will organize our work in such a way that would permit us to work evolutionarily. We decided on sprints that each last a week and would end right before the deadline, giving us enough time to plan and execute our project. The sprints were 13 in total. We decided on the week-long sprints so that we would have enough time to accomplish our tasks and even solve any errors that might occur while working. The structure was meetings on Wednesdays to discuss sprint result and decide on next week's goals.

The semester goal was to make a functioning client server system with database storage while efficiently and correctly implementing the SCRUM and UP (Unified Process) methodologies. The system was to be hosted locally and not over the network. Our plan was to make a socializing app that would conform to these rules. During the making of this system the sprint goals and functional requirements were met in due time. On average we spent 1 to 2 hours per weekly sprint meeting and about 4 hours overall work each day to realize our project. Daily SCRUM meetings were held at a convenient time and lasted approximately 10 minutes. These helped clear any confusion and steer the focus on the next important step. The project was done on time without any glaring oversights.

The supervisor's help was used along the whole span of the project on various different topics and areas of interest for our project. Supervisor meeting were mostly arranged online and consisted in 20 to 40 minutes of directly addressing questions or uncertainties and resolving them together with the supervisors. These were as well a very important step for the end result.



In conclusion the project progress and execution was slow yet steady, increasing in productivity with time, once the team figured out how to apply the methodologies better. Having the possibilities to work iteratively allowed us to fix problems that we encountered and soundly plan our development which resulted in very few changes needing to be done towards the end.



2 Group Description

Lukas Suslavicius

Team 4 or "Team cake" consists of three nationalities Romanian, Turkish and Lithuanian. Looking at the Hofstede insight chart (Image 2.1) we can see that the three countries are radically different.

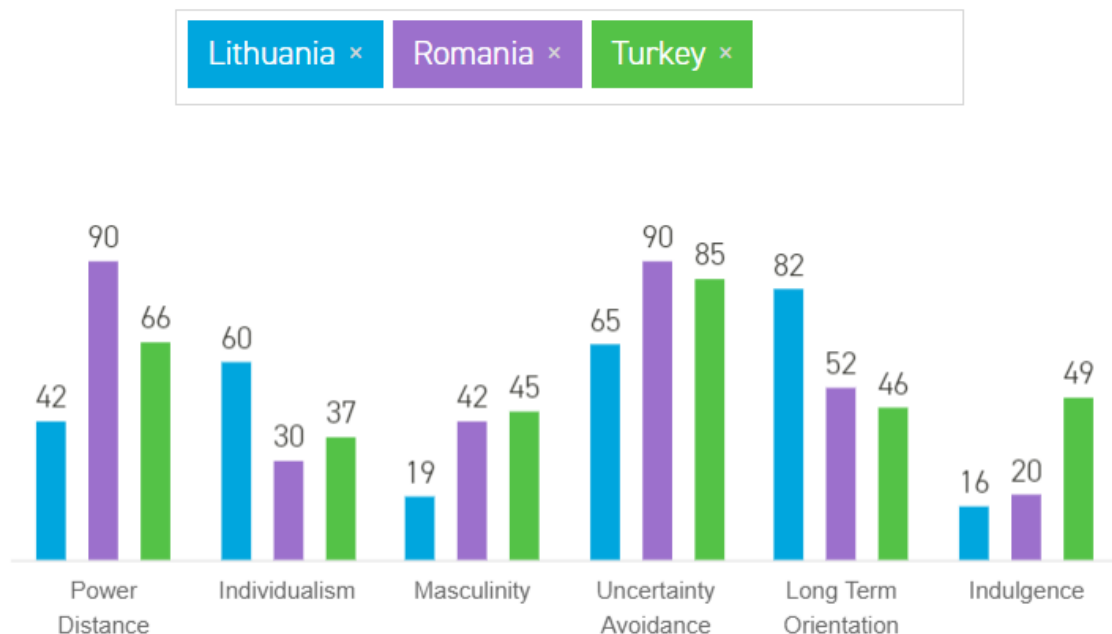
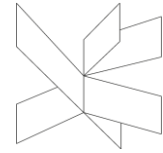


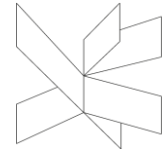
Image 2.1: Hofstede's diagram

- Power distance:** This dimension deals with the fact that all individuals in societies are not equal – it expresses the attitude of the culture towards these inequalities amongst us. Power Distance is defined as the extent to which the less powerful members of institutions and organisations within a country expect and accept that power is distributed unequally. In Power distance we have drastically different understandings of equality. While Turkey comparatively is in the middle Lithuanians and Romanians understanding of power distance is drastically



different. From the chart we can see that Romanians prefer I strict hierarchy while Turkey is in the middle and Lithuania prefer a flexible work environment.

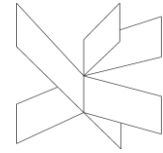
- Impact on our team: Even though our countries power distance is so different we decided to work as a team on equal ground. In this case some parts of SCRUM worked better in our team than others. For example: weekly meeting worked great in our team but having a product owner was problematic. Since we found it hard to give orders and preferred to work organically finding answers through discussion.
- Individualism: The fundamental issue addressed by this dimension is the degree of interdependence a society maintains among its members. It has to do with whether people's self-image is defined in terms of "I" or "We". In Individualist societies people are supposed to look after themselves and their direct family only. In Collectivist societies people belong to 'in groups' that take care of them in exchange for loyalty. While Romania and Turkey have relatively low individualism Lithuania has almost double that. This means that for Lithuanians it is very important to succeed on their own unlike for Romania and Turkey, they prefer to see the team succeed.
 - Impact on our team: Although Lithuanians are high on the scale of Individualism we found that we prefer to work as a team and seek for the success of the whole team instead of our own as an individual. This show in our help for each other and that our power distance inside the team is low.
- Masculinity: A high score (Masculine) on this dimension indicates that the society will be driven by competition, achievement and success, with success being defined by the winner / best in field – a value system that starts in school and continues throughout organisational life.



A low score (Feminine) on the dimension means that the dominant values in society are caring for others and quality of life. A Feminine society is one where quality of life is the sign of success and standing out from the crowd is not admirable. The fundamental issue here is what motivates people, wanting to be the best (Masculine) or liking what you do (Feminine).

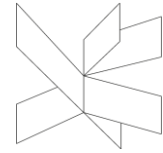
All three countries are considered feminine, especially Lithuania. This means that in general our countries are soft spoken, diplomatic and can not take praise.

