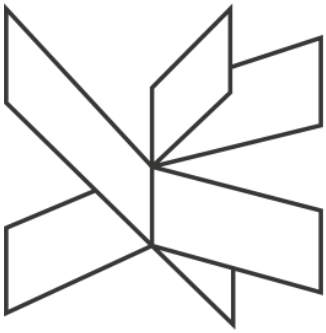


VIA University College

Semester Project: Heterogeneous System

Project Report – The Happy Pig Company Time Logging System

Supervisors:

Jakob Knop Rasmussen

Jan Munch Pedersen

Students:

Gais El-AAsi – 279910

Marcel Notenboom – 279963

Number of characters incl. spaces, images and footnotes 71,316 characters;

51,316 excluding images

Third Semester: Software Engineering

VIA University College – Horsens Campus 20th of December 2019

Table of content

List of Figures and Tables

Abstract

This project aims to provide a time logging system for The Happy Pig Company. The goal is to create a software system that will be more efficient, easy to use and maintain, and less prone to errors.

To achieve this, as well as provide possibilities for future scalability and expandability, the system must be distributed having a three-tier architecture. In addition, the system should be connected to a relational database and have a user interface.

The system should be accessible from anywhere with an internet connection, to satisfy that the user interface is a C# web application. The logic tier is designed as an Application Programming Interface that allows communication through HTTP Requests, as well as the possibility for future expansion of the client tier. The data tier uses Entity Framework technology that allows an easier design of the database through object-mapping.

Because the system is exposed to the internet, certain security policies were defined and the mechanisms to enforce them were put in place to ensure a degree of safety for the system.

The result of this project is a software system that has the basic architecture for a working product. Unfortunately, the system does not yet satisfy all the requirements in order to be usable in the company on a daily basis.

1. Introduction

As a part of the vegetarian growing industry, ***The Happy Pig*** is a Danish-based producer of meat-free sausages. Founded in 2015 with an initial workforce of 10 employees, it has grown year on year and now has plants across multiple countries (Lithuania, Germany).

In Denmark, the company manages a total number of around 100 employees including, production employees that consist of approximately 70 factory workers who deal with the manufacture of the products as well as production management. The rest of the employees are spread across the Management, HR, Research, and Design and, Sales and Marketing Department.

At the facility, the majority of the employees (production) work **fixed hours** (based on 3 shift/day system) some of them working **flexible hours** (engineering teams) with both **full time** and **part-time** employees.

Some of the employees (mostly the Sales Team) **work remotely**, with again fixed and flexible hours as well as full time and part-time workers. As a result of this growing workforce, the company has found it increasingly difficult to manage and keep track of the hours worked by its staff.

At present, employees that work at the facility with fixed and flexible hours register their hours via weekly timesheets, which then need to be checked and authorized by a manager and finally submitted to the Accounting Department for inclusion on the payroll. The employees that work remotely (Sales Team) have to send a weekly e-mail to their Supervisors, that will make their weekly timesheets and submit them to the Accounting with the specification of hours per day worked.

This method is inefficient and costly as it often results in incorrect data being input to the payroll. This is predominantly down to human error, be it on the part of the Accounting Department misreading a timesheet, an employee and /or their manager incorrectly remembering and recording the actual hours worked and also that of time theft, whereby employees deliberately falsify hours in order to gain payment for work they haven't done.

These problems have led the company to look for alternative methods that are available on the market for time logging.

The traditional punch card clock systems where employees each have individual cards that they use to stamp time and date for logging in and out are not a viable option anymore. It is inefficient and difficult to maintain, additionally, it creates space for errors since the staff has to manually input data that will be retrieved later on. In a 2017 survey of 500 small business owners, roughly 80 percent of the businesses surveyed admitted they regularly have to make corrections before they can run payroll (Soco Tax, 2019).

In regards to preventing and/or decreasing the cases of time theft, there are different techniques that can be applied, but it is still down to the Management of the company to ensure proper discipline in the company.

Currently, there are different, systems that use biometric authentication such as biometric terminals, proximity terminals, web-based clocking, mobile clocking, etc.

Biometric terminals do have the advantage of practically eliminating the problem of time theft via buddy punching, on the other hand, it usually involves relatively high costs for the equipment particularly iris and retina scanners.

Fingerprint authentication systems at the lower end of the market can be purchased and set up relatively cheaply, however, performance can be a problem. Environmental factors can play a major role in the accuracy of these systems, which can lead to one of two errors.

In regard to other systems, it is down to the company's needs to determine which system will fit best their situation and will improve the current state.

The problems faced by ***The Happy Pig*** company are certainly not unique and the process of developing a solution to this multifaceted problem through an exploration of required technologies and techniques should impart knowledge onto the team that can be adapted and utilized across a wide variety of possible future scenarios.

The purpose of this project is to provide ***The Happy Pig*** with an improved time logging system that will focus on the problems that the current system has, thus decreasing the number of issues as well as reducing the likelihood and frequency of erroneous data in their records.

