



# Process Report

## JustWork

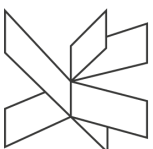
### Group members:

Daniela Garcia Kristiansen (269363)  
Ziad Akram Bathish (273442)  
Maria Louisa Failli (273437)  
Naya Al Tahan (273461)  
Chunhui Liu (273452)

### Supervisors:

Michael Viuff  
Kim Peter Foged  
Joseph Chukwudi Okika  
Jesper Kehlet Bangsholt  
Line Lindhardt Egsgaard

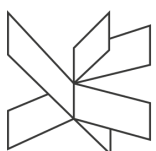
JUNE 2019



## Table of content

1	Introduction	3
2	Group Description	4
3	Project Initiation	6
4	Project Description	8
5	Project Execution	9
6	Personal Reflections	11
7	Supervision	13
8	Conclusions	14

Appendices



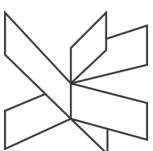
## INTRODUCTION

The following report is divided into different sections to describe our process working on the semester project.

First, there is a group description that introduces each member of the group, continuing with the definition of the project initiation, description and execution, our personal reflection according to the execution and experience working as a group, and finally our observation on the supervision and our final conclusions.

We have been working once a week until the beginning of May with the project description and in the product backlog. Then we started working in sprints dividing tasks meeting once or twice a week. By the end of May we worked daily on the implementation and testing. The day of the deadline was only for revision.

The purpose of the project is to create a system using 3 tier architecture using different languages. Eclipse, Visual Studio, and PgAdmin were used for the implementation and Astah for the Analysis.



## GROUP DESCRIPTION

Daniela Garcia Kristiansen:

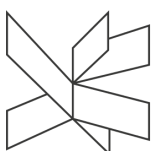
I'm half Spanish and half Danish but I was raised with Spanish education which is very different from Danish. However, I've been studying in Denmark for 3 semesters now, so I'm now used to group work. I enjoy working and learning when people are used to working in a group and appreciate the advantages of it.

Chunhui Liu:

I come from China. I came to Denmark about 1.5 years ago. There is no project of group work in China's education system. After studying in Denmark, I have adapted to and begun to enjoy my study here. I enjoy the way of study that a group work and study together and help each other when someone has problems.

Maria Failli:

I'm half Nigerian, half Italian. I've lived in Denmark for almost two years now. I lived in Nigeria till I was 14 and moved to Italy where I completed my high school which lasted 5 years. The education system was pretty different in both countries but we never really worked in groups, therefore, studying in Denmark where a lot is done in a group was pretty new to me. But after almost 3 semesters I'm quite used to it now and I really enjoy working within a group.

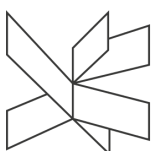


Naya Al Tahan:

I come from Syria, but I was born in Denmark, I lived most of my life in Syria, and I had my primary and secondary education in Syria, so I am used to how it goes there, where we depend on tutors and we never work in teams, but I prefer how education goes like in Denmark, and I'm getting used to it. It's been almost two years since I returned to Denmark and I'm looking forward to finishing my studies and starting my work life in Denmark. Since this the third semester, I have already tried this project work process for two semesters, so I have some prior knowledge about how the process will go like, especially that I have worked with the same group members last semesters as well.

Ziad:

I come from Syria as well but I was born there. I lived in Denmark for about four years. I finished my high school in 2012, and I studied at Damascus University in Economic department for one year. I didn't use to work in a group while we had no group assignments there, and therefore it was really difficult to handle that in the first semester. But now after three semesters it's much better, while we have better knowledge about working process.

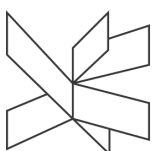
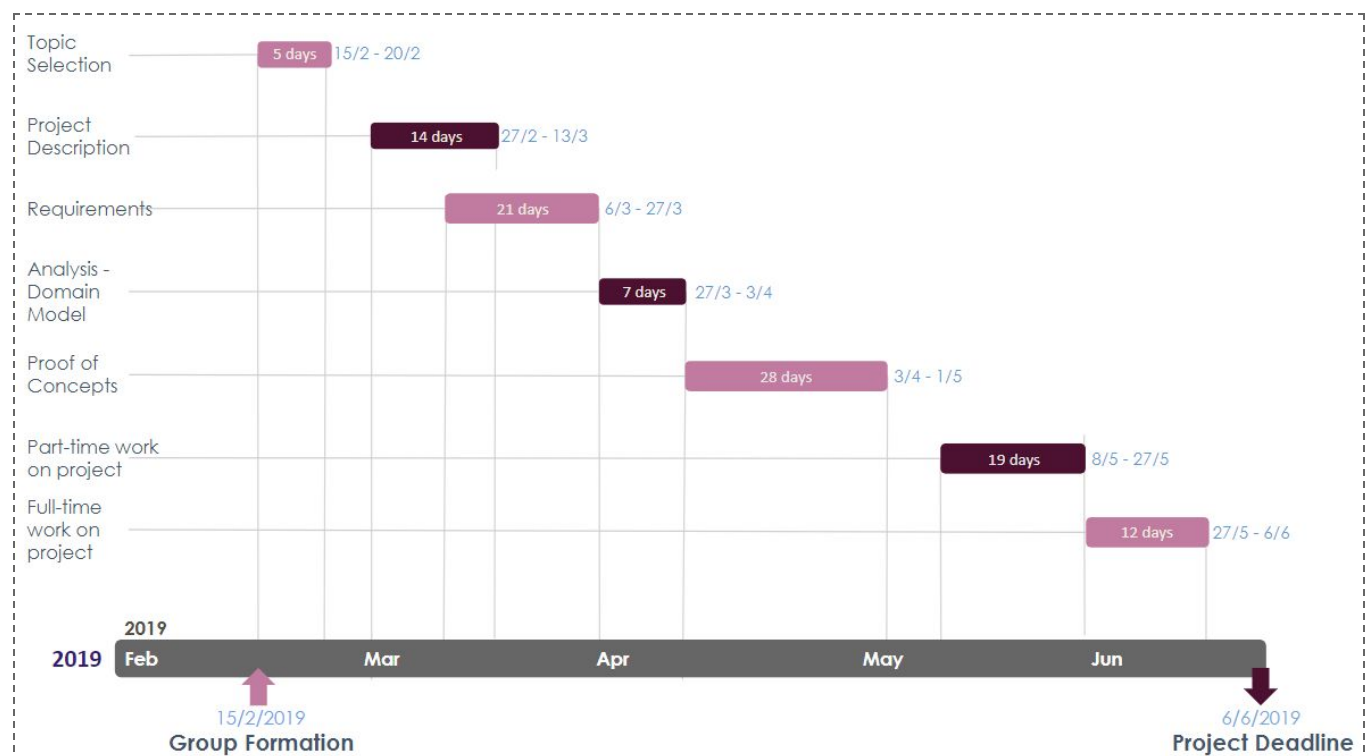


## PROJECT INITIATION

The project was initialized by forming a group, we decided that since we worked well together in previous semesters, we should work together as well this semester. At this point, we had to write a group contract and present a couple of ideas. Establishing the group contract and considering different ideas was an easier process because we knew the way we like to work. In the beginning, we had a couple of ideas: draw and guess game, JustEat clone, the last idea was to re-do last semester's project. After meeting with the supervisor who explained that the drawing game would be a better project for another semester where we don't have to apply the three tiers architecture. Therefore, we came up with a new idea, where it won't be a clone to another already existing system, and also it would suit the fact that the system should be implemented in three tiers.

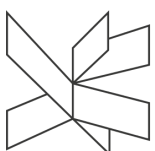
The new idea is a web application that helps match employers with workers- the topic will be discussed more thoroughly in the next section. We found that this topic is really important since it is hard for us to find jobs so we decided to choose this topic.

At this point, we set the main plan of how we will work in the project description and we used gantt chart to show it.



We stuck to the plan that we had made; Starting by writing the project description, the analysis and starting with the scrum process, we had a solid plan that suited us, we estimated how much time we need to finish each milestone, and what time should we meet to check on the progress.

We also used Google keep as a tool to set tasks for each group member and to keep track of what we have to do next, we would update it every group meeting we have.



## PROJECT DESCRIPTION

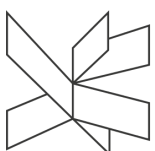
It's common in Denmark that families participate in household chores, but sometimes, it can be too much, so they need to hire someone to do it, but hiring a professional company can be too much since it could be only temporary or even a punctual occasion.

At the same time, there are many students or part-time workers who would like to have an extra income but finding a job can be hard with their strict schedules, also most of the students have no experience which can make it even harder.

The members of the group realised that some people post in some pages as Facebook about this temporary job offers or about their skills but it can be difficult to find a page that is used specifically to find the perfect match and which have some rules and policies with their payments, to try to avoid misunderstandings or scams.

We defined the previous problem by feeling related and affected by it, but since the semester courses don't include how to use and implement payment systems, and also we wanted the users to be able to contact each other using our system by a chatting system functionality, where it was hard to implement considering the requirements of the project (use web service).

What we wanted to achieve is a safe system that helps the users hire and be hired by other users for basic services, and we could achieve that.





## PROJECT EXECUTION

We used Scrum agile to execute the project and we used it by the following plan:

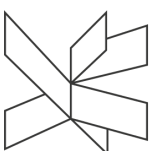
The project execution started by analyzing the project, creating all the diagrams to help us understand the domain of the project. After that, we wrote down the user stories of the project, then we started declaring the different sprints that we will go through and assigning some user stories to each sprint and then for each sprint we design, implement, and test it, to make sure we have a full functioning product by the end of the sprint, then we declare the results of the sprint and decide if we need to make any changes to our plan.