- Impact on our team: Our team reflects our countries charts greatly. Because as mentioned before in power distance part we prefer discussion and diplomacy and the wellbeing of the team. Rather than trying to be the best.
- Uncertainty avoidance: The dimension Uncertainty Avoidance has to do with the way that a society deals with the fact that the future can never be known: should we try to control the future or just let it happen? This ambiguity brings with it anxiety and different cultures have learnt to deal with this anxiety in different ways. The extent to which the members of a culture feel threatened by ambiguous or unknown situations and have created beliefs and institutions that try to avoid these is reflected in the score on Uncertainty Avoidance. While Lithuania has a bit lower uncertainty avoidance level overall the three countries have very high uncertainty avoidance scores.
 - Impact on our team: Uncertainty avoidance can be seen very clearly in our team. We tend to avoid problems that may occur later down the line by having quite strict notes about our work and very defined goals and work schedule.
- Long term orientation: This dimension describes how every society has to maintain some links with its own past while dealing with the challenges of the present and future, and societies prioritise these two existential goals differently.



Normative societies, which score low on this dimension, for example, prefer to maintain time-honoured traditions and norms while viewing societal change with suspicion. Those with a culture which scores high, on the other hand, take a more pragmatic approach: they encourage thrift and efforts in modern education as a way to prepare for the future. Lithuania has a very high score in long term orientation which means that they are pragmatic and adaptable. On the other hand Romania and Turkey have very intermediate scores therefore no dominant cultural preference can be inferred.

- Impact on our team: Our teams orientation leans more towards the high side. This is due to the fact of us as a team being adaptable and having a strong drive to achieve results.
- Indulgence: One challenge that confronts humanity, now and in the past, is the degree to which small children are socialized. Without socialization we do not become “human”. This dimension is defined as the extent to which people try to control their desires and impulses, based on the way they were raised. Relatively weak control is called “Indulgence” and relatively strong control is called “Restraint”. Cultures can, therefore, be described as Indulgent or Restrained. Turkey has a very average score as such no dominant cultural preference can be determined. On the other hand Romania and Lithuania have a very low score. This means that these societies are very restrained have a tendency for cynicism and pessimism. Also not much time is dedicated for leisure.
 - Impact on our team: Our teams Indulgence is a mix of both high and low some of us prefer leisure time over work and some work over leisure. The same can be said about pessimism and cynicism. Therefore we often have to reach a consensus about these topic. When to rest and when to work.



3 Project Initiation

Daria-Maria Popa & Lukas Suslavicius

Lukas Suslavicius

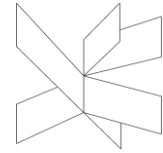
Project initiation for our group was swift and decisive. From the very beginning we enjoyed having the freedom to choose our project theme while still having a general idea of what rules it needed to conform to.

After a short discussion we decided on two possible project ideas that seemed meaningful, interesting and relevant to the subject. We quickly drafted the minimum requirements for both ideas and after a few days of refining them we presented them to our supervisors.

One of them described a planner app, where admins could create and manage private rooms intended for planning activities. The other idea, the one we ended up creating, was titled "Puppr – a dog dating app!". The "dating" aspect of this described a more unusual take on human communication, one where people would bond through their common love for dogs. An app that would put the focus on dogs more, letting people showcase their dogs like never possible before, organize meetups and interact freely, while an admin would manage everything to ensure good functionality and respect in the community.

Daria-Maria Popa

"Puppr" the idea we put most effort in and felt would challenge, entertain and prove our skills gained by the end of the semester. The idea came from a mix of the general idea of creating a blog/social app project, current social media influences and the drive to do something unique and relatable for people. Those are the main reasons the "social platform for dog owners" came into existence.

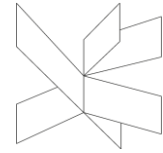


As a general rule, we all wanted to work as effectively on this as possible while still being creative and fun with it. Based on these common conceptions of how the project should be structured and dealt with we created and signed our group contract. The contract had three major points: communicate, be honest, participate.

It described our general consent to work hard in order to achieve the result we were wishing for, to communicate in a friendly and helpful manner and if any problems occur solve them together, as a team.

The group contract represented the first step in organizing our workflow even before deciding on the exact Scrum sprints and iteration planning by noting down that we agree to meet at least once a week for the project.

Overall, the inception of the project went over quick and smoothly and we were all ready to start planning the project and slowly start working and doing our sprints.



4 Project Description

Daria-Maria Popa & Lukas Suslavicius

Daria-Maria Popa

The work for the project description started early on, right after the unanimous decision on the iterations and vision for the project. The project description work focused on identifying the problem domain together, defining it and setting relevant project boundaries.

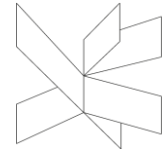
For the first step of the project description, the background description, the team described the importance of dogs both socially and economically, put an emphasis on the pet market being always in need of new products catering to user needs and described the most known social apps today and their lack of adaptation to this special niche of clients.

The problem statement set the scene for the problem domain taking note of what we defined as the main problem statement, the one explaining the need for the project and system, and the question dealing with the different areas of the problem, from needed features to the behavior and thinking of the people that would be interested in this app.

Definition of purpose represented the statement concluding the whole reason the "Puppr" app was needed and relevant and to who exactly it was intended for use while the delimitation enumerated the objectives or features the project will not have. These delimitations were based on what the team considered unnecessary for this app, out of its scope, or impossible to achieve in due time.

Lukas Suslavicius

Because this project was to be done using Unified Process (UP) and Scrum, the next methodology talked the team roles, sprint length and Scrum meetings. Choosing SCRUM role was relatively easy because we had several exercises in SWE where we



had to work in SCRUM teams. This allowed us to know what we want and what roles best suit us. When we decided on our roles which ended up as Bogdan – product owner, Lukas – SCRUM master, Daria and Natali – SCRUM/ Development team.

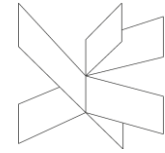
For the time schedule of our project we decided on 13 weeks because it was a clear timeframe and fit our one week long sprints perfectly, ending right before the final deadline of the project.

Daria-Maria Popa

We were keen on starting the iterations early because we knew it would take some time to get accustomed to the sprint length and amount of workload per sprint. The whole team also wanted to have a good and generally stable workflow by the time we were in the elaboration phase of the project and most of the coding work was being done.