For defining the broadness of the project, it is required to mention the areas and issues that this project will not address:

- The solution will not be for the facilities and/or employees in other countries where the company has facilities with the exception of Denmark;
- The solution will not involve any kind of biometric terminals (included but not limited to fingerprint terminals, retina terminal, etc.);
- The solution will not have a mobile application or any kind of mobile clocking;
- The solution will not have a Danish interface (only English);
- The solution will not include any kind of near-field communication technologies;

Next, the *Analysis Phase* of this project will be described, which is meant to provide the foundation and a skeleton for this project in the form of a **Domain Model**.

2. Analysis

In this chapter, a series of analyses will be done with the scope of understanding and elaborating on the problem domain, whilst focusing on what the involved stakeholder expects.
The result of this chapter will be a Domain Model that will describe how the different building blocks will interact with each other, the Use Cases and Test Cases as well as other relevant illustrations and diagrams that will help understand better the domain problem.
But first, the requirements for this project will be described to ensure the basis of what is expected from the outcome of this project.

2.1. Requirements

In this subchapter, the requirements for the project will be described, which will be crucial through the entire project as a baseline to follow and keep in mind. The requirements were developed together with the product owner that provided insights into what is expected from the solution.

When defining the requirements, the SMART principles (Cristancho, 2019) were kept in mind to safeguard that they will be of value through the project, rather than an impediment.

2.1.1. Functional Requirements

The functional requirements needed a significant amount of attention as they will provide the backbone for both the project's outcome, as well as a reference point. The requirements were defined in the form of *User Stories* (Larman, 2004).

A general form was defined for the functional requirements to provide a similar definition across all of them: *As a **user,** I want **feature** so that a **user** can **business value/benefit***.

In addition, the requirements were categorized using the MoSCoW technique (volkerdon, 2019), eliminating the **W** (won't have) section as, they were described in the *Introduction Chapter* or in the *Delimitations Chapter*, of the Project Description (Appendix A).

Table 1 - Functional Requirements

MOSCOW	REQUIREMENTS
MUST	As a **Manager,** I want to be able to **see work and break registrations** so that **I can have a better understanding of who was working when.**
MUST	As a **Manager**, I want to be able to **add/edit/delete an employee from the system** so that **I can manage the currently employed people.**
MUST	As a **Manager**, I want to be sure **that information is securely stored** so that **I know it is not lost in cases of power outages, floods, etc.**
MUST	As an **Employee,** I want to be able **to punch in/out in the system quickly** so that **I can spend my time doing something else.**
MUST	As an **Employee,** I want to be able **to easily register when I am taking breaks** so that **the management knows when and how long a break was.**
MUST	As an **Employee,** I want to be **informed if a registration was not made due to various reasons** so that I know that **I have to try again or contact one of the managers.**
MUST	As an **Employee,** I want to be sure **that my information is private to me/managers** so that **nobody without authorization can access it.**
MUST	As a **Remote Employee,** I want to be able **to punch in/out from any location with access to the internet** so that **I can work easily remotely.**
SHOULD	As a **Manager,** I want to be able **to see statistical information in regards to a certain employee/group of employees** so that **I can better prepare/adjust my management planning.**
COULD	As a **Manager,** I want to be able **to easily create/edit/delete accounts for temporary workers** so that I can **easily assign them to work.**

Above are presented some of the requirements from each MoSCoW category, the rest of them can be found in the Appendix B.

2.1.2. Non-Functional Requirements

When defining the non-functional requirements, the focus was on establishing the overall internal functionality of the system and the basic technologies needed to achieve the desired outcome for the project.

- The **system** must be a three-tier system setup –thus having a distributed system architecture;
- The main programming languages for the **system** must be C# and Java (may also include other support languages – HTML, CSS, JSON, JavaScript, SQL, etc);
- The **system** must have a Graphical User Interface at least on the client-side (accessible on world wide web);
- The **system** must use web-services and socket communication protocols between different tiers;
- The **system** must include a relational database to store/retrieve data;
- The communication between the tiers must be secure thus including SSL/TLS protocols;

2.2. Definitions and glossary

Before caring on with the analysis it is essential to define a common ground of terms used throughout the entire project. Because of that a need for a glossary arises.

2.2.1. Actor Description

- **Manager** – refers to both upper management (Accounting, HR, Production Management, etc.) as well as middle management (Shift Managers, Team Leaders, etc.) they do not register their time as it is counted in other ways that do not include the punch in system, still, they have an interested for accessing the system to retrieve information in regards to other employees, all managers having the same access level to the information about the employees;
- **Facility Employee-Fixed** (full-time/part-time) – employees that work in the facility fixed schedules (shifts) that are agreed upon, they have to be physically present to punch in/out of the system;

- **Facility Employee-Flexible** – employees that work in the facility flexible hours, meaning they are not bound by a schedule but rather by an "on-call" scheme; they still have to be physically present to punch in/out;
- **Remote Employee** (full-time/part-time) – employees that not necessary work only in the offices, many times working from special locations (home, partners/clients offices, etc.) it is important for them to get an easy access to the punch in system to register their work-time; They do not have a fixed schedule, but rather a certain amount of hours per day that they need to work;
- **Temporary Employee** – are facility employees with a fixed schedule that do not have a direct contract with the company but are rather employed for a short amount of time (minimum one day) through a recruitment company;
- **Employee** – generally refers to all of the above (except managers);
- **Recruitment Company** – partners that provide the company with temporary workers;