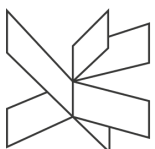
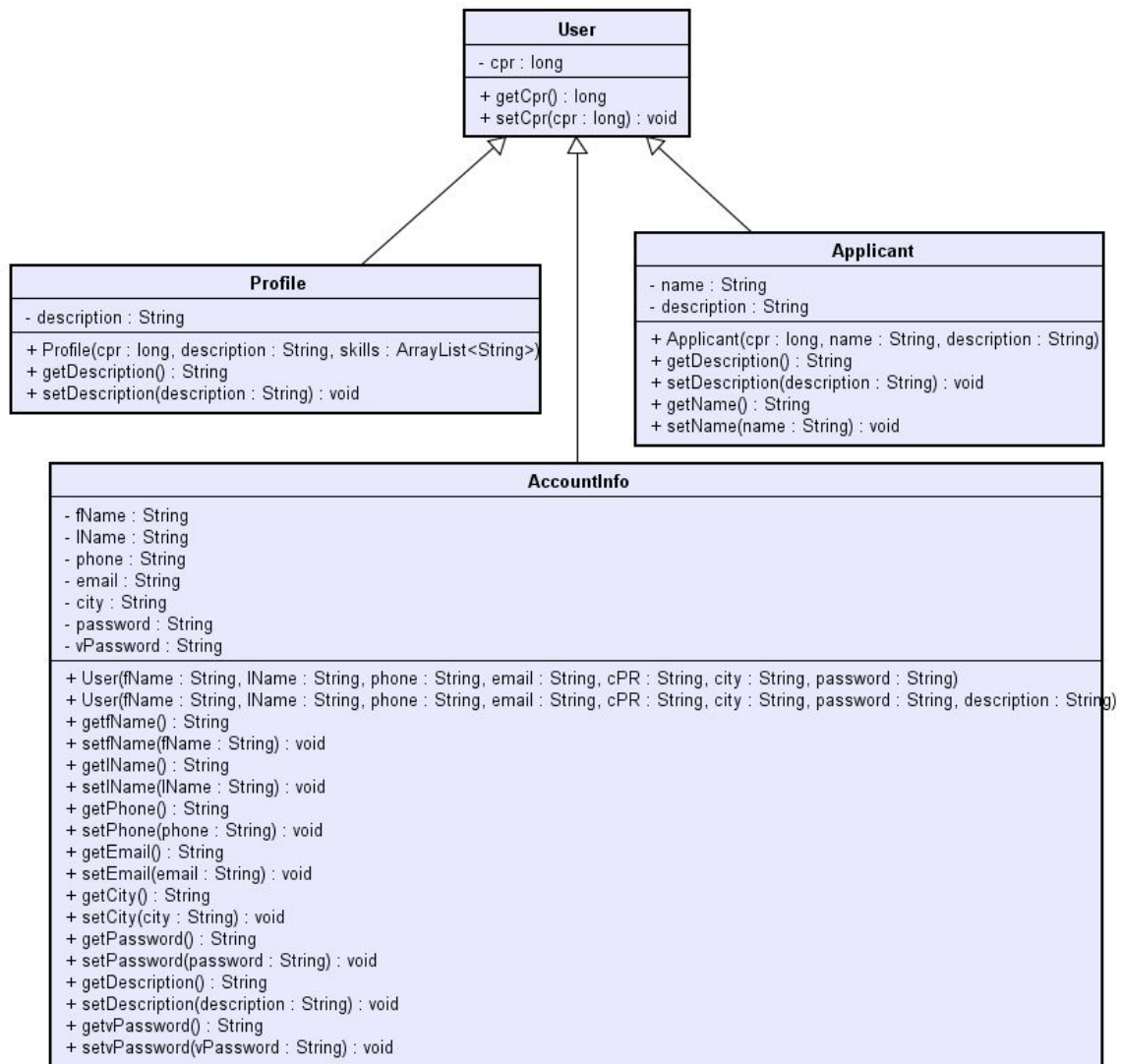
That is how the project execution went like, we had some difficulties during some sprints, for example making a chat system using web services seemed impossible, since we couldn't figure out a way where the web service will be able to invoke methods on tier1 since it doesn't keep track of the online users.

We knew that the chatting system could be hard to implement so before starting working in sprints we did some research to see how we can make it work using web services to stick to the requirements of the semester project

Therefore, when we got to the chatting system sprint and we only had some days left for the project, we decided to stop implementation and start writing the reports.

Since we were working in sprints, it was hard to make the design of the final product, so there were some design mistakes in the model classes and how they related to each other, So the model classes should have been structured like this :



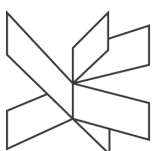


We also didn't set values to each user story, which made it hard to keep track of how much is left to do after each sprint, but after the project execution finished we fixed that by assigning points to each sprint and estimated work time for each sprint, the estimated time and actual time is how much time each member of the group spent on working on the sprint. We were able to do this at the end because we have a book log where we've always written about what we've done each time we all met up to work together this helped us figuring out how much time we would need to finish all the user stories.

Main				
sprint Id	Sprint Title	Points(1-10)	Estimated Time (Hours)	Actual Time (Hours)
1	Start-Up	15	15	22.5
2	Sign In and Log In	10	10	20
3	Create Service Request and Service Offer	20	20	10
4	View Service Request and Service Offer	10	10	5
5	View Logged -In User's Service Request and Service Offer	20	20	5
6	Apply and Unapply for Service Offer	25	25	17.5
7	View and Edit Logged-In User's Profile	20	20	15
8	Delete logged-In user's service offer and request post	15	15	12.5
9	View list of applicants to service offer post	20	20	20
10	Contact users	10	10	20
11	Employ User and Pay	20	20	5
Total		185	185	152.5

At this point, we had to make changes to the analysis diagrams such as the domain model, to make it only cover what is implemented.

The final product meets and solves the problem we were trying to solve, which is helping users finding jobs and workers.



## PERSONAL REFLECTIONS

Maria Failli:

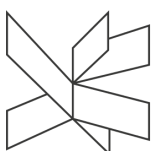
Since the first semester we've all been members of this group, except for Ziad who joined in the beginning of the 2nd semester. We have become quite familiar with each other and we've created a fun and friendly atmosphere where we are able to work in a way that we all try to achieve the best that we can and we have fun doing that. This has allowed us to learn a lot about each other's strengths and weaknesses and we try to help each other out whenever necessary and also use each other's strengths to the advantage of the project. Working in a group sure has its ups and downs, sometimes there can be miscommunication between the members: when someone has an idea and tries to explain it to the team members and each one of us interpretes it in a different manner, this isn't necessarily a bad thing because in the end it allows us to use all the tools we have in order to really understand the original concept such as using class diagrams or sequence diagrams. In a way it pushes us to understand each other better and to use the tools and guidelines we've learnt since the beginning of this course. It will also make us good team workers when we get into the real world and start working.

Daniela:

This group has been working together since the first semester, so we have become familiar to each other and the working atmosphere is very good. We tried to keep the same friendly environment and it made it easier with the communication between us, and it also makes the working time more fun. At the beginning we were not sure about the project idea so we decided to change it to keep us more motivated when working on it. I think we were more organized that last semester and we learnt from our mistakes, however we still could have worked better with Scrum. For the next project I will try to document better each sprint and try to do better with the time estimation because we couldn't finish implementing all the use cases that were planned.

Naya:

I like to work with this group since I'm familiar to its members and their skills, like what they are good at, so it was easy to assign work to each member. We tend to work together and try to help each other all the time. However, I find it hard to work in a group since it could be hard to keep track of changes we each do to diagrams and the code, so what I liked the most was



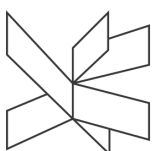
that we could work on tier1 and tier2 on separate laptops, which helped the group work. Also, designing each sprint alone made it hard for me to get the rightest model classes, but the mistakes are not crucial, but it could be designed better (tier 2.model).

Liu:

We still cooperate well this semester. The learning atmosphere is very good. Everyone is friendly and helpful. However, our mistake is that we did not consider the difficulty of the goals we set carefully, so we did not leave enough time to finish the report, so we were very busy in the past two weeks. But everyone worked hard, so we could still finish the report in time. Next time we will carefully consider the feasibility of the goals we set and spare enough time to finish the report. I can say that I have learned a lot by myself and from my group members that helped each other if something went wrong. I gained a lot of experience and skills that will help me to grow as a person and as a software engineer.

Ziad:

I feel really comfortable with working in this group, while I joined them last semester but I feel we could understand each other quickly. I like the humour they have while they are working which create a better environment to work and finish tasks. while 3 tier architecture was a bit difficult to understand in the beginning of the semester, but it helped us to work more separately. last semester we worked with scrum, but many things went wrong while it was definitely new for us. this semester we decided to use scrum again to learn it while many companies in the world use it now. I think we made a mistake with requirements, we add many of them where we don't have enough time to make everything working. Each semester we learn new things and have a better knowledge of avoiding mistakes which can stop our working process.



## SUPERVISION

We are satisfied with the supervision of the semester project. However, we also thought that it'd have been very useful if the DNP teacher was one of the supervisors because his subject is one of the main parts of the semester project.

In general, whenever we feel unsure about a project-related issue, we would contact the responsible supervisor that knows best how we can solve the issue, and request a meeting. We mostly prepared the meeting by setting the questions we have for the supervisors prior to the meeting.

However, we would have preferred if the supervisors were clearer with what they are expecting, like if they set a list of diagrams they are expecting, and also to mention how much we should document our scrum process.

During the first sprint, we were unsure about how to connect tier2 with tier1, and how to implement it. Therefore we asked the supervisor Joseph to help us.

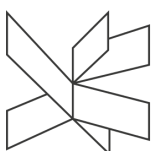
Later on, we had some issues regarding separating the tasks of the three tiers and we also asked the supervisor Joseph and he made it clear to us how to separate them.

Every now and then, we would request a meeting regarding some small questions, such as report questions, and most of them were successful.

We just wish the requirements of the project were more fulfilling and cleared out, since we are in a learning stage, it would be nice for us to know how to perfectly work on a project instead of trying to figure it out ourselves by making mistakes and learn from it.

Just like how the security supervisor set it straight what he's expecting from us, if all the supervisors did something similar it would be easier to learn how to make a fine project.

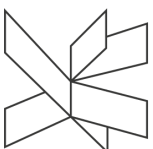
Even just one example of a properly done and documented project of a previous sep3 group so we can learn how to reach the same level.



## CONCLUSIONS

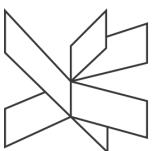
The project gave us a lot of experience in documentation, and programming, especially using .NET Framework. Also, the project had a huge influence on our personal development. For instance, we became better team workers and every time we work as a team, we realize that experience like this will be very helpful after university when we start working as engineers in the teams.

It has been a nice process working as a group and though it had its ups and downs, but we are satisfied with the results and we felt like we learnt a lot from our mistakes and we are looking forward for next semester to apply what we learnt from our mistakes into the process of semester project 4.



## Appendices

1. Grouper-Contract
2. LogBook
3. Sprint documentation







# Project Report

## JustWork

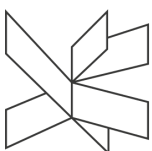
**Group members:**

Daniela Garcia Kristiansen (269363)  
Ziad Akram Bathish (273442)  
Maria Louisa Failli (273437)  
Naya Al Tahan (273461)  
Chunhui Liu (273452)

**Supervisors:**

Michael Viuff  
Kim Peter Foged  
Joseph Chukwudi Okika  
Jesper Kehlet Bangsholt  
Line Lindhardt Egsgaard

3<sup>RD</sup> SEMESTER  
JUNE 2019



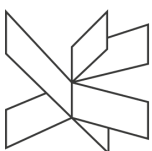
VIA University  
College

## Table of content

Abstract	3
Introduction	4
Requirements	5
<b>2.1 NON FUNCTIONAL REQUIREMENTS</b>	<b>6</b>
Analysis	7
Design	10
Implementation	13
Test	18
Results and Discussion	20
Conclusions	25
Project future	26
Sources of information	27
Appendices	28

## List of figures and tables

Optional



## Abstract

JustWork systems is developed to help people in Denmark to provide and find job easily. The main purpose of the system is to simplify and facilitate people's lives with computerized and well-organized system.

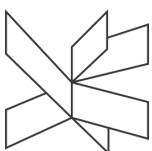
Clients can access the system using the secured web application which has an interface for users that are accessible by a login system.

System is based on 3-tier architecture and uses web API as presentation layer for the system. users can use a browser to use the system.