Out of the four risk we outlined in the project description we handled all of them according to expectations except for choosing goals poorly. While we never hindered our productivity by picking goals that were wrong for a sprint we did on occasion decide on a goal that was not feasible, but with time we learned to pick goals better and work smarter on our project.

In conclusion, the project description was done taking into consideration and the advice received and guidelines and so needed only minimal tweaks later down the line of the project work which helped tremendously later on and shorten the overall time it took to finish the whole documentation.



5 Project Execution

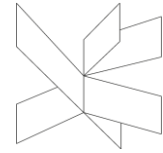
Daria-Maria Popa

Setting the time schedule and sprint plan in the project description was extremely helpful in the long run when executing the project. Although following the schedule proved to be a bit tricky at first, having one made all of all aware of the time we had to complete our tasks and pushed us to work harder to achieve that and have no extra work for the next sprint.

In order to follow SCRUM and have our work structured we made an Excel file. In it we would write down our SCRUM sprint plans and results, while also having a section where each team member could write what they achieved on a daily basis. This was of course managed by the SCRUM Master, Lukas.

The general schedule was pretty much respected with slower sprints intervening from time to time. Since most of if not all SDJ classes helped or were required for project completion, as we learned new skills our project moved forward to completion each week. This is also why the focus on the code work for the project only started in one of the later sprints, sprint 5/6, when the knowledge we have gained was enough to start creating an architecture we would be proud of.

Our workflow followed the Unified Process principle of being “iterative in the small”, and, combined with the SCRUM principles, we focused on delivering the best quality product each week that was ready to be a part of the final system. With the help of the Product Manager is assessing the needed quality and some little fixed along the way in how we worked, by the third sprint we were producing good quality code or documentation which could be directly introduced in the software, respectively the reports.



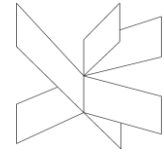
The workload was most of the time split evenly, with everyone having an equal saying in what they wanted to focus next on based on what was already done. The weekly SCRUM meetings helped set the weekly objective list which we all split and worked on together or alone in order to achieve. At the end of each sprint, at the next meeting, everything was evaluated and marked as done, partially done or not started based on what we as the team manage to deliver.

In the Inception phase of the project, as aforementioned, we mainly focused on finding the problem domain, drafting the first list requirements, and doing our project description. During the Elaboration we managed to get a list of requirements which was very little changed until the end, make the first version of the user case diagram and the descriptions for it, and start on the base architecture of the client-server system.

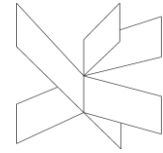
Construction contained sprints with more tasks that took generally less time to do, alternating between new parts of code, testing it and once it reached a satisfactory level adding new information to the documentation in relation to it. By the end of the construction the code was technically finished, with only small adjustments to be done on it and the closing chapters of the documentation needing a bit more work on.

The transition was represented by the last week before the deadline and consisted in the last part of documentation being polished and ready for hand in, and an overall look over all the documents, diagrams and code.

The project results were more than satisfactory, having an end product with all the requirements fulfilled and some added fixes or functionalities along the way. The "Puppr" app functions just as expected, having a nice design and user interface too beside the functionality. The documentation was thorough and went through all the steps taken to create the final product, from the project description to the result and possible project future.



Overall, our project execution was planned effectively using the specified methodology, was monitored closely by the whole team, and showed great success in creating the app at a comfortable and efficient rhythm.



6 Personal Reflections

Personal reflection – Daria

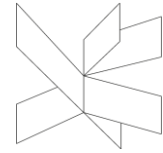
This semester project represented the first time we all got to work on a subject picked by us with was very exciting and personally made me want to work even harder on it.

Our team, “Team cake” was formed before the semester began, and we were more than happy to work together as friends and teammates. We already had a better formed idea of what we each wanted from a project and how we worked from last semester so we agreed we would form a good team.

I knew that we are all pretty motivated and the type of people to want to start work early and work effectively, and as a result the planning took part in the first or second meeting we had. “Puppr” first came as a “Tinder, but for dogs” idea but soon shifted to be a much more social and Twitter influenced app which we all preferred. It was a fun idea that we knew would take hard work to execute right but we wanted to show the supervisors and ourselves that we can do it.

We decided early on we wanted to use Git for the coding part because it would be much easier to add the code after each sprint and at the end commit the whole branch and create the final code version. As for the general planning part of the whole project, we of course used SCRUM and UP methods, respecting the project requirements, but to also try this agile working technique for ourselves. I really liked being able to work agile for the first time and I think it helped us significantly in reaching our goal in time even if we had to learn to apply it correctly along the line.

For the “Puppr” app we wanted it to have a fully functional posting, commenting, and liking functionality, while also proving managing and sign up functions, while presenting this whole data in a pretty and unique format. The project included a



significant list of requirements, but we knew that if we stuck closely to our plan, we would be able to make it in time with some time to spare for other minor fixes.

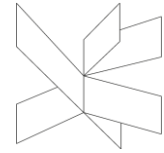
I am very happy with how the whole planning went considering it was the first time using most technologies and we learned most things helpful for the project later in the semester. I think SCRUM is especially one of the methods I would like to use in future project, especially that now there will not be any time lost of figuring out how to use it. The concept of having the SCRUM notes we made each week, at the end of each sprint, helped us prioritize better and realise when we were running behind schedule or should limit the tasks and focus harder on just a couple.

All the risks we mentioned when creating the project description were very well handled and by the end of the project, we were all very happy with how everything worked. I do not think there is anything that was “unsuccessful” seeing how we achieved our requirements, but of course new features to make the app even better could always be added, which we mentioned in the project future part of the project report.

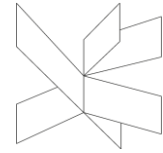
The group contract I believe was fully respected by all of us, there were no conflicts and we handled and shared the work together, always communicating what we thought would work better for the sprint or what should be done differently. I really liked working in this team and grew to appreciate even more having people to ask for advice or help when something was not working as expected.

Responsibilities were shared, with no one doing only a thing during the whole length of the project but changing the focus on something else occasionally. While the SCRUM roles never changed, we all helped with different documentation parts or code at certain times or read over some else part and gave advice other times. This helped making sure everyone was content with the final product and that we all contributed as equally as possible.