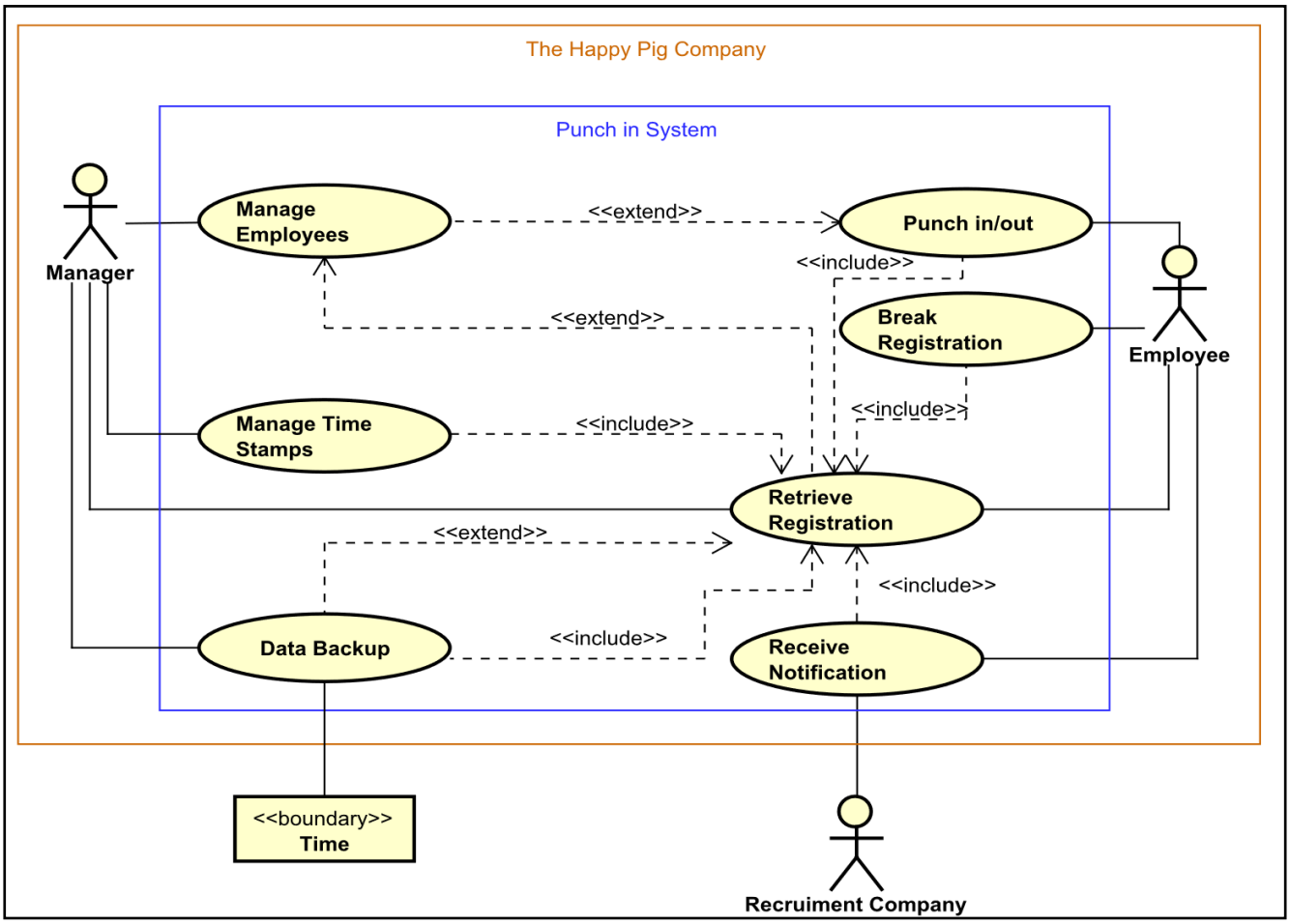
2.2.2. Project Glossary

- **Time Stamp** – Commonly refers to both of the below ones;
- **Punch Stamp** – A punch stamp is either a punch in or a punch-out time registration for work. It denotes when an employee started/finished the work time;
- **Break Stamp** – A break stamp is either a punch-in or punch-out time registration time for a break. It denotes when an employee started/finished the break time;
- **Employee ID (**also referred to as employee number) – is a unique identification number for each employee and manager. It is used in diverse functions of the system;
- **Registration** – Refers to both time stamp registration as well as a record of an employee in the system;

2.3. Use Case Diagram

The next step in the analysis process will be defining a ***Use Case Diagram*** that will describe the overall functionality of the system. When shaping it, the requirements described above were considered for the diagram to be within the expectations of the product owner.