Whereas the application layer is implemented in java and connected to the presentation layer by JAX-WS

The data layer is a database implemented using PostgreSQL and is connected to the second tier using JDBC.

The end-product provides for the user most of the requirements of the project, and most of the testing went positive



# 1 Introduction

<sup>1</sup>In most Danish families, each member of the family participates in household chores such as cooking, gardening, doing the laundry and house cleaning. So there is generally no need for a domestic worker. But sometimes each family member has got a handful to do in their daily lives outside of home, for example: the older kids in the family might have a big exam or internship so they don't do much at home, if the chores at home become too much for the other family members to handle they might consider having a temporary domestic help. It could also be that the family is planning a big event or they need to do something where no member of the family has the skills for it.

On the other hand, there are students and some part-time employees who have some spare time and would like to earn a little extra income. However, it can be hard to find a formal part-time job that suits their time. Because the job would have a strict schedule that it would make it hard to find one with fixed hours.

While it is already an option to post about your skills or job opportunities on Facebook, it is quite difficult to find a match with what one is looking for and there are no rules or policies for the payments, the two parties would have to agree together, this presents a risk of one not getting paid.

The company members have noticed some posts on Facebook and other websites (figure 1 & figure 2) where people announce that they need help with moving stuff, organizing a small party and cleaning up afterwards. Just like we've noticed people post about their skills and qualifications that they'd like to offer.

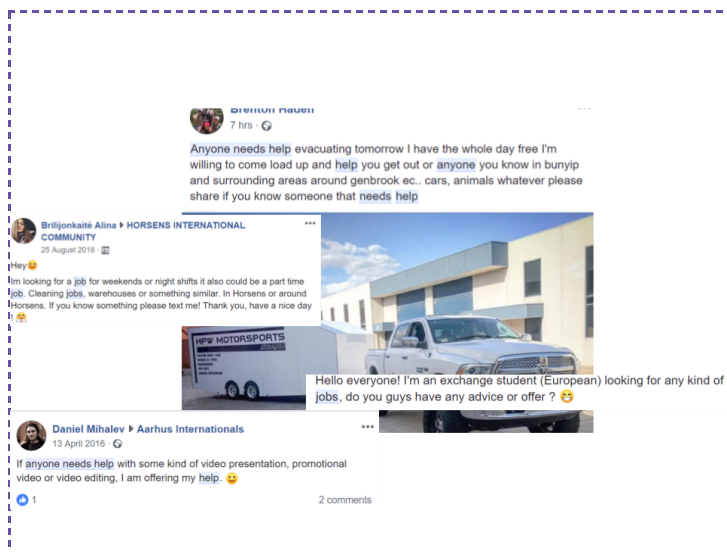


figure 1

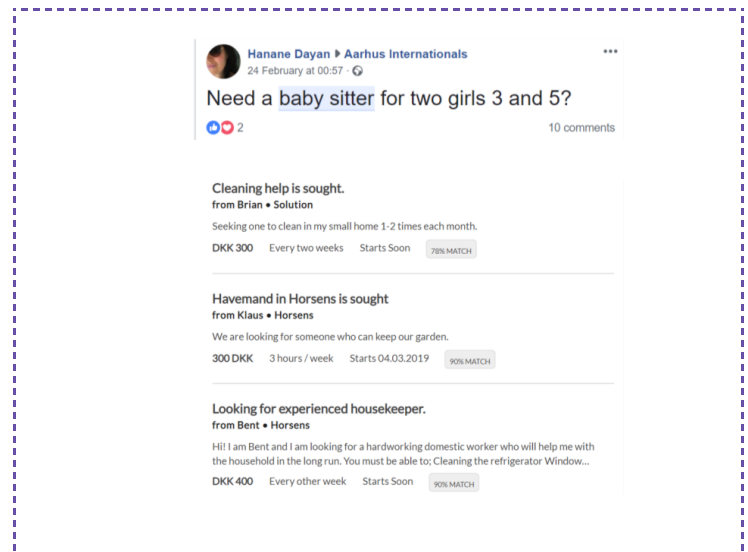
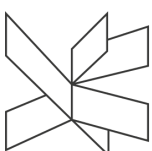


figure 2

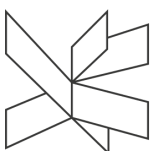


## 2 Requirements

The system will have an actors which is user, and the user can be employer or worker, depends on chosen services from the system.

Throughout the project report and process report, we will refer to the employer and worker together as users. And the administrator will be the group who made the system.

1. When a user first signs up, they will be asked about their name, phone, email, CPR, city, and they get to choose a password.
2. Users will login using their CPR and password
3. The worker will be able to post their job requirements by filling out a form where they write the job title, choose the category of the job, and a description.
4. The employer will be able to post their job offers by filling out a form where they mention: job title, job category, time, description and address.
5. Workers will be able to apply for different jobs
6. The employers will be able to see the list of applicants and their profile(name, contact information) that applied for his/her job offer.
7. Employers will be able to choose a worker from a list of all workers that applied
8. The user will be able to create a profile with their description and skills
9. The user will be able to edit their profile information ( phone, email, city, and password).
10. Users will be able to see all the job offers close to their area
11. Workers and Employers will be able to communicate with each other using a chatting system
12. Employers will pay the money once they choose a worker
13. The worker will receive the money once the job is done
14. Old and outdated posts will not be shown in the list of offers/requests.
15. The administrator will delete inappropriate(such as offensive language use) posts and users, and irrelevant posts(such as a cleaning job in gardening category).
16. The website will have contact information of the administrators.
17. The users will be able to contact the administrators by some live chat functionality.



## 2.1 NON FUNCTIONAL REQUIREMENTS

1. The application will be only available in English.
2. The system will be a responsive website for user.
3. The system will be a management system for the administrators where they can access the data.
4. The system must support Microsoft Internet Explorer 9,10 and 11, Google Chrome 28-47, Mozilla Firefox 32 and Microsoft Edge 20.
5. The system needs to answer within 2 seconds 95% of the time.
6. The system must be usability tested by end-users.

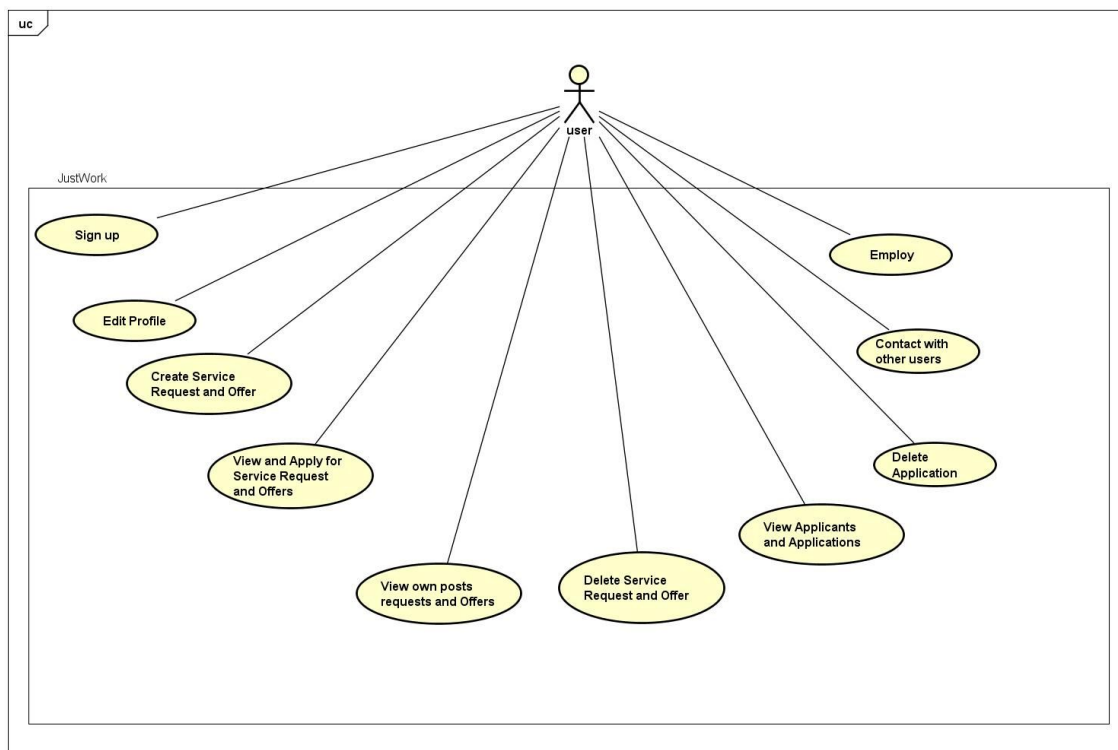


### 3 Analysis

After reading the requirements, it was possible to start with use case diagram to define common user goals.

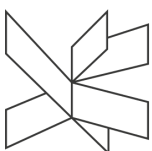
#### a. Use case Diagram

The Analysis part was started by discussing and making use case diagram to illustrate what the user can do while using the system, where the user must be able to sign up, log in and use the system's features as posting service requests or even offers. And the user is able to edit posts or delete them when needed. the user can also apply for requests from other users or see applicants who applied to the user's requests.



#### b. Use Case Description:

the attached use case descriptions show how the user can perform tasks on this system. for each use case, a use case description has been made with a summary where a brief explanation of the use case was written, pre and post conditions when the user needs to perform a task, base sequence and finally exception sequence. Two examples here of use case descriptions of the system are shown, the first one is "Create Service request and offer " where the user must be logged in before being able to post any type of service requests or offers. The steps of posting are described in the Base Sequence, the user decides first what to post a request or an offer, then clicks the wanted service, fills a form to describe the posted service with details, and finally clicks "create" button to submit posting.



The second one is "View and Apply for request service and offers "

a precondition is to be logged in, and what steps needed to view and apply for offers or requests.