Overall, working in this team was great and I would not change anything. Even if splitting work can sometimes be stressful, it is much harder doing it alone and I am



grateful for the help and own input my teammates provided. I am very happy and proud of this project and I hope the mutual effort put into making this idea come true shows in the "Puppr" app.



Personal reflection - Natali

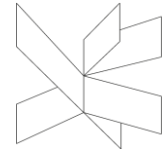
We decided our group members at the end of the first semester. Our experiences from the first semester affected this decision. I think we have very different personalities, but we had a good relationship with our group work's earliest days.

We prepared a group contract and documented group rules in writing at the beginning of the semester. I think having a group contract was beneficial for us because it ensured that all group members would be aware of their responsibilities and the consequences of failing to comply with group rules.

As a person who prefers to work alone, group work is always a little bit challenging for me. Although we had group work in the first semester, this semester was a completely different experience for me since I could feel the effects of having different personal characteristics. However, we could manage to work together without any conflict and had pleasant group work.

Since we were used to working together with people from other cultures thanks to our former group project, being from different cultures and having different backgrounds did not have any negative effect on our group. After all, working together with people with different characteristics in a multicultural environment is quite interesting and pleasant. I think each semester project teaches us something new about group work and these experiences will be useful for our future career.

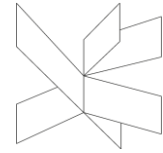
The idea of developing a social media application for dog owners was very interesting and exciting from the beginning and I really had fun while working for Puppr app. This app provided us an opportunity to improve our skills in many different subjects. We could practice our knowledge and skills in JavaFX, design patterns, database systems, RMI, agile working, SCRUM method and so on.



Unlike the first semester, we had a SCRUM meeting every Wednesday. They were very efficient and beneficial. In these meetings we gained a good understanding of what work has been done and what work remains, so we could perform our tasks in a planned way. Therefore, we did not have any problem with time management and completing tasks on time. The meetings also helped us to improve our team cohesion, we could easily share tasks and responsibilities. Each group member was aware of the importance of performing group tasks in time and working cooperatively.

We had unpredictable circumstances in this semester because of Corona. We had online education and we could not spend much time with our group members. Most of our communication was on the internet. This situation could have caused negative effects on our group work, but we could overcome these difficulties and stay motivated.

In conclusion, it was a good experience to work in this group. I am glad that I learnt a lot of new knowledge during this semester project. I hope I will have the chance to use all this knowledge in an efficient way in future group works.



Personal reflection - Bogdan

Our group was formed even before the second semester started because of our experiences from the first semester and because we already knew the working styles of each other. In my opinion, everyone in the group respected the group contract we signed at the start of the semester which made cooperation very easy and ensured our success.

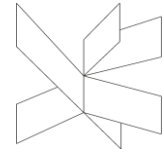
I feel like, as a team, we got along really well from the beginning when we had to come up with project ideas and I believe everyone loved the idea of working on a social media platform targeted at dog lovers from the very start.

All of our documents and code have been very well organized using Google Drive and Git (with GitHub) which made everything accessible to everyone instantly. I think we split the responsibilities very well between us and no one was left feeling overwhelmed. One thing I particularly appreciated this semester was the creative liberty I was given since I was the one to design the views. While our roles did not change throughout the semester everyone was providing feedback which meant we were occasionally shifting focus and we agreed with everything the others were working on.

As a team, we paid particular attention to the requirements and by the end, we managed to fulfill each one of them which meant that our project was ultimately a success. When we created the project description we listed some potential risks which I believe have been dealt with without any problems.

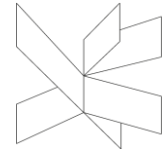
Working on the project with SCRUM and UP took some getting used to at first but in my opinion, everyone got accustomed to it very fast, and working agile has proved to be extremely beneficial to our team's productivity.

I personally enjoyed working with SCRUM much more than Waterfall mainly because of the flexibility. We had weekly meetings that made the whole process more dynamic and just talking about what we did, what we struggled with, and what we are



going to do made teamwork and cooperation much more manageable even while working almost exclusively online because of quarantine.

In conclusion, I am very happy with how the project turned out and working as part of “Team Cake” has been extremely rewarding as all of my teammates were remarkably helpful giving input on everything and I believe this experience has been invaluable for me as a software engineer and team member.



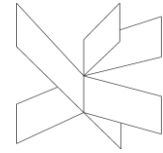
Personal reflection - Lukas

By learning from the group work of the previous semester, I decided to form a group consisting of members whom I personally knew. In my opinion, familiarity was the main reason why our team worked so great together and were respectful as well as managed to follow the group contract we set out for ourselves.

For this semester while working in groups we used SCRUM which for me was amazing. I was the SCRUM master, which allowed me to schedule meetings, manage details and plan our teams progress during the project. Of course, we still worked as a team and sometimes managed to make our roles in the team blend. Blending roles helped out tremendously, while we never outright changed our roles in the SCRUM team, the occasional blending of roles allowed us to work more effectively. Because of it, we would cover each other shortcomings. SCRUM meetings happened on Wednesdays. Their structure was:

- How is everybody doing both project wise and general life wise.
- Going over the goals that were set for the sprint.
- Going over the goals one by one and taking notes on what and how everything was done.
- Giving an overall conclusion of the sprint by deciding if we did enough work, set enough goals.
- Deciding on next weeks goals.
- Talking about everybody's wellbeing and some general topics.

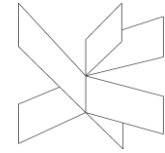
Usually SCRUM meetings would last for about an hour. These meetings were invaluable. Because it showed if some team members were overworking or stressing. This allowed us to tailor our sprints to have less of a workload. For example, during Easter we decided to do a bit less work by combining two sprints into one effectively. This allowed us to rest and be motivated and hard working after Easter.



The outcome of the project is more that anybody could've asked for. Firstly, the project looks wonderful and has a beautiful colour pallet which is not heavy on the eyes. And most important the program works. There are a few problems of course, but in my eyes a project like this is never over. It can always be improved, however due to the time constraints we had to temper our expectations for the final product. But as it was discussed in the section of "project future" we are brimming with ideas and see the endless possibility for our project.

The second semester for myself was a major improvement on the first. The Semester project gave free reign on doing whatever you wanted. This was in stark contrast with the first semester. In the first semester we had specific project and guidelines we needed to follow. This semester whilst having restrictions we could do whatever we wanted. In my opinion, this made our team shine because it allowed us to be creative. And our creativity combined with the restrictions, allowed us to make one of the best projects we could have made.