Figure 1 - Use Case Diagram



The first point to observe is actors that are within ***The Happy Pig Company*** - Manager and Employee – and outside the company which is presented by the Recruitments Company.

A ***boundary*** is present that represents the time, it was needed to be included to point out the fact that *Data Backup* use case will be automatically triggered by a time frame. The foundation for not making *time* an actor, but just a boundary was inspired by Anthony's Crain article *Dear Dr. Use Case: Is the Clock and Actor?* (Crain, 2002)

Another aspect is that *Data Backup* is in close relation with the *Retrieve Registration* use case as they both rely on each other. When *Data Backup* is triggered it is retrieving the registrations from the *Retrieve Registration* use case thus the <<include>> relation.

On the other hand, the *Retrieve Registration* use case can sometimes rely on the *Data Backup* use case when certain data was lost and it needs to be accessed from the backup, therefore the <<extend>> relation.

One last point worth mentioning is that all the use cases rely on the *Retrieve Registration* use case as it is the backbone for any data manipulation within the system.

2.4. Use Case Description

In this subchapter the Use Case (UC) Descriptions (Punch in/out Use Case – the rest of the use case descriptions can be found in Appendix D) that helped understand better how a certain task will be performed.

When defining the UC Description, both the Base Flow and Alternative Flow were considered and indications made based on the state of the situation.

Table 2 - Use Case Description - Punch in/out

ITEM	VALUE		
Use Case	Punch in/out		
Summary	An employee is able to register a timestamp at which the employee started/ender the work		
Actor	Employee		
Precondition	The employee must be registered in the system in order to punch		
Base Flow (BF)		**Actor**	**System**
	1.	The employee decides to punch in/out	
	2.	The employee enters his/her credentials	
	3.		The system notifies that the registration was successful with the time stamp
	4.		The system sends a text message with the time stamp
	5.	The employee receives a text message with the time stamp	
	6.		System returns to the initial state
Alternative Flow (AF)	A. (Base Flow 2.) The employee enters invalid (not present in the system) credentials 1. The system notifies that the registration of the time could not be performed 2. System returns to the initial state after the notification is closed B. (Base Flow 5.) The employee did not receive a text message with the time stamp 1. Employee retries to perform the Base Flow again, and it is successful; 1.1. (Recommendation) The employee informs the manager to check for anomalies 2. Employee retries to perform the Base Flow again, with the same result described in step AF - B; 2.1. The employee informs the manager to check for anomalies		

	C. (Facility Employee – fixed: Base Flow 2) The employee enters his/her credentials 1. The system notifies that the time is either over the start-time or before the end-time 2. Employee selects the reason of him/her being late/early 3. (Optional) The employee enters a comment in the comment section 4. The system notifies that registration was successful 5. The system sends a text message with the time stamp 6. The employee receives a text message with the time stamp and the specified reason 7. System returns to the initial state; D. (Base flow 2) The employee enters his/her credentials for punch out; 1. The employee is notified that he/she cannot punch out before coming from the break; 2. Employee punches out for the break (see Break Registration Use Case); 3. Employee retries to punch out successfully -> Base Flow
Note	After each task performed the system updates the changes; In cases of the terminals used in the facility the text should be big enough so it can be seen from at least 1m distance; In cases of the terminals used in the facility the terminal so be positioned at the main worker entrances of the facility as well as installed at a comfortable height to be used when standing.

During the development of the UC Descriptions, significant attention was paid to considering all the possible paths that a user of the system might encounter during his/her experience with the system.

As an example, the alternative flow considers different paths that are both triggered by the human error (*Alternative Flow A. − Employee enters invalid credentials)* as well as anomalies in the system (*Alternative Flow C. No text notifications sent)*.

The UC Descriptions try as well to cover what are the preconditions for it to be used (ex. *The employee must be registered in the system)* as well as some notes that are some additional information (*After each task […] when standing)*.

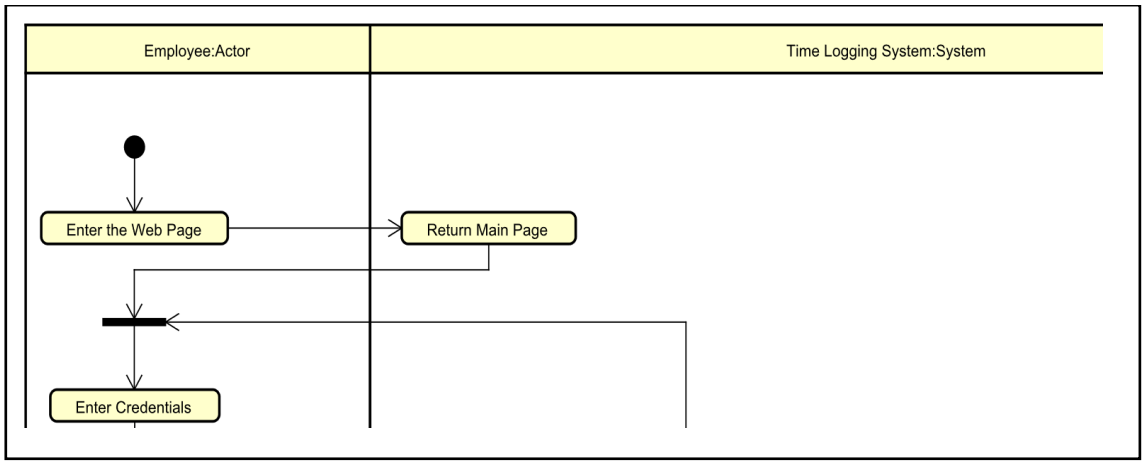
All in all, the UC Descriptions provide great insight for understanding the overall functioning principles of the system and foundation for the *Design Phase*. When considering the presented above UC Descriptions, combined with the Activity Diagram presented next, a clearer picture of the system can be observed.