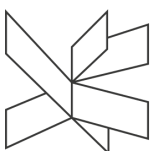
ITEM	VALUE	ITEM	VALUE
UseCase	Create Service Request and Offer	UseCase	View and Apply for Service Request and Offers
Summary	user is able to post a service request or service offer	Summary	user is able to see other's posts and apply to offers
Actor	user	Actor	user
Precondition	user logged in	Precondition	user logged in
Postcondition	the post will be shown up to other users	Postcondition	user applied an offer
Base Sequence	Case A: Create Service request 1- click: "create service request" from the home page 2- fill the form with the title and description of the request 3- click: "create" button Case B: create Service offer 1- click: "create service offer" from the home page 2- fill the form with the title, description, address, time and salary of the request 3- click: "create" button	Base Sequence	1- click: "JustWork!" 2- home page will show up 3- all offers and request from other users shown to the user 4- click: on wanted offer 5. Click: on 'Apply for Job' button
Branch Sequence		Branch Sequence	
Exception Sequence	1- a field is empty	Exception Sequence	none
Sub UseCase		Sub UseCase	
Note		Note	

#### c. System Sequence Diagram:

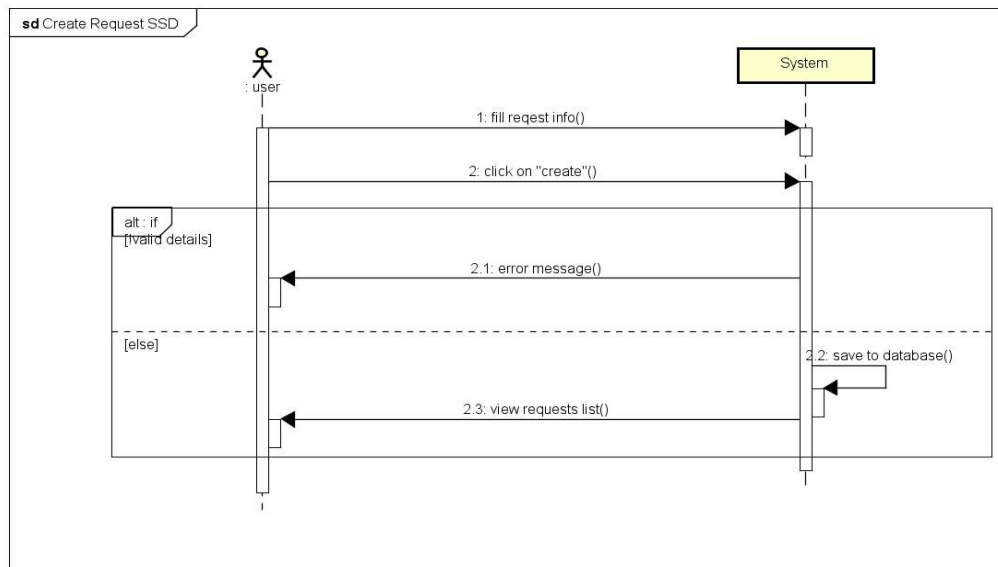
Using the use case description, the System Sequence Diagram has been made to have a better understanding of the system and its interaction with users. Here are two SSDs as examples:

##### - Create Request:

the user should fill some information as Service title, choosing which category is it and a description about the request, then other users have better knowledge of what the user can do. After clicking "create" the information about this new request will be saved on the database and the system will react by showing a view of user's offer or requests if there are no errors or empty fields, otherwise the system will react with an error message.

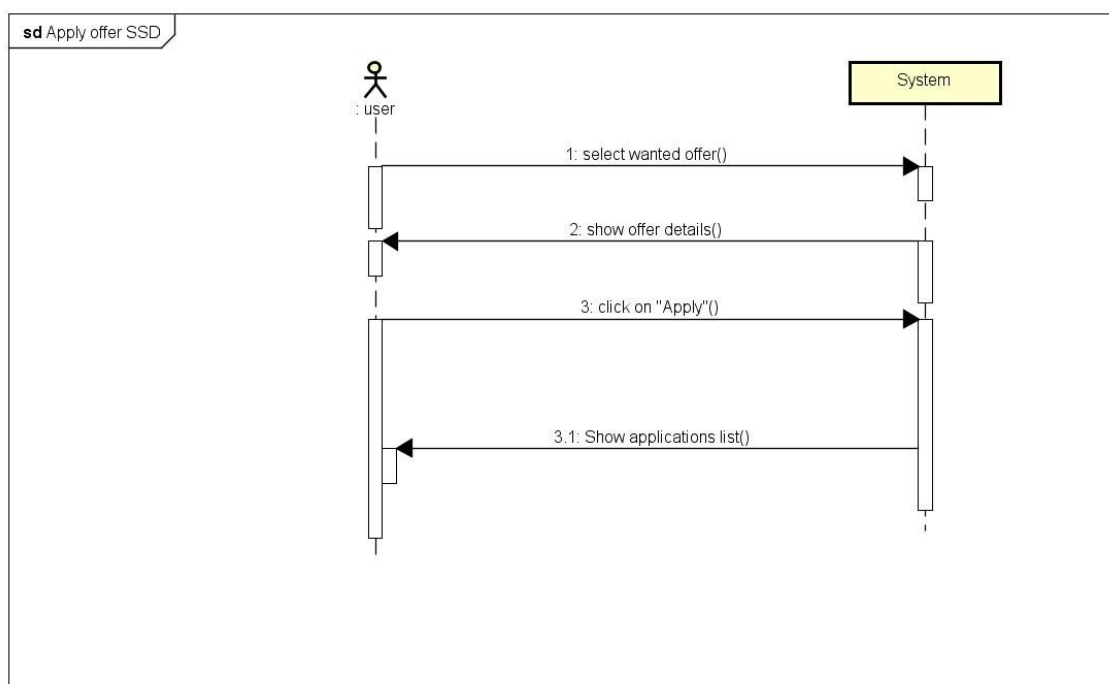






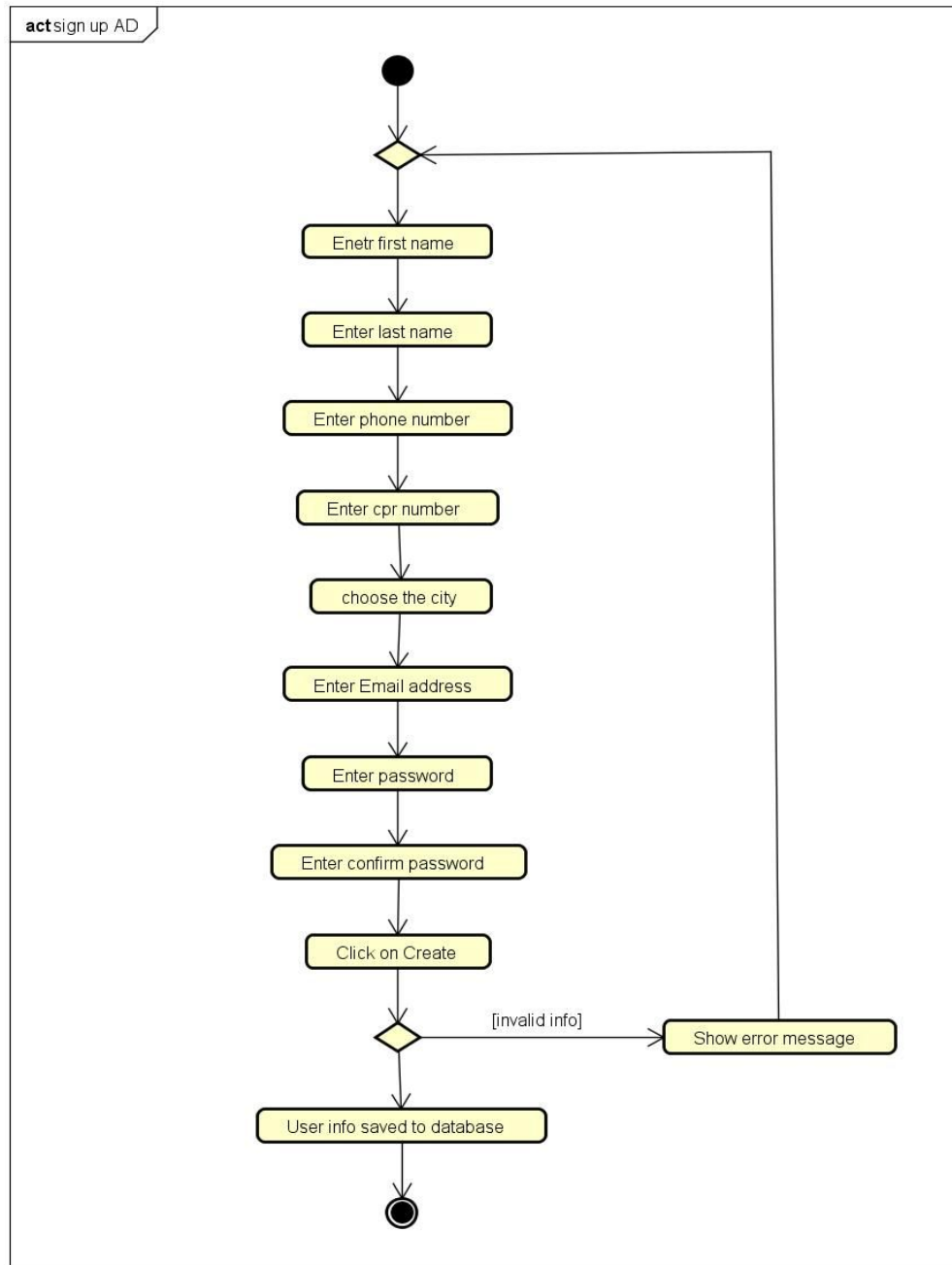
- Apply for offers:

while the user can create requests and offers, user will be also able to see other users' posts and apply for it. In the system, a list of requests and offers from others will be shown on the user's Home page. The user can click on offer post and click apply afterwards to be as an applicant, and the System will show the Home page.

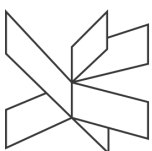
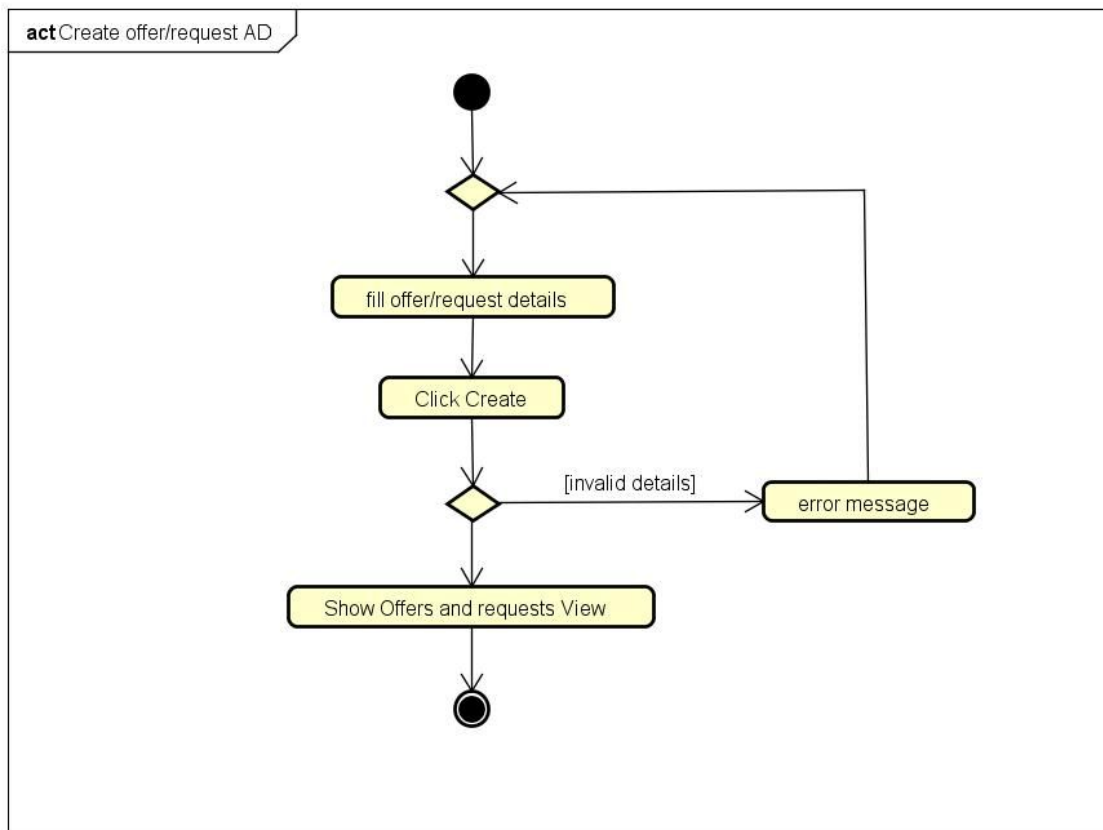


d. Activity Diagram:

The following Activity Diagram shows the steps of how the user can sign up, and the exception that happened if any info is wrong as date form or an empty field.