In conclusion, if not for the restrictions set out by our supervisors and my teams passion to create, we would have never made such a great project. Honestly, I don't see a single flaw in our team that could be complained about or in the supervision from the teachers.



7 Supervision

Daria-Maria Popa & Lukas Suslavicius

Lukas Suslavicius

Our supervisors for semester project were Henrik Kronborg Pedersen and Joseph Chudwudi Okika. Our experience with the teachers and anyone from VIA is very positive. If we needed help or answers the response was always quick precise and very helpful, and this semester showed no change.

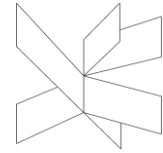
Despite that in the last part of the semester communication was done solely online, asking for advice from the supervisors was still quite easy and the responses were quick.

During the semester we got help individually or as a team multiple times in order to correctly understand certain topics or do parts of the project. One of the first supervisions we had were in the very beginning of the semester regarding the vision of the project and the project description work. We found this one particularly helpful because it set us on the right path early on and gave us a better understanding on how to structure our work.

Most supervisions were either related to the Java, database or documentation and testing side of the project separately and so they were done with only one of the two supervisors present or asked. Despite this, they were still tremendously helpful in fixing little bugs, tackle misunderstandings or just clarifying new topics and how to apply them to our particular app.

Daria-Maria Popa

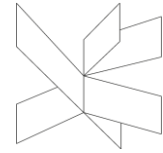
While in the construction phase, most supervisions were naturally needed in the coding department and involved solving together with the supervisor some more or less complex errors found in the code of the project or other semester exercises or



assignments, the answer to which still helped when creating the “Puppr” app. Both Joseph and our other SDJ teacher, Steffen, were particularly helpful during this time.

In the last parts of the project, both Henrik and Joseph were helpful by pointing out little changes that should be done in the system or diagrams, giving advice on certain topics we had questions on and resolving any doubts we had before hand in.

Overall, the supervision for the second semester was excellent and we hope this kind of supervision continues in other semesters. The sessions for supervision were short and to the point, the answers came quick and were detailed enough and the overall impression left after the supervision was always positive.



8 Conclusions

Daria-Maria Popa

During the project, a particular idea was strongly reinforced by the results in our work and workflow: you need to plan ahead and work smart.

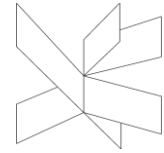
Being the first time using UP (Unified Process) and SCRUM it was also the first time for all of us that we got to try working iteratively and in sprints on a software development project, we were keen on learning how to apply these methodologies and it proved to be extremely rewarding to do so.

That is why we think in group work you should always try and work in an evolutionary way, start in a fixed point and work towards the goal but never be restricted from turning back and improving on past work. It helped a lot to learn from our mistakes and have the ability to fix parts after we understood how to do something better.

A team or team member should not be afraid to try and do something new even if they have little idea how to do it at first. After you take the initiative it is much easier to find the creativity and answers to do it correctly afterward. Any progress is a good step in motivating the whole team to try and figure out the solution to any problem together.

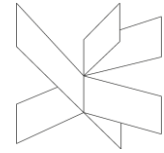
Documenting the code is often perceived as the hardest part for this kind of projects and we would probably agree too. Despite this, starting parts of the documentation, designing something right after you are sure on a part of the system and adding parts over time, eases the documentation process exponentially. It is something we learned from experiences comparing the two semesters and we would advise any team to work incrementally and over time on any project.

To summarize, the list of recommendations for group work our team has are as follows:



- Plan ahead and start early
- Work evolutionary and smart rather than trying to do everything right the first time
- Start early with any documentation work and try to explain the project as best as you can
- Communicate anything to your teammates and supervisors whenever you are unsure about a certain thing
- Try your best and show you are interested in what you are doing

All in all, the recommendations match our own beliefs about project work and how we tried to work on this project. We are proud of the end result and want for the goals enumerated earlier show in what we did as well.

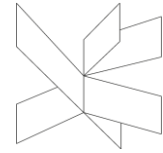


Appendices

APPENDIX A GROUP CONTRACT

APPENDIX B SCRUM NOTES

APPENDIX C SCHEDULE



APPENDIX A GROUP CONTRACT

Group Name
Team Cake (4)

Date:
02/23/2020

These are the terms of group conduct and cooperation that we agree on as a team.

Participation: We agree to....

Work hard and do everything we can in order to get the best results.

Communication: We agree to...

Always communicate our intentions with the other team members, respect each other's

ideas and participate in group discussions.

Meetings: We agree to....

Meet at least once a week(Wednesday), be punctual and let the others know if we

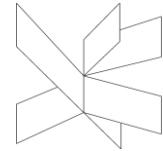
cannot make it.

Conduct: We agree to....

Be respectful, friendly, helpful and focus on teamwork in an informal environment.

Conflict: We agree to....


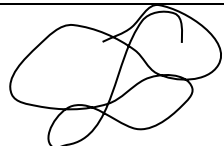


Solve all our problems, together as a team, in a civilized manner before the conflict gets out of hand.

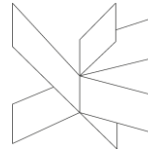


Deadlines: We agree to....

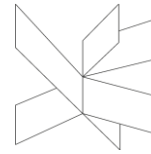
Start working early, set realistic goals and finish before all the deadlines.

Other Issues:

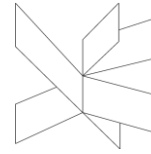
Group member's name	Student number	Signature
Daria-Maria Popa	293087	
Lukas Suslavicius	293717	
Natali Munk-Jakobsen	293132	
Bogdan-Alexandru Mezei	293137	