2.5. Activity Diagram

In this subchapter, an Activity Diagram (AD) was developed to better observe and understand the flow of actions needed to perform the *Punch in/out and Break-in/out* time registration process. The AD was needed to properly understand the entire process, the time registration process being at the core of software and one of the *musts.*

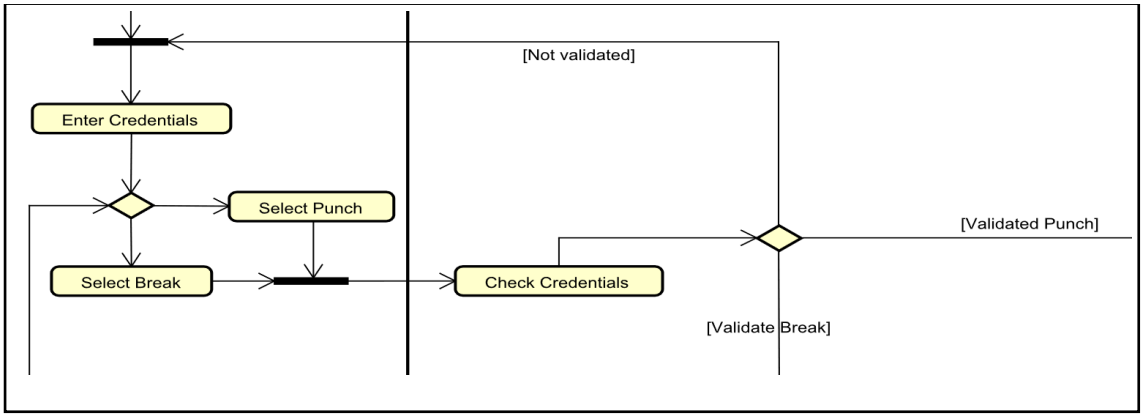
The AD not only helps better observer the inner functionality of the system but will be of much use in the next chapters when the design and implementation of the software will start.

Figure 2 - Activity Diagram Punch in/out and Break in/out – Part I



The entire process starts with Actor's desire (an Employee in this case) to register a time, thus accessing the web page (using the terminals in case of facilities) and entering his/her credentials (Employee number).

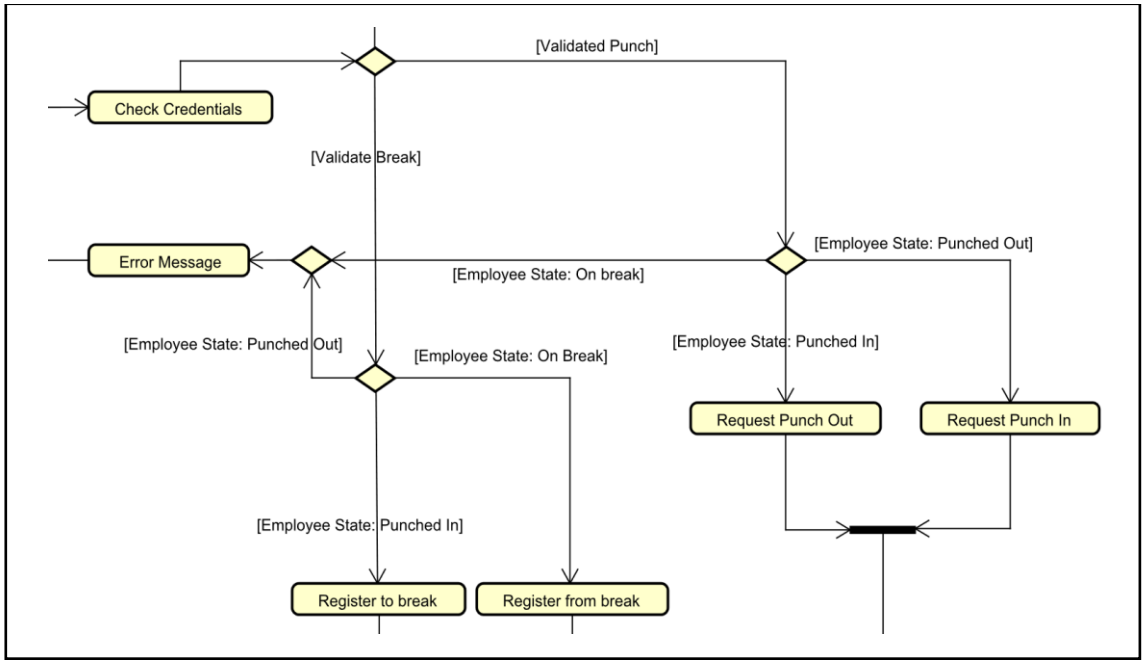
Figure 3 - Activity Diagram Punch in/out and Break in/out – Part ||