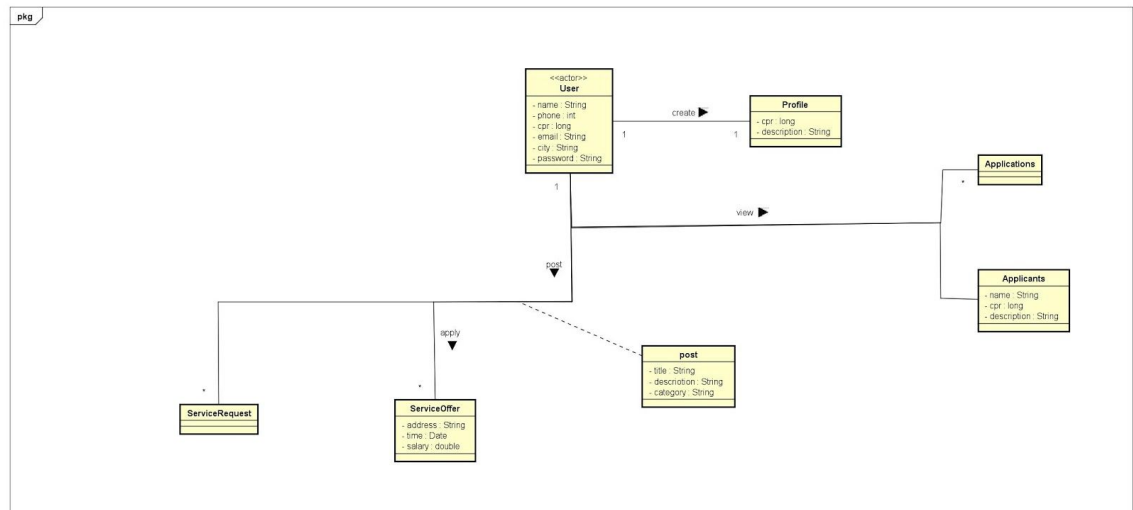


and another example about how user can create a service request or service offer, and how the system is going to interact after clicking “create” button



#### e. Domain Model:

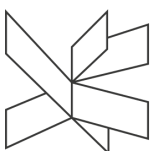
Here are the important problem domain concepts as classes and attributes and the relationships between them. Where the actor is the user, can create only one profile, the user can post many service requests or service offers and see all application or applicants related to what the user posted or applied.



#### f. Security Analysis

##### Threat model

Threat	Vulnerability	Asset and consequences	Risk	Solution
User system passwords can be stolen	Phishing or brute force attack	Unauthorized access	Very high	Require min length, for the password and password hashing
Payment system information can be stolen	Phishing or brute force attack	Sensitive data could be misused and causing a loss of capital	Very high	Data encryption and input validation
Modification of data inside the database	SQL injection	Sensitive data could be misused	Very high	Data encryption and input validation
Cross-site scripting	Injection of malicious code	Disclosure of sensitive data	Very high	Escaping, validating and sanitizing the data
Man in the middle attack	Unprotected network	Access to the transmitted data	Very high	Use HTTPS protocol



### Security objectives:

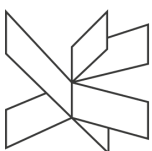
1. Authentication: Verification of the user. What are the permissions of the user based on their authorization levels.
2. Authorization: Assurance that the user has the right to access the system and perform activities in it.
3. Data integrity: Protection of data from unauthorized access and changes.
4. Data confidentiality: Private or confidential information is not made available or disclosed to unauthorized individuals.
5. Availability: Assurance of reliable access to, and use of, information.

### Risk assessment model

Threat event	Threat source	Relevant	Likelihood of attack intention	Severity	Likelihood of attack to succeed	Level of impact	Risk	Risk score
User system passwords can be stolen	Hackers	Possible	High	High	High	High	High	10
Modification of data inside the database	Hackers	Possible	Moderate	High	High	High	High	10
Cross-site scripting	Hackers	Possible	Moderate	High	High	High	High	10
Man in the middle attack	Hackers	Possible	Moderate	High	High	High	High	10

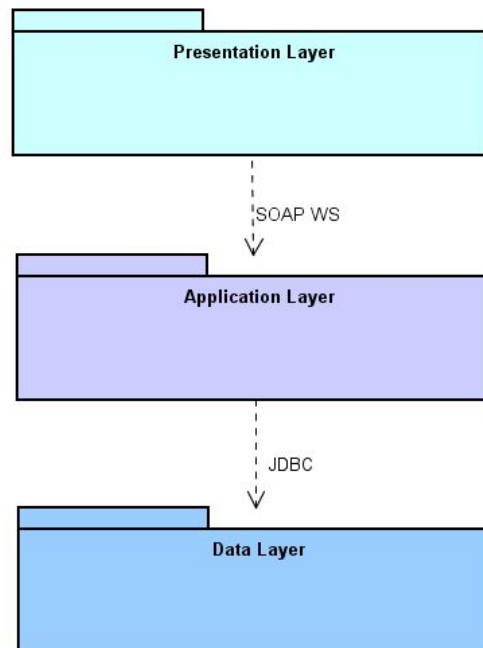
### Discussion

JustWork application contains a sensitive data of users such as login information or CPR-NUMBER which are prone to misuse. That's why it is necessary to find all possible threats to prevent them and fulfill the security objectives which are stated in the Threat model. The possibility and overall risk for each threat from Threat model is shown in the Risk assessment model. The solutions for these are described in the design part of the project report.



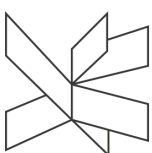
## 4 Design

The architectural pattern used to design the system is three-tier architecture, due to the fact that the system is based on providing the end-user the ability to interact with data stored at some server.



The three-tier architecture perfectly suits the system, where it divides the system into three layers:

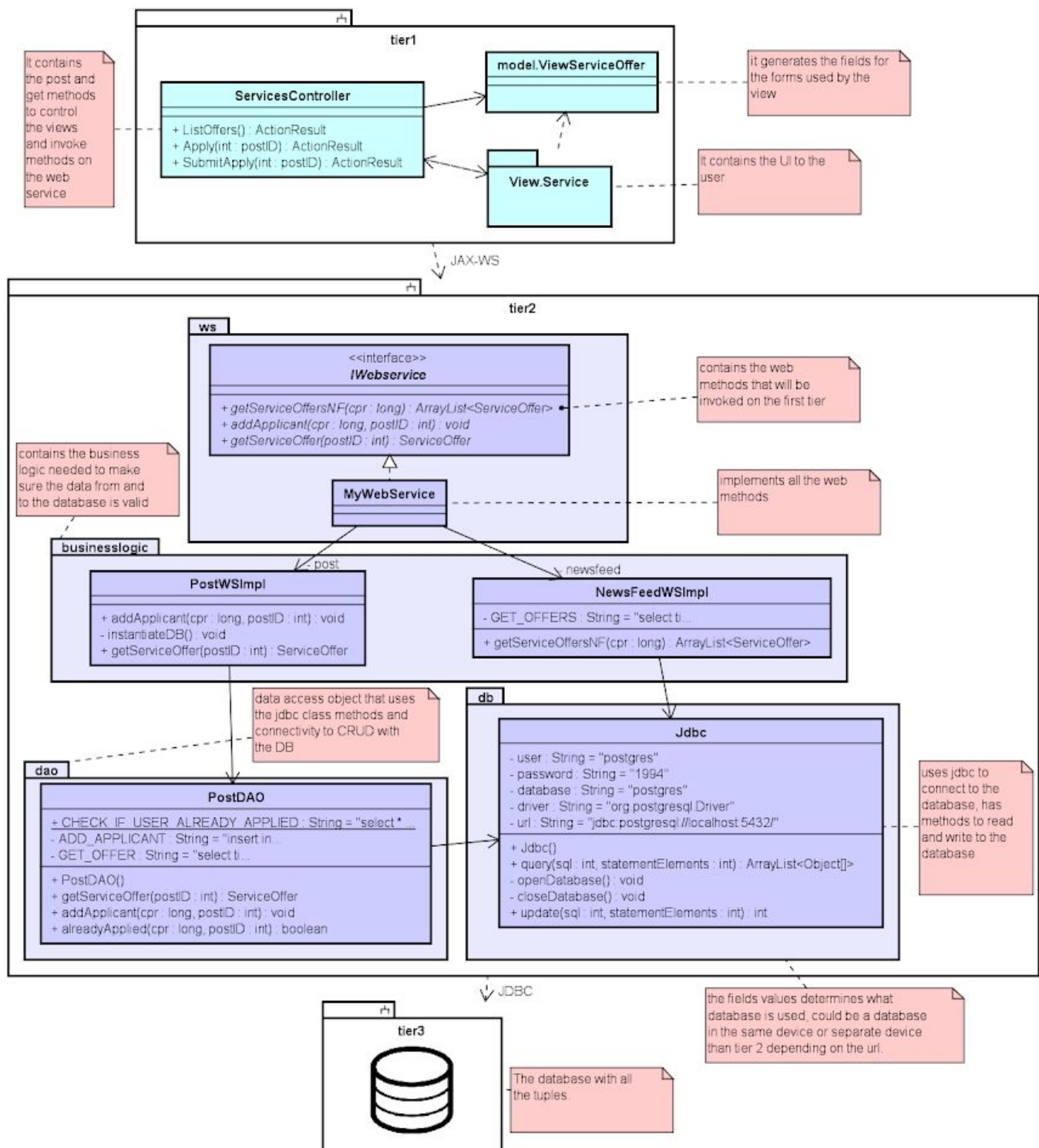
1. Presentation layer: the design pattern used to design this layer is MVC, where the controller would get info from the second tier, create a model object and send it to the view to show it to the end-user. The controllers contain all the action listeners of different view components.
2. Application layer: handles the business logic by checking the validation of forms being posted from the presentation layer, throwing specific exceptions depending on what went wrong, exchanging data with the third tier by translating the data that it gets from the database into the model objects, and the other way around.
3. Data layer: stores the data that is used by the different tiers in a database.



The first layer is connected to the second layer through SOAP Web services, where the first tier invokes web services methods on the second tier providing the ability to pass data through SOAP messages.

The second tier is connected to the third layer through Java Database Connectivity using a class that provides the second tier with methods to manipulate the data in the database.

The following class diagram shows how the classes included in view service offers and apply use cases are structured:



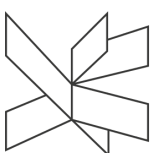
The 1st Tier was designed using asp.net MVC pattern. Using this pattern, requests are routed to a Controller which is responsible for working with the Model to perform actions and/or retrieve data. The Controller chooses the View to display, provides it with the Model and invoke methods on the web service. The View renders the final page, based on the data in the Model.