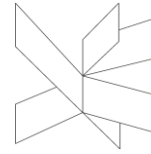
APPENDIX B SCRUM NOTES



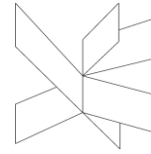
	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 1	1. Fix the project description: formating, background, problem statment b and c, SCRUM. 2. Use cases 3. JavaFX 4. Requirements 5. Schedule	1. 03-04 User cases for the user level (half complete) 2. 03-05 Fixing the project description 3. 03-06 4. 03-07 5. 03-08 6. 03-08 7. 03-09	1. 03-04 2. 03-05 3. 03-06 4. 03-07 - Requirements 5. 03-08 6. 03-08 7. 03-09	1. 03-04 2. 03-05 3. 03-06 Created the shcedule in excel. 4. 03-07 5. 03-08 Made the skeleton for login and home view 6. 03-08 Worked a bit more on the schedule 7. 03-09	1. 03-04 2. 03-05 3. 03-06 - Did some preliminary JavaFX desing work. Made a main menu. Main menu only has sign up and Log in. This feels wrong perhaps a better alternative solution would be a wall where people even if not loged can view but not interact and at the top or bottom corner have a sign up/Log in option, should discus implementation possibilities. Made a UI for the previously mentioned idea still feels to empty and missing stuff. Did not begin JavaFX for user walls and sign up. User walls for we should decide if we use the main menu method with the madnatory sign in or not. Sign up we should decide on what a user must input. Basically we neet to talk a lot with the project manager and each other. 4. 03-07 5. 03-08 6. 03-08 7. 03-09 8. 03-10 Listed the requirements. Have not sorted them by importance. Made 17 of them don't know if we need more.	1. Fixed all of it timetable still missing 2. Mainly done. Missing block user and liking comments and hall of fame 3. Made a mock up only the login also made some mock ups for the wall. 4. should compare uses cases 5. nope in progress There was enough time not everything was finished to completion but it was either imposible or unclear if more needs to be added



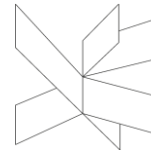
	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 2	1. Use cases 2. Schedule 3. Domain scetch 4. Refine Requiremetns 5. UI connecting coding button working 6. Report formating	1. 03-11 2. 03-12 Added use cases for managing users and hall of fame Modified the managing profile to take into account adding a new pet 3. 03-13 4. 03-14 5. 03-15 6. 03-16 7. 03-17 8. 03-18	1. 03-11 2. 03-12 - Requirements based on use cases 3. 03-13 4. 03-14 5. 03-15 6. 03-16 7. 03-17 8. 03-18	1. 03-11 2. 03-12 - Started working with the UI. Created the loginView and the homeView as a sketch. 3. 03-13 - Finished up both views and polished them. 4. 03-14 5. 03-15 6. 03-16 Added functionality for some buttons (closing the window and traversing) 7. 03-17 8. 03-18	1. 03-11 2. 03-12 3. 03-13 4. 03-14 5. 03-15 6. 03-16 - Started formating on the Project report did not finish becaue want to ask Henrik if some of the thing i wanna do allowed since Michael mentioned that last semester our formating was strange and I don't want to repeat the same mistakes. 7. 03-17 8. 03-18	1. Use cases ar mostly doen and all are correct some more might need to be added 2. Not hing was done dragging from sprint one1 3. Refined and streamlined seems correct. EMAIL HENRIK! 4. Started the Domain model will carry over to Sprint 3. 5. Most of the UI is conneceted 6. Did the title page



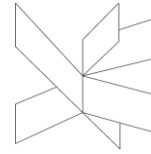
	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 3	1. Add requirements for sorting and editing. 2. sort requirements. 3. Testing SWE assignment must be done by Monday. 4. Setup GitHub !!!!! 5. Finish the Domain in Astah. 6. Finishing up the view. 7. Making the Model classes 8. Do the SDJ assignment chat room thingy	1. 03-18 Added requirements for sorting and editing 2. 03-19 3. 03-20 4. 03-21 SWE assignment 5. 03-22 6. 03-23 7. 03-24 SDJ Assignment 8. 03-25	1. 03-18 2. 03-19-SDJ assignment 3. 03-20 4. 03-21 5. 03-22 6. 03-23-SWE assignment 7. 03-24 8. 03-25	1. 03-18 2. 03-19 Set up github 3. 03-20 4. 03-21 SWE assignment 5. 03-22 Added profileView and polished homeView. 6. 03-23 7. 03-24 SDJ assignment 8. 03-25	1. 03-18 2. 03-19 3. 03-20 4. 03-21 Did the test case assignment with the group 5. 03-22 6. 03-23 - Sorted the requirements by priority need to be checked if sorted adequately. 7. 03-24 - worked on the assignment for SDJ1 8. 03-25	1. Accomplished 2. Sorting is done the team still needs to look over it. 3. It is done and submitted 4. Setup is done 5. The domain is not finished some work was accomplished but not done. 6. 2 views are finished and polished but profile is not done yet 2/3 done 7. It was not done. 8. It is in the testing stages so basically done Overall this week was slow since we had a lot of assignments



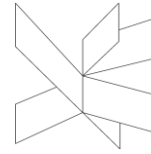
	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 4	1. Finish the Domain model. 2. SWE assignment for Monday. 3. Finishing up the view. 4. Some dicumentation ir project report 5. Some logic viewmodel	1. 03-25 2. 03-26 3. 03-27 4. 03-28 Domain model done 5. 03-29 6. 03-30 7. 03-31 8. 04-01	1. 03-25 2. 03-26 3. 03-27- SWE assignment 4. 03-28 5. 03-29 6. 03-30- added some model classes 7. 03-31 8. 04-01	1. 03-25 2. 03-26 3. 03-27 4. 03-28 Debugging, mostly visual stuff 5. 03-29 6. 03-30 7. 03-31 8. 04-01	1. 03-25 2. 03-26 3. 03-27 - did SDJ assigment UML and general code cleanup 4. 03-28 - finished UML with Daria. 5. 03-29 6. 03-30 7. 03-31 - wrote the analysis and put in the requirements the Analysis part will have to be changed not very coherent right now. Ned to talk about functional and non functional requirements. It says to put in UML diagrams and use cases but i think we need to look it over as a team and decide a bit more 8. 04-01	1. Domain is done 2. Swe is submitted 3. nothing was done 4. Requirements and introduction was done 5. did nothing