In the second part the Employee chooses whether he/she wants to register a work or break time and submitting the request. After submission the request is taken over by the system that performs a check of the credentials, resulting in three outcomes:

- Validated for Break;
- Validated for Punch;
- Not Validated;

Figure 4 - Activity Diagram Punch in/out and Break in/out – Part III



The next part is very similar to a *state machine* where the system is acting accordingly based on the previous state of the registration as follows:

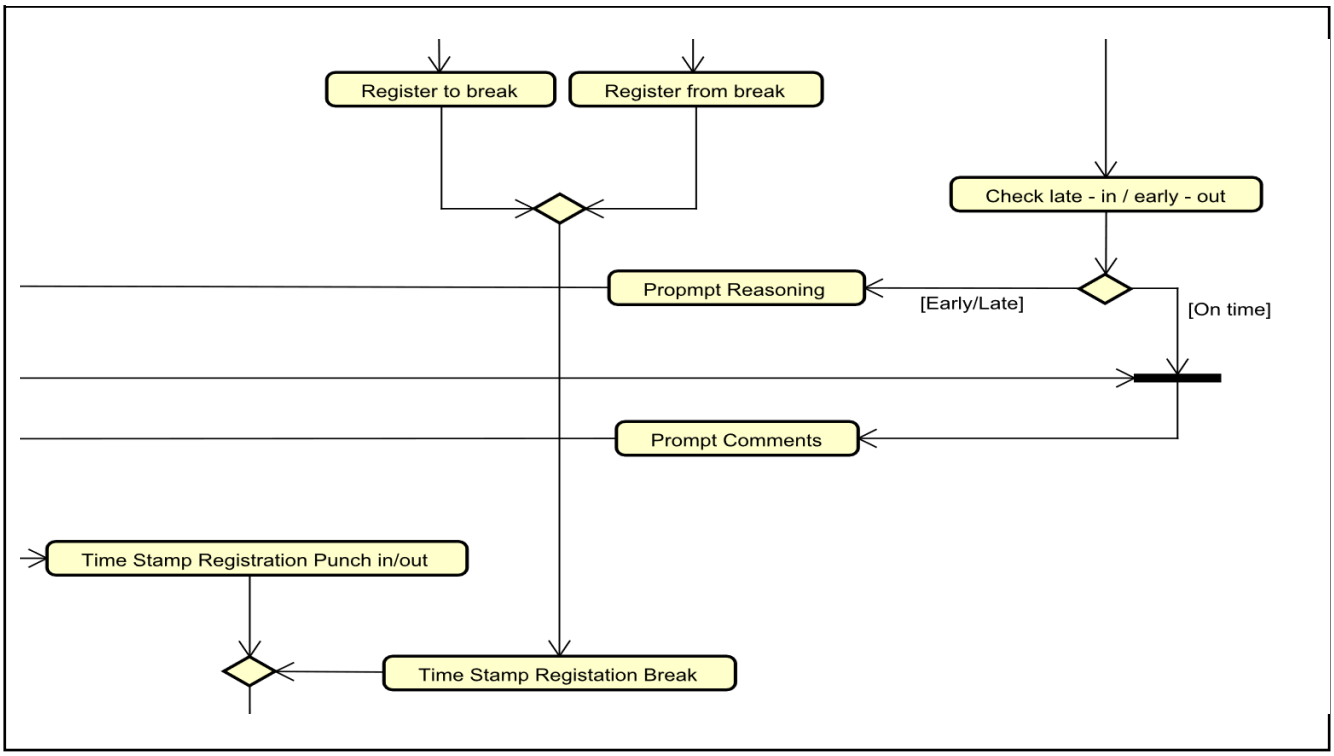
Table 3 - Registration Logic

PREVIOUS STATE	SET STATE
WORK TIME REGISTRATION	
PUNCHED OUT	Punch In
PUNCHED IN	Punch-Out
PUNCHED IN & TO BREAK	Error Message
BREAK TIME REGISTRATION	

PUNCHED OUT	Error Message
PUNCHED IN	To Break
PUNCHED IN & TO BREAK	From Break

Based on the state the system will follow a certain path, including the *go back* path in case of an error.