Since the 1st Tier does not have any business logic, The model was used for generating fields for forms, when the user fills out a form, the View simply displays the forms and the Controller takes the data and passes them to the Model in the 2nd Tier which is where the business logic all lies.

The following section is describing the sequence diagram of viewing service offers and applying for one. The diagram can be found in appendix B (Diagrams/DesignDiagrams/Show Offers and apply.svg)

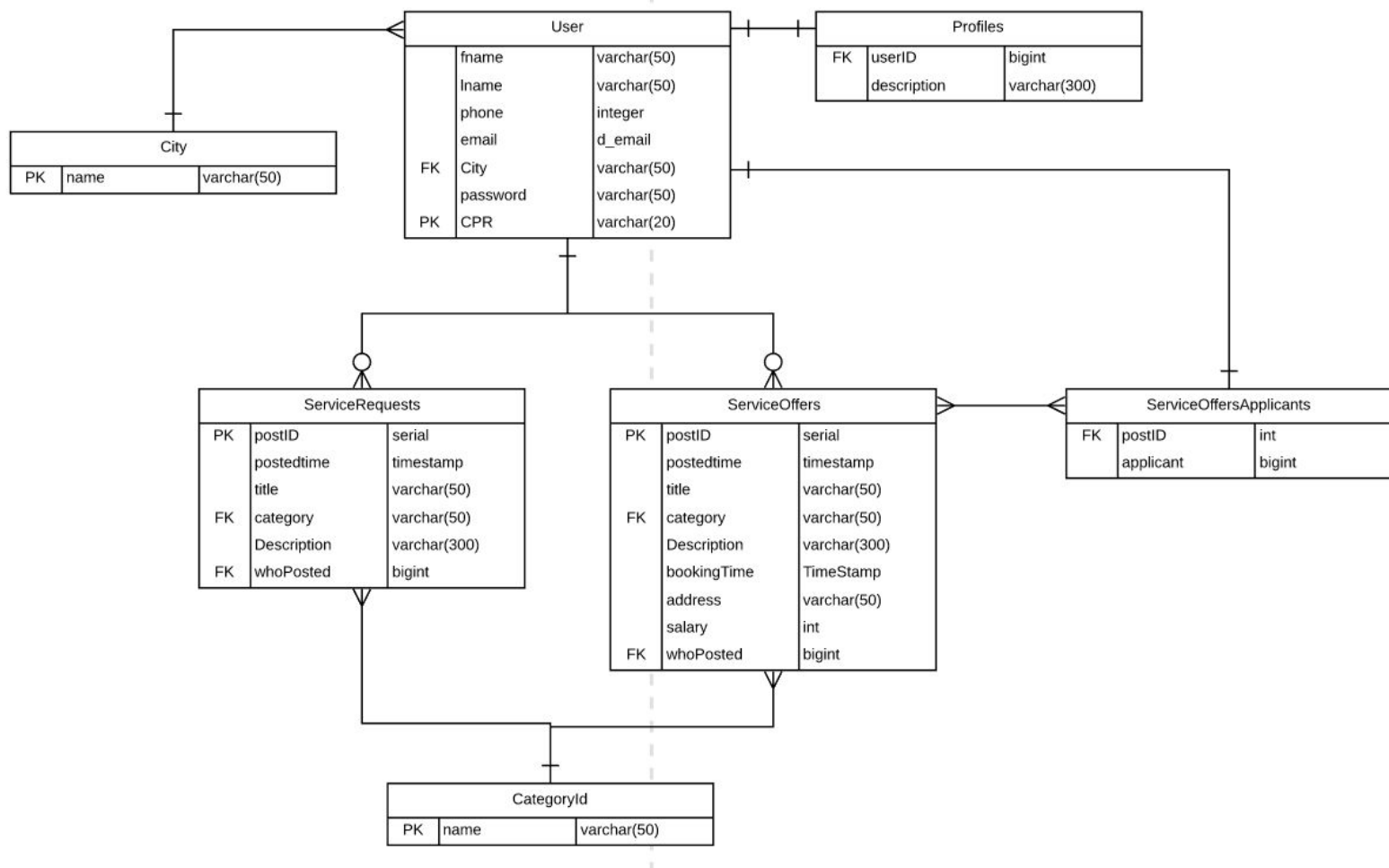
When the user chooses to view offers list, the ListOffers() method on the controller will be used, which uses the web service to invoke the getServiceOfferNF(cpr) that returns the ArrayList<ServiceOffer> list object, and uses the user's cpr number as a parameter. This method is implemented in the MyWebService class, which will get the list by using the getServiceOffersNF(cpr) on the NewsFeedImpl class, that will use the query method on the Jdbc class, that takes the GET\_OFFERS string which is the sql statement that will get a table of type ArrayList<Object[]> that holds the offers retrieved from the database, each row in the table represents an offer, the NewsFeedImpl class will have to translate each row into a Service Offer Object and add it to the list that it will return. If the table was empty, an exception which has a message of "The list is empty" will be thrown to the first tier and it will be handled there, otherwise, the controller in the first tier will return a View containing the list and show it to the user.

Now the user chooses to apply to a certain offer with the postID (postid), they will be redirected to another page using the Apply(postid) on the controller, which return the other page that there the user has to click apply button. The controller will call the method SubmitApply(postid) that invokes the addApplicant method on the web service, it takes the user cpr and the post id of the offer that he chose as parameters, then the implementation of the webservice calls the addApplicant method on the PostWSImpl, that contains the business logic of the posts, which includes checking if the user has already applied to that offer before adding the applicant to the database, which is done using the alreadyApplied method, so it checks the third tier's tuple (ServiceOffersApplicants), using the PostDAO, if it contains a row with the same cpr and post id, if there is, the query will return a table with it, where the alreadyApplied method will see if the table contains anything or not, if alreadyApplied returned true, the user will not apply again, and nothing will happen, if it returned false, it will use the addApplicant method on the dao to update the tuple.





The design of third tier was done mostly by upgrading the logical ER diagram into a physical one that can be implemented in PostgreSQL later on in the implementation phase.

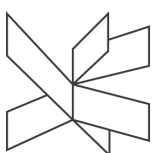


This diagram sets all the tuples needed to implement the third tier and the relationships between them and all primary keys and foreign keys as well.

The ServiceOffersApplicants tuple is what makes it possible to implement the relationship between the ServiceOffers and the User tuples.

The city table will have the different counties in Denmark that the user can choose between to set as where they are staying.

The CategoryId table will also have a list of categories of the different service offers and requests.



## Security mechanisms

Security mechanisms to prevent threats :

1. User system passwords can be stolen – To prevent a Brute force attack there is a minimal required length of 8 characters.

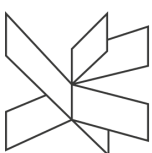
Other security mechanism to avoid password being stolen is hashing the password string value.

2. Modification of data inside the database – Possible solution to prevent SQL injection is the encryption of confidential data and connection strings. Other mechanism is input validation to prevent improperly formed data from entering database

3. Cross-site scripting (XSS) – XSS can be prevented by escaping, validating input and sanitizing.

- **Escaping:** By escaping user input the key characters of the received data by a web page will be prevented from being interpreted in malicious way
- **Validating input:** Application is only rendering the correct formed data and prevents malicious data from doing harm to the site, database and users.
- **Sanitizing:** Checking user input before storing it in a database.

4. Man in the middle attack (MITM) – MITM can be prevented by using SSL (Secure Sockets Layer) which provides an encrypted link between a web server and a browser so the data passed remain private and is not vulnerable to MITM.



## 5 Implementation

As mentioned earlier, the 1st Tier was designed using asp.net MVC pattern. This allowed us to easily separate the presentation layer from the data application layer. The Model was used to generate fields for forms and it contained no business logic, the View displayed the fields contained in the Model.

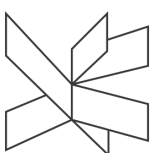
The first tier is connected to the second tier by having a service reference to the JAX web service. Therefore, it makes it possible to have a proxy of the web service, that lets the first tier invoke methods on the second tier, and pass info from and to the second tier through these web methods.

The setup of the second tier included injecting the maven project with some injections, such as JAX-WS injection, JUnit injection and PostgreSQL injection, which were a part of the process of connecting tier 2 with tier 1 and 3.

Also, a xml file was created in the deployed resources in the maven project, that contains the web service endpoint declaration.

The interface that holds the web methods of the web service is class IWebService and MyWebService class implements that interface and will call methods on data accessing objects and business logic classes as will be shown in the example in the next page.

To connect to the third tier, a class that uses JDBC to connect to a database using PostgreSQL JDBC driver, this class is called Jdbc, it uses classes from java.sql to create statements, get Connection to the certain database used, using the url, username, and password, and to get result sets from the database and turn it into an ArrayList of Object arrays.



We'll look at how we display the list of service offers and how a user applies for the offer.

In order to implement this, we first need to get the list of all the service offers available in the user's county. The `ListOffers()` method is called (figure 5.01), in here we invoke a web service method `ws.getServiceOffersNF(long cpr)` which is in the 2nd Tier (figure 5.02).

```
public ActionResult ListOffers()
{
    try
    {
        offers = ws.getServiceOffersNF(long.Parse(Session["user"].ToString())).ToList();
        return View(offers);
    }
    catch (System.ServiceModel.FaultException e)
    {
        ViewBag.ErrorMessage = e.Message;
        return View("Error");
    }
}
```

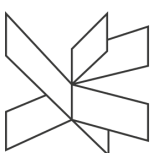
figure 5.01

The `getServiceOffersNF(long cpr)`, which is implemented in `MyWebService` class, returns an arraylist of `ServiceOffer` from the database.

```
@Override
public ArrayList<ServiceOffer> getServiceOffersNF(long cpr) throws SQLException, EmptyListException {
    return newsfeed.getServiceOffersNF(cpr);
}
```

figure 5.02

The `getServiceOffersNF(long cpr)` method throws an exception in case the list is empty. The exception is caught on the 1st Tier, if the list is empty, meaning there are no offers in the user's county then we return a View of an error page. If it's not empty, the controller returns a view containing the list of service offers.