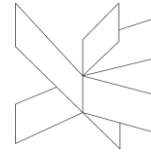
	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 5 + Sprint 6	1. SDJ RMI assignment individual work submission on 10th and then the upload to it's learning on 13th 2. Start viewmodel 3. continue the documentation 4. Have a nice easter 5. Make a method for wall posts 6. DBS assignment. 7. Fill out the schedule according to work done until now	1. 04-01 2. 04-02 3. 04-03 4. 04-04 - worked on sdj assignment 5. 04-05 - coded part of viewmodel 6. 04-06 - added more to code, especially model and did some base logic for buttons and mvvm, worked on sdjassignment and did the uml for it 7. 04-07 8. 04-08 9. 04-09 10. 04-10 11. 04-11 12. 04-12 13. 04-13 14. 04-14 15. 04-15	1. 04-01 -sdj assignment 2. 04-02 3. 04-03 4. 04-04 5. 04-05 6. 04-06--worked on database and database connections 7. 04-07 8. 04-08--created server side classes 9. 04-09 10. 04-10 11. 04-11- RMI connection between client - server sides 12. 04-12 13. 04-13- worked on some parts of view and viewmodel 14. 04-14 15. 04-15	1. 04-01 2. 04-02 3. 04-03 Made the final version for profile view 4. 04-04 SDJ assignment 5. 04-05 6. 04-06 Added a manage profile view. 7. 04-07 Solved some visual bugs. 8. 04-08 9. 04-09 Polished up manage profile view. 10. 04-10 Uploaded SDJ assignment 11. 04-11 12. 04-12 13. 04-13 14. 04-14 15. 04-15	1. 04-01 2. 04-02 3. 04-03 4. 04-04 5. 04-05 6. 04-06 7. 04-07 8. 04-08 9. 04-09 10. 04-10 11. 04-11 12. 04-12 13. 04-13 put in the use cases in the project report 14. 04-14 15. 04-15	1. Uploaded 2. -database and database connections. -All server side classes. -RMI connection between server-client. - signup and login views are working. -profile and home views partly working also did a lot of the view part profile view manage profile 3. added use cases 4. Easter was nice yet boring 5. add card done 6. uploaded and done 7. nothing was done



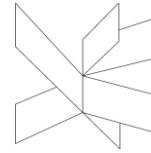
	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 7	1. Manage profile and create profile need to be done in the view 2. Flesh out Viemodel 3. And round out Model 4. Add design to project report	1. 04-15 2. 04-16 linked new view parts and implemented functionalities 3. 04-17 modified some home view functionalities 4. 04-18 basic diagraming for design part based on current architecture 5. 04-19 6. 04-20 debugging code parts 7. 04-21 8. 04-22	1. 04-15 2. 04-16 3. 04-17-worked on some parts of view and viewmodel 4. 04-18 5. 04-19 6. 04-20 7. 04-21 8. 04-22	1. 04-15 2. 04-16 Polished manage profile view again. 3. 04-17 4. 04-18 Final version for create profile view. 5. 04-19 6. 04-20 7. 04-21 8. 04-22	1. 04-15 2. 04-16 3. 04-17 4. 04-18 5. 04-19 6. 04-20 7. 04-21 put in the UI design for the project report very minimal weill need to be expanded upon started the UML 8. 04-22	1. is done and on GitHub 2. some work was done 3. adding post was don and dog cards some work was done on liking added date and time for posts 4. UI added UI design explanation 5. started the UML



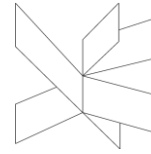
Sprint 8	<ol style="list-style-type: none"> 1. viewpost post view needs to be done 2. finish as much UML as i can for now 3. show comments parts need to be done 4. addcomment/comment card need to be done 5. modify post cards to have a "see post" button which will open that new post view 6. delete and edit buttons need to be done 7. add usertype 8. DBS assigment 	<ol style="list-style-type: none"> 1. 04-22 2. 04-23 done the dbs assignment 3. 04-24 4. 04-25 5. 04-26 fixed bugs regarding delete/edit 6. 04-27 looked over usertypes and switching between them in app 7. 04-28 8. 04-29 	<ol style="list-style-type: none"> 1. 04-22 2. 04-23 3. 04-24 4. 04-25- added edit/delete post and dog buttons and implemented functionalities 5. 04-26- added usertypes to the code and database 6. 04-27 7. 04-28 8. 04-29 	<ol style="list-style-type: none"> 1. 04-22 2. 04-23 3. 04-24 Created viewPostView 4. 04-25 Solved some visual bugs. 5. 04-26 6. 04-27 7. 04-28 8. 04-29 	<ol style="list-style-type: none"> 1. 04-22 Did the whole serverside UML for the code we currently have 2. 04-23 3. 04-24 4. 04-25 5. 04-26 6. 04-27 Did the UML 7. 04-28 Finished the UML. Still needs to be connected but has all the methods and classes that are in the latest code. Probable are typos and mistakes need to be looked over. 8. 04-29 	<ol style="list-style-type: none"> 1. Was kinda done it's in the proces of bieng finished 2. UML is finished and uploaded may have errors 3. Not been able to work on this 4. the view almost ready 5. we have the button show edit and delete editing and deleting is wokring but showing is not working but it is ready. 6. works 7. works admin type and admin user types are added 8. Upleaded and done
----------	--	---	--	--	---	---



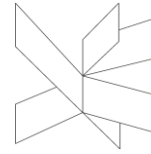
	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 9	<ol style="list-style-type: none"> 1. Send the Email about us being in the same group for next semester!!! individual emails. 2. changing the view to be "nicer" 3. commenting the code 4. work on the project report 	<ol style="list-style-type: none"> 1. 04-29 - sent the mail 2. 04-30 - linked some view parts 3. 05-01 - continued view word and fixed some bugs 4. 05-02 5. 05-03 6. 05-04 - worked on sdj assignment 7. 05-05 - worked on sdj assignment 8. 05-06 	<ol style="list-style-type: none"> 1. 04-29 2. 04-30 3. 05-01 4. 05-02-edit/delete comment 5. 05-03-worked on some other parts of view and viewmodel 6. 05-04 7. 05-05 -SDJ assignment 8. 05-06 	<ol style="list-style-type: none"> 1. 04-29 2. 04-30 Went through most of the views and fixed all visual problems (alignment, weird fonts, spacing) 3. 05-01 4. 05-02 Solved some visual bugs. 5. 05-03 Also fixed the rest of the views 6. 05-04 Solved some visual bugs. 7. 05-05 Solved some visual bugs. 8. 05-06 	<ol style="list-style-type: none"> 1. 04-29 2. 04-30 3. 05-01 4. 05-02 5. 05-03 6. 05-04 worked on assignment 4 7. 05-05 worked on assignment 4, Did some project report work. 8. 05-06 	<ol style="list-style-type: none"> 1. Daria and Lukas sent it . Natali and Bogdan will send it today. 2. the views are made nicer :D 3. Commenting was started. 4. some work was done.