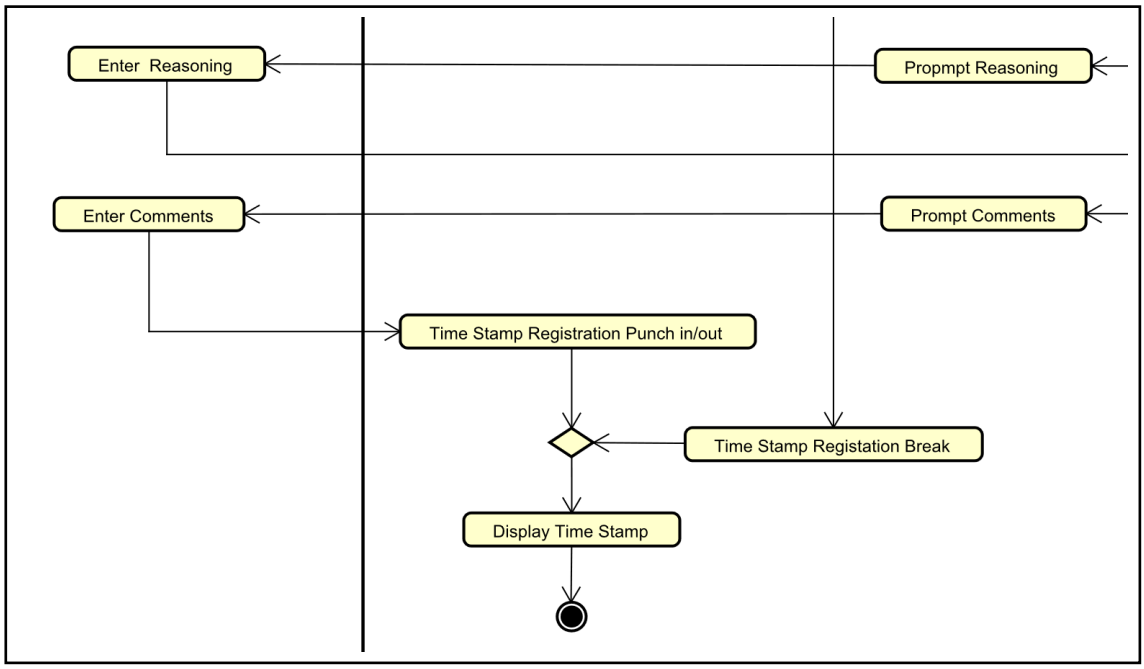
Figure 5 - Activity Diagram Punch in/out and Break in/out – Part IV



After deciding on the right registration, in case of break registration, the system will register the time stamp in the system. In the case of work time registration, the system will check if the employee is late for work/early to leave and ask for reasoning as well as ask for additional comments before registering the time stamp.

The last part where the system prompts for reasoning and comments, as well as registers the time stamp and terminates the process.

Figure 6 - Activity Diagram Punch in/out and Break in/out – Part V

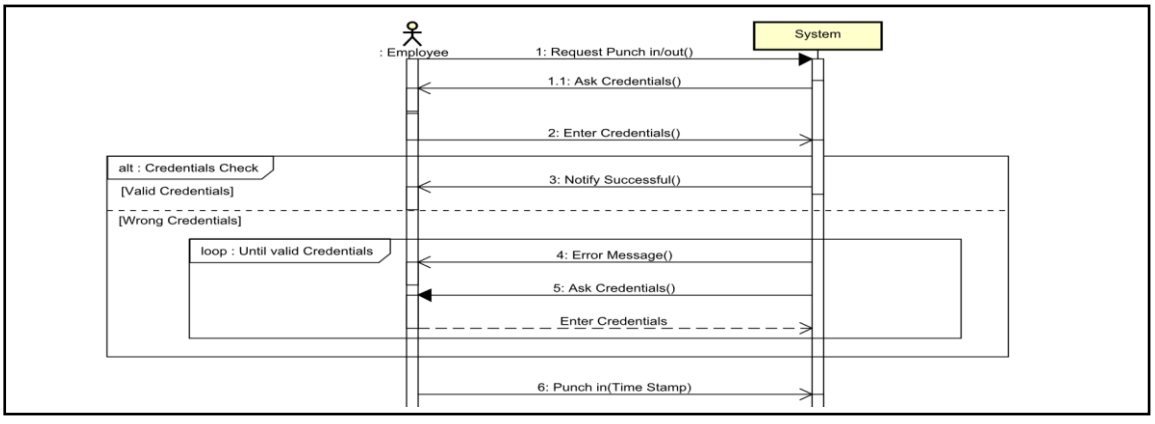


Developing the AD provided great knowledge when it comes to components of the system that will be necessary to consider when starting the design, as well as a good visual illustration for the Product Owner.