The next figure (figure 5.03) shows how the NewsFeed class gets the ArrayList of ServiceOffers from the database.

```

public ArrayList<ServiceOffer> getServiceOffersNF(long cpr) throws SQLException, EmptyListException {
    if (db == null) {
        db = new Jdbc();
    }

    ArrayList<Object[]> table = db.query(GET_OFFERS, cpr, cpr);
    if(table.size() == 0) {
        throw new EmptyListException("There are no service offers in your county");
    }
    ArrayList<ServiceOffer> list = new ArrayList<ServiceOffer>();

    for (int i = 0; i < table.size(); i++) {
        Object[] row = table.get(i);
        // String jobTitle, String jobCategory, Date bookingTime, String
        // whoPosted, String description, String address, double salary
        // 2019-06-10 10:12:01.0
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        ServiceOffer offer;
        if(row[4] == null) {
            row[4] = "Not specified";
        }
        try {
            offer = new ServiceOffer(row[0].toString(), row[1].toString(), format.parse(row[2].toString()),
                row[3].toString(), row[4].toString(), row[5].toString(), Double.parseDouble(row[6].toString()),
                Integer.parseInt(row[7].toString()));

            list.add(offer);

        } catch (NumberFormatException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
return list;

```

figure 5.03

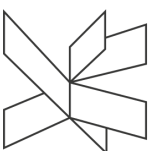
The following figure shows how the view loads the list of offer list retrieved from the second tier.

```

@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.jobTitle)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.jobCategory)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.description)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.whoPosted)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.address)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.bookingTime)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.salary)
        </td>
        <td>
            @Html.ActionLink("Contact", "Contact", new { id = item.id }) |
            @Html.ActionLink("Apply", "Apply", new { id = item.id })
        </td>
    </tr>
}

```

figure5.04



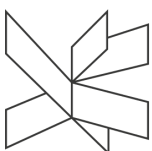
The next step is to make it possible for the user to apply for a service offer. To achieve this, a method `Apply(int Id)` is created, in this method a `foreach` loop (figure 5.05) is used to go through the `Ids` of each item in the list of offers to order to see which one has the same `Id` as what the user clicked "apply" on.

```
public ActionResult Apply(int Id)
{
    offers = ws.getServiceOffersNF(long.Parse(Session["user"].ToString())).ToList();

    foreach (var item in offers)
    {
        if (item.id == Id)
        {
            Session["id"] = item.id;
            return View(item);
        }
    }
    return View();
}
```

figure 5.05

After the user clicks on `apply` from the list of offers, they are redirected to a page that only shows that specific offer. Here they can now click `apply` again and they'd have applied for the service offer.



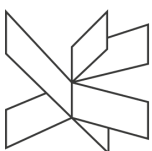
Submit apply:

Once they click the apply button, the cpr number and the Id of the listOffer item is passed as argument to the method addApplicant(long cpr, int postId) in Tier2(fig 5.2) where a check is done to see if the applicant has already applied for that specific offer.

```
@{  
    /**/  
    ViewBag.Title = "Apply";  
}  
<h2>Details</h2>  
  
@using (Html.BeginForm("SubmitApply", "Services"))  
{  
    @Html.AntiForgeryToken()  
  
    <div class="form-horizontal">  
        <h4>post</h4>  
        <hr />  
        @Html.ValidationSummary(true)  
        <div class="form-group">  
            @Html.LabelFor(model => model.description, htmlAttributes: new { @class = "control-label co  
            <div class="col-md-10">
```

```
public ActionResult SubmitApply()  
{  
    ws.addApplicant(long.Parse(Session["user"].ToString()), int.Parse(Session["id"].ToString()));  
    return RedirectToAction("Applications", "UserProfile");  
}
```

figures (5.06, 5.07) shows the tier1 code



```
@Override
public void addApplicant(long cpr, int postID) throws SQLException {
    post.addApplicant(cpr, postID);
}
```

figure 5.08 implementation of addApplicant in MyWebService class

```
public void addApplicant(long cpr, int postID) throws SQLException {
    instantiateDB();
    if(!db.alreadyApplied(cpr, postID)) {
        db.addApplicant(cpr, postID);
    }
}
```

figure 5.09 shows implementation of addApplicant in PostWSImpl

```
public void addApplicant(long cpr, int postID) throws SQLException {
    db.update(ADD_APPLICANT, cpr, postID);
}

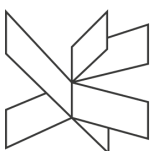
public boolean alreadyApplied(long cpr, int postID) throws SQLException {
    ArrayList<Object[]> table = db.query(CHECK_IF_USER_ALREADY_APPLIED, cpr, postID);
    if (table.size() >= 1) {
        return true;
    } else {
        return false;
    }
}
```

figure 5.10 shows implementation of addApplicants in PostDAO

After the SubmitApply() method in Tier1 controller has sent the post Id and the user's cpr number, the user is redirected to a new view where they can see a list of all their applications

```
public ActionResult Applications()
{
    try
    {
        return View(ws.getOffersIAppliedFor(long.Parse(Session["user"].ToString())).ToList());
    }
    catch (System.ServiceModel.FaultException e)
    {
        ViewBag.ErrorMessage = e.Message;
        return View("error");
    }
}
```

figure 5.11 shows how getting the user's applications process is done in the controller





The Applications() method in Tier1 invokes getOffersIAppliedFor(long cpr) in Tier2 which retrieves the list of applications from the database. If the list happens to be empty, an EmptyListException is thrown. This exception is caught in Tier2 in the Applications() method. If the list is empty, the user is redirected to an error page with an error message "You haven't applied for any service offers yet".

```
public ArrayList<ServiceOffer> getOffersIAppliedFor(long cpr) throws SQLException, EmptyListException {
    instantiatedDB();
    return db.getOffersIAppliedFor(cpr);
}
```

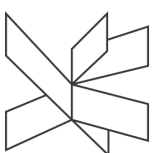
figure 5.12 shows how getting the user's applications process is done in PostWSImpl

```
public ArrayList<ServiceOffer> getOffersIAppliedFor(long cpr) throws SQLException, EmptyListException {
    //title, category, bookingtime, whoposted, description, address, salary, postid
    String cprAsString = cpr + "";
    ArrayList<Object[]> table = db.query(GET_OFFERS_I_APPLIED_FOR, cpr);
    if(table.size() == 0) {
        throw new EmptyListException("You haven't applied for any service offers yet");
    }
    ArrayList<ServiceOffer> list = new ArrayList<ServiceOffer>();

    for (int i = 0; i < table.size(); i++) {
        Object[] row = table.get(i);
        // String jobTitle, String jobCategory, Date bookingTime, String
        // whoPosted, String description, String address, double salary
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        ServiceOffer offer;
        try {
            offer = new ServiceOffer(row[0].toString(), row[1].toString(), format.parse(row[2].toString()),
                                    row[3].toString(), row[4].toString(), row[5].toString(), Double.parseDouble(row[6].toString()),
                                    Integer.parseInt(row[7].toString()));

            list.add(offer);
        } catch (NumberFormatException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    return list;
}
```

figure 5.12 shows how getting the user's applications process is done in the PostDAO



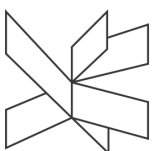
## 6 Test

The functionality of the system was tested at every sprint mainly with scenario testing, following the use cases, requirements and user stories. The following table shows each of the use cases and when the program works as expected after the testing process.

Use Case	
Sign Up	✓
View/Edit Profile	✓
Create Service Request and Offer	✓
View and Apply for service Request and offer	✓
View own posts requests and offers	✓
Delete service request and offer	✓
View applications and applicants	✓
Delete application	✓
Contact with other users	X
Employ	X

For each sprint it has been created a testing planning following the use case description related to the sprint with the intention of finding any possible error or a missing requirement.