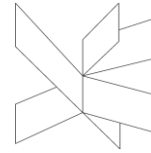
	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 10	1. finish SDJ assignment 4. 2. Guest actor feature 3. continue commenting the code 4. Project report 5. And testing/ decide who want to take it up and do it 6. add in the proces report that we have the SCRUM notes in the apendicies	1. 05-06 continued commenting the code 2. 05-07 continued commenting the code 3. 05-08 finished the sdj assignment 4. 05-09 looked over minor code bugs 5. 05-10 fixed some more on the code 6. 05-11 started on project report 7. 05-12 8. 05-13	1. 05-06 -SDJ assignment 2. 05-07- Guest actor feature added 3. 05-08-blocking/unblocking user 4. 05-09 5. 05-10 6. 05-11 7. 05-12 8. 05-13-worked on some other parts of view and viewmodel	1. 05-06 2. 05-07 SDJ Assignment 3. 05-08 Solved some visual bugs. 4. 05-09 Solved some visual bugs. 5. 05-10 6. 05-11Made a plan for testing 7. 05-12 8. 05-13	1. 05-06 Did the non-functional requirements for project report. Has missing parts and a bunch of nonsense needs to be fixed. Table done in accordance to the documentation that was givent in the guidelines. 2. 05-07 Reformated the project description added some information about SCRUM needs to bee looked over but so farr seems decent. Also needs to be turned into pdf. Created a new folder for upload ready files. 3. 05-08 4. 05-09 5. 05-10 6. 05-11 7. 05-12 Did the whole group description in Procces report 8. 05-13 Wrote something about prject initiation in procces report will need to be changed i feel.	1. uploaded and finished 2. implemented 3. almost done 4. procces report and description were worked on 5. Bogdan 6. added



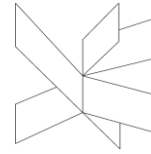
	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 11	1. Project report testing, implementation, Design 2. Finnish procces 3. Some tweaks for the views 4. Finnish commenting the code 5. Start testing	1. 05-13 2. 05-14 - finished commenting the code 3. 05-15 statrted work on analysis 4. 05-16 polished the use cases and use case diagrams 5. 05-17 continued with analysis work and fixed some requirements 6. 05-18 finished analysis chapter 7. 05-19 8. 05-20	1. 05-13 2. 05-14 3. 05-15- fixed some code bugs 4. 05-16 5. 05-17 - worked on writing project report-implementation part 6. 05-18 7. 05-19 8. 05-20	1. 05-13 Created the intro for testing. 2. 05-14 Solved some visual bugs. 3. 05-15 Wrote about half of the test cases 4. 05-16 Finished the test cases 5. 05-17 Solved some visual bugs. 6. 05-18 7. 05-19 8. 05-20	1. 05-13 2. 05-14 Finished most of the procces report intorduciotn, personal reflection and conclusion are not done. Also it is rather short probably will need to comeback to it and add information. 3. 05-15 4. 05-16 5. 05-17 6. 05-18 7. 05-19 8. 05-20	1. testing not done, almost finished, analsis is done. 2. finihsed procces 3. done 4. code is commented 5. code not doen testing not started



Sprint 12	<ol style="list-style-type: none"> 1. finish the code 2. UML split 3. Help on analysisso activity diagrams and sequence diagrams 4. Project future 5. Design 	<ol style="list-style-type: none"> 1. 05-20 added last code functionalities 2. 05-21 tested and fixed some bugs 3. 05-22 finished the diagrams 4. 05-23 finished the analysis 5. 05-24 started on design 6. 05-25 design chapter work 7. 05-26 design diagrams 8. 05-27 finishing design chapter 	<ol style="list-style-type: none"> 1. 05-20 2. 05-21- finished implementation part in project report 3. 05-22 4. 05-23 5. 05-24- project report-Design part- architecture and design patterns 6. 05-25 7. 05-26 8. 05-27 	<ol style="list-style-type: none"> 1. 05-20 2. 05-21 Added requirement traceability matrix for testing 3. 05-22 Solved some visual bugs. 4. 05-23 Added table for non functional requirements + grades 5. 05-24 Fixed buggs in Hall of Fame 6. 05-25 7. 05-26 Finished the testing part. 8. 05-27 	<ol style="list-style-type: none"> 1. 05-20 2. 05-21 3. 05-22 4. 05-23 5. 05-24 6. 05-25 7. 05-26 8. 05-27 	<ol style="list-style-type: none"> 1. the code is complete 2. did not split 3. done 4. not done 5. implementation is done
-----------	---	--	--	---	--	--



	Tasks	Daria	Natali	Bogdan	Lukas	Overview
Sprint 13	<ol style="list-style-type: none"> 1. UML split 2. Project future. 3. abstract 4. result and siscusion 5. conclusion 6. sources and information 7. user manual. 8. personal reflection 9. finsih procces report. 10 finish testing 11. checkk if it is ready to upload 12 upload the SEP 13. Add our names to the project report 14. formatting of the files. 15. Monday emergency meeting. 16. SCHEDULE 	<ol style="list-style-type: none"> 1. 05-27 worked on process report 2. 05-28 woked on process report 3. 05-29 finished the process report chapters and personal reflection 4. 05-30 did the client side uml connections 5. 05-31 - did abstract and helped add some finishing touches to the report 6. 06-01 final guidance and formatting help 7. 06-02 8. 06-03 	<ol style="list-style-type: none"> 1. 05-27 - Result and discussions 2. 05-28 3. 05-29- project report formatting + added image/ table names and lists 4. 05-30 - personal reflection 5. 05-31 - user guide + added appendices to project report 6. 06-01 7. 06-02 8. 06-03 	<ol style="list-style-type: none"> 1. 05-27 2. 05-28 Solved some visual bugs. 3. 05-29 Started personal reflection 4. 05-30 Finished personal reflection 5. 05-31 Started conclusion 6. 06-01 Finished conclusion 7. 06-02 8. 06-03 	<ol style="list-style-type: none"> 1. 05-27 2. 05-28 Split the UML's into server and client. In clinet UML view still needs to be finished. In Server UML it should be done will check for sure on 29-th. Uploaded the new astah files to Drive 3. 05-29 UML done and uploaded. Might still ahve errors would be good if it was checked. 4. 05-30 Project Future written and uploaded. 5. 05-31 Wrote the process report intorduction and uploaded. 6. 06-01 7. 06-02 8. 06-03 	<ol style="list-style-type: none"> 1. Project finished



VIA Software Engineering Process Report "Puppr"

APPENDIX C SCHEDULE

PUPPR

VIA University College