During the implementation of the business logic, it has been used Unit testing, to ensure that the code was working as intended and detect any defects before having the view. Here is an example of white box testing used for Post offers.



```

@Test(expected = InvalidFormException.class)
public void testOfferTitleLong() {
    try {
        test.postOffer("oijuhygtfrdfghjkl;lkoijhugyftdrsedrfghjkl;klijuhygtfrdfghujikolhgfdxszasdfghjkl;lkjhg");
    } catch (ExpiredSession | InvalidFormException | SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

@Test(expected = InvalidFormException.class)
public void testOfferTitleEmpty() {
    try {
        test.postOffer("", "Cleaning", "hello", "98765437", "Kollegievaenget", "03-mar-23 22:33:22", "1000");
    } catch (ExpiredSession | InvalidFormException | SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

figure 6.01

Once the view was finished and all the Tiers have been connected, black box testing proceeded, and the program was tested with a user point of view, following the requirements to verify that the outputs were the expected.

More specify documentation about the testing for each sprint can be found in

## Appendix C.

## 7 Results and Discussion

Just Work is a website where users can post services offers and requests, and find temporary or punctual jobs in their city.

Users will have to log in with their account to get access to all of the website functionalities

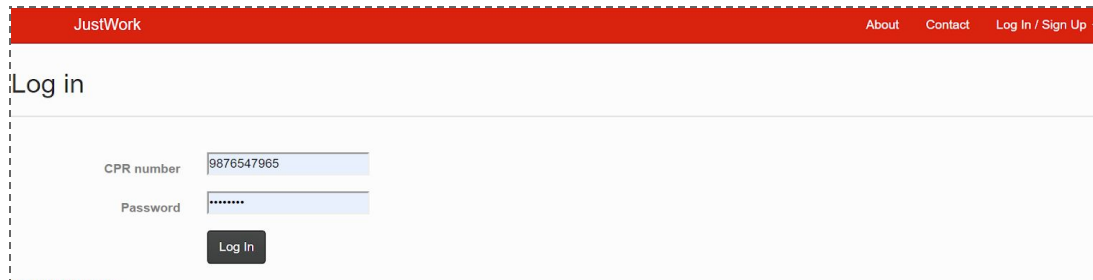
A screenshot of the JustWork website's login page. The page has a red header bar with the text 'JustWork' on the left and 'About Contact Log In / Sign Up' on the right. Below the header, the text 'Log in' is displayed. The login form contains two input fields: 'CPR number' with the value '9876547965' and 'Password' with masked characters '\*\*\*\*\*'. A 'Log In' button is positioned below the password field.

figure 7.01

In case that the user does not have an account they can create it completing all the fields.

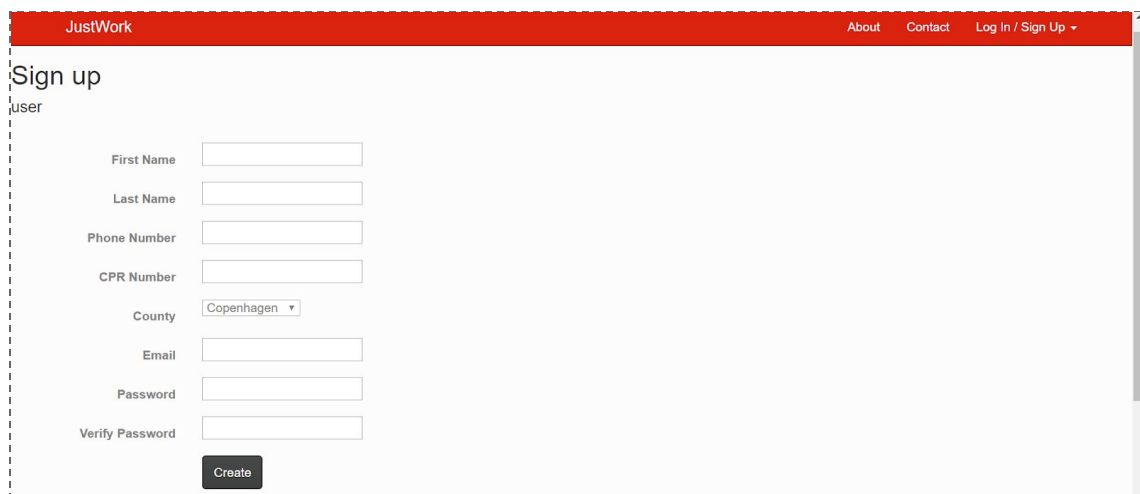
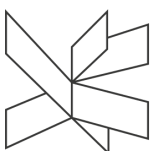
A screenshot of the JustWork website's sign up page. The page has a red header bar with the text 'JustWork' on the left and 'About Contact Log In / Sign Up' on the right. Below the header, the text 'Sign up' is displayed. The sign up form contains several input fields: 'First Name', 'Last Name', 'Phone Number', 'CPR Number', 'County' (a dropdown menu with 'Copenhagen' selected), 'Email', 'Password', and 'Verify Password'. A 'Create' button is positioned below the 'Verify Password' field.

figure 7.02



Once they are logged-in users will access to the homepage of the website with a new navigation bar with more functions.

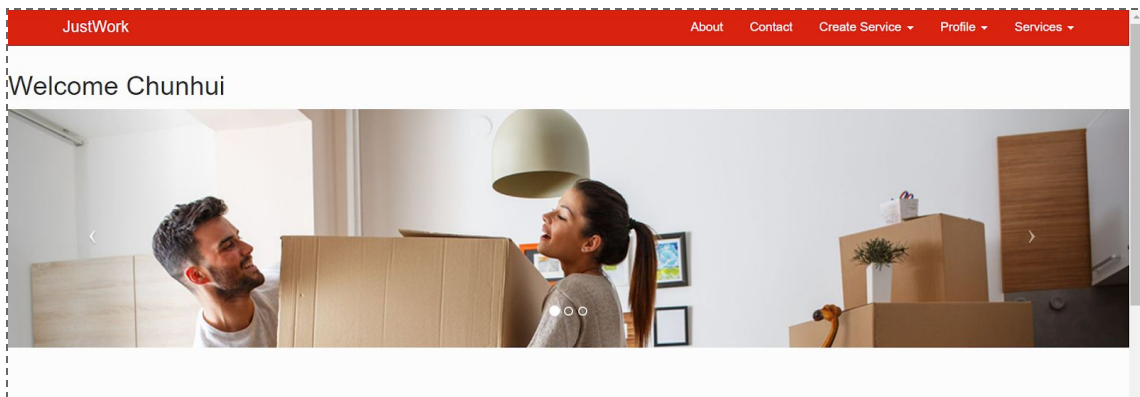


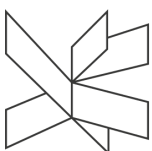
figure 7.03

From there, they can access their profile information add a description and edit their city, description, email and phone number. They can also change their password.

The description can include their skills and their contact information, since this information will be visible to other users.

A screenshot of the 'My Profile Details' form in the JustWork application. The form is located below the 'Welcome Chunhui' banner. It contains several input fields for user information: 'First Name' (Chunhui), 'Last Name' (Liu), 'CPR Number' (9876547965), 'County' (Vejle), 'Description' (Not specified), 'Email' (Liu@gmail.com), 'Change Password' (empty), 'Verify Password' (empty), and 'Phone Number' (92345678). A 'Save' button is at the bottom right of the form. The top navigation bar is red with white text for 'JustWork' and links for 'About', 'Contact', 'Create Service', 'Profile', and 'Services'.

figure 7.04



Users can create a post offering or requesting a service, once they have done it other users living in their same city would see them in a list.

JustWork

[About](#) [Contact](#) [Create Service](#) [Profile](#) [Services](#)

Welcome Chunhui

## Service Request

[Home](#) / [Service Request](#)

CreateServiceRequest

Service Title

Moving to new apartment

Service Category

Moving

Service Description

I need help with moving my t

Create

figure 7.05

From this list they would be able to apply to this service offer. First they will see a page with the details of the service and from there they can click apply

JustWork			About	Contact	Create Service	Profile	Services
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Contact   Apply
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Contact   Apply
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Contact   Apply
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Contact   Apply
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Contact   Apply
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Contact   Apply
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Contact   Apply
Justnow	Cleaning	hjhjhj	Chunhui Liu	kamtjatka 13	10/06/2019 10:12:01	45678	Contact   Apply
Moving to new apartment	Moving	I need help with moving my things to my new apartment	Chunhui Liu	kamtjatka 13	28/07/2019 10:12:01	500	Contact   Apply

figure 7.06



Welcome Maria

Details

post

description

jobCategory

jobTitle

whoPosted

figure 7.08

Once they have applied they will go to a page that will show all the applications they have made and unapply in case they are not interested anymore.

JustWork

AboutContactCreate ServiceProfileServices

welcome Maria

My Service Applications

Title	Category	Description	Posted By	Address	Booking Time	salary	
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Delete
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Delete
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Delete
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Delete
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Delete
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Delete
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Delete
juhykjhg	Cleaning	hello	nayyaa iuy	Kollegievaenget	03/03/2023 22:33:22	1000	Delete
Spising	Cooking	Help cooking and spising	Dan Iela	kamtjatka 13	03/06/2019 20:12:01	98	Delete
Justnow	Cleaning	hjhjhij	Chunhui Liu	kamtjatka 13	10/06/2019 10:12:01	45678	Delete
Moving to new apartment	Moving	I need help with moving my things to my new apartment	Chunhui Liu	kamtjatka 13	28/07/2019 10:12:01	500	Delete

figure 7.09

At the same time, once a user Apply for a service, their name and description will appear in a list of applicants that can be seen for the user that created, for that they need to go to My Service Offers and from there they will have the option. They can also delete the post.

JustWork

About

Contact

Create Service

Profile

Services

Welcome Chunhui

My Service Offers

Job Title	Description	Category	Address	Booking Time	Salary	
Justnow	hjhjhij	Cleaning	kamtjatka 13	10/06/2019 10:12:01	45678	<a>Edit</a>   <a>Applicants</a>   <a>Delete</a>
Moving to new apartment	I need help with moving my things to my new apartment	Moving	kamtjatka 13	28/07/2019 10:12:01	500	<a>Edit</a>   <a>Applicants</a>   <a>Delete</a>

figure 7.10



When the list of applicants is shown, users will be able to see the name of the applicant and their description. They will have the option to employ them or delete them from the list.

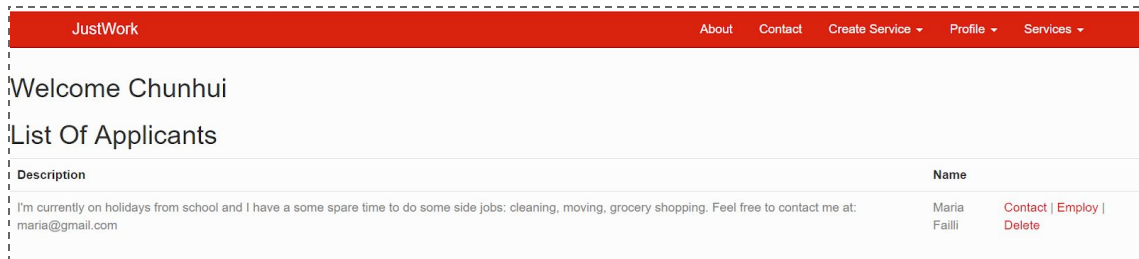
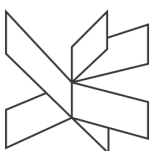


figure 7.11

To check that the final result was expected the project has been compared with the requirements previously chosen.

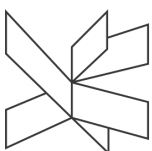
Requirements	
When a user first signs up, they will be asked about their name, phone, email, CPR, city, and they get to choose a password.	✓
Users will login using their CPR and password	✓
The worker will be able to post their job requirements by filling out a form where they write the job title, choose the category of the job, and a description.	✓
The employer will be able to post their job offers by filling out a form where they mention: job title, job category, time, description and address.	✓
Workers will be able to apply for different jobs	✓
The employers will be able to see the list of applicants and their profile(name, contact information) that applied for his/her job offer.	✓
Employers will be able to choose a worker from a list of all workers that applied	✓
The user will be able to create a profile with their description	✓
The user will be able to edit their profile information ( phone, email, city, and password).	✓
Workers and Employers will be able to communicate with each other by chatting system	X
Users will be able to see all the job offers close to their area	✓





Employers will pay the money once they choose a worker	✓
The worker will receive the money once the job is done	X
Old and outdated posts will not be shown in the list of offers/requests.	✓
The administrator will delete inappropriate(such as offensive language use) posts and users, and irrelevant posts(such as a cleaning job in gardening category).	✓
The website will have contact information of the administrators.	✓
The users will be able to contact the administrators by some live chat functionality	X

As it is shown the main requirements are fulfilled, some of them, like the payment security could not be handled by our team. The initial idea of the project was that users could contact through an online chat, but this function could not be implemented on time so the users would have to contact by the information provided in their profiles. In order to complete this project with all the functionalities the group predict that it will be needed 150 more hours.



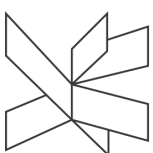
## 8 Conclusions

The main purpose of the system was to make it easier for users to provide and get a job. It is done using a 3-Tier-Architecture and a C# application together with java application and all data is stored inside a database. The conducted tests proved that all the critical parts of the system are working, and the goal of the project is fulfilled except for chatting and payment system.

After setting the functional and nonfunctional requirements of the system, the conclusion was a deeper understanding of the final product, whereas the analysis phase could start.

We can conclude from the analysis section the plan to start the design of the project, so from the domain model, the design of the system started by a class diagram and other designing diagrams, which led to the ability to implement those diagrams and test it.

Security aspects are really important when working on such a project with crucial information that must stay private, such as passwords, cpr numbers, credit card info, therefore it is important to keep the security in mind while analysing and designing, and always make sure that the risk + danger of a certain problem is as low as possible.



## 9 Project future

We have not managed to finish it all the way without any problems, we still have some small problems and bugs. Chat system and payment system hasn't been made because of lack of time. That will help employees and employers to use JustWork system better.

Systems security is not as good as it should be. To make it ready for production it should be added, because the system is handling sensitive information about employees and employers that must be secured well from the people who might want to use it for different purposes.



## 10 Sources of information

1. EUROSTUDENT comparative, 6th March, 2018:Combining studies ant paid jobs [online] Available at:

<[http://www.eurostudent.eu/conferences/finalconference/documents/EMP\\_II-1.pdf](http://www.eurostudent.eu/conferences/finalconference/documents/EMP_II-1.pdf)>

[Accessed 27<sup>th</sup> February 2019]

2. Kay Xander Mellish, 2019. Tips for living with a Danish family [online] Available At:

<<https://www.howtoliveindenmark.com/stories-about-life-in-denmark/living-with-a-danish-family-2/>> [Accessed 1<sup>st</sup> March 2019]

3. FlexJob: job flexibility is more important than pay. Available at:

<<https://www.thebalancecareers.com/working-parents-rank-flexibility-higher-salary-4085889>>

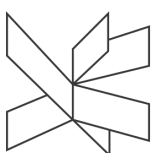
[Accessed 2<sup>nd</sup> March 2019]

4. Team LocalWise. 37 Best part time jobs for college students. Available at: -

<<https://www.localwise.com/a/100-37-best-part-time-jobs-for-college-students>>

[Accessed 2<sup>nd</sup> March 2019]

5. Kurose, J. and Ross, K. (2017). *Computer Networking*. 6th ed. Harlow, United Kingdom: Pearson Education Limited.



## 11 Appendices

Appendix A: Project Description

Appendix B: Diagrams

Appendix C: Testing Documentation

Appendix D: Tier1 implementation

Appendix E: Tier2 implementation

Appendix F: Tier3 implementation

Appendix H: User Stories