2.6. System Sequence Diagram

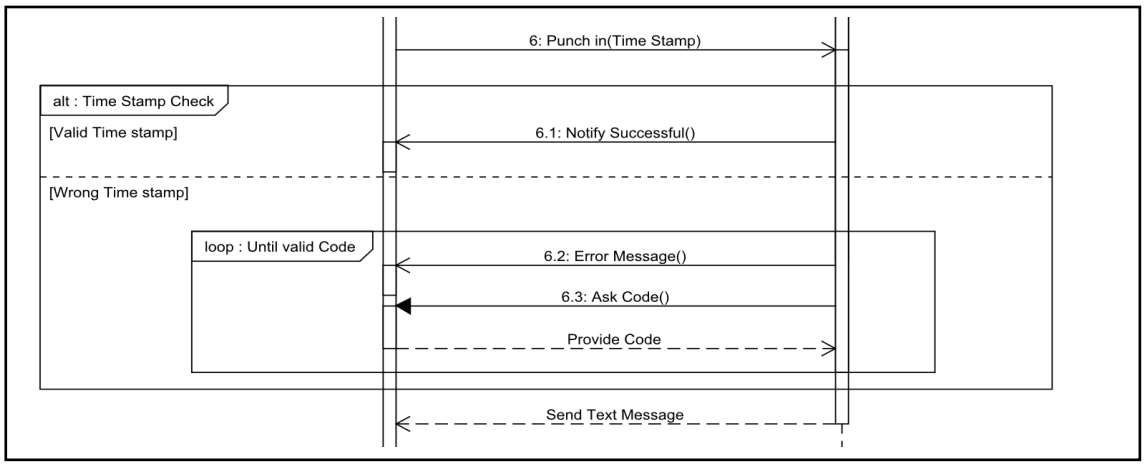
An additional diagram, that will focus more on understanding the flow of information between the user of the system and the system itself will be the System Sequence Diagram (SSD). The diagram presents essentials functions of the system that go through it, but in this chapter, the focus will be on the time stamp registration, the entire diagram can be found in Appendix F.

Figure 7 - System Sequence Diagram Part I



In the part I the request to register work time is made, the system than asking for the credentials and not allowing to pass through until the credentials are validated.

Figure 8 - System Sequence Diagram Part II

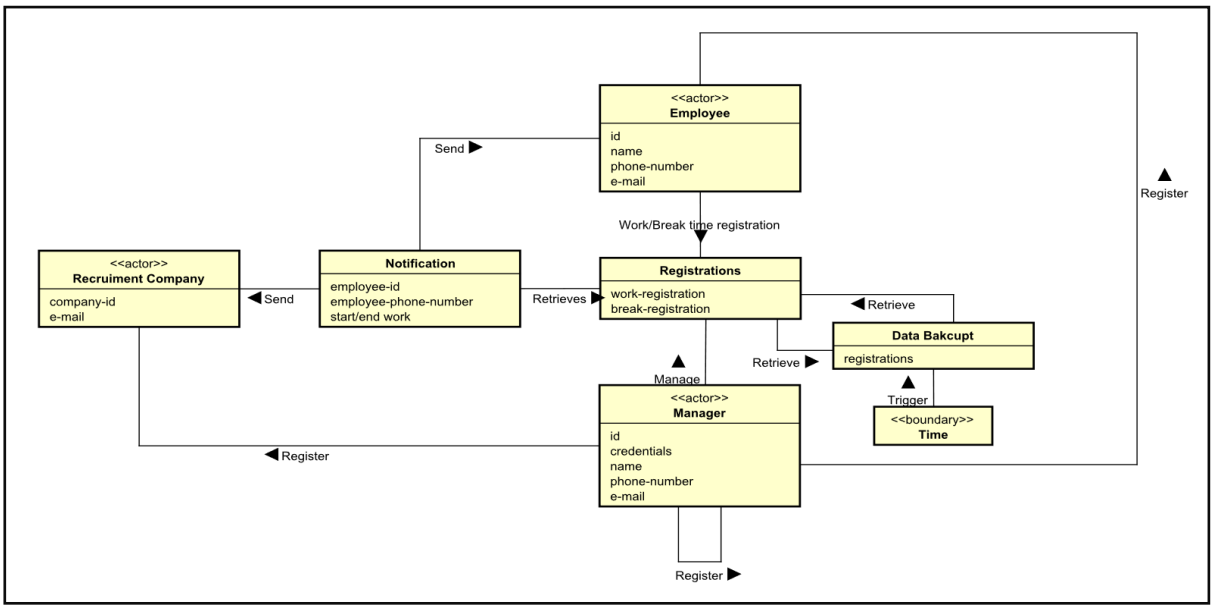


Next in the case of wrong information (punch-out while on break, coming late, etc.) the system prompts for additional information and to re-enter the wrong information in order for the flow to continue.

2.7. Domain Model

As an overview of the entire system and how different components of the system interact with each other, developing the Domain Model (DM) will provide great insights.

Figure 9 - Domain Model



When developing the DM, all previous analyses and diagrams, served as a foundation for defining the concrete components presented in the model.

As can be seen, there are a few essential blocks of the model, the *Employee/Manager*, and *Registrations* that are central for the functionality of the system, and that will the building "bones" of the skeleton for the project. Those concepts are central also because they are present through the entire system, beginning with the *musts* requirements.

2.8. Test Cases

To be able to measure and analyze the proper functioning of different functionalities, Test Cases (TC) (Larman, 2004) were made. The TC targets small functionalities of the system, by applying an action and expecting a certain reaction from the system. The TC was made to be used after the implementation phase as a measurement tool to compare if what was indented from the analysis was achieved. The results of the test will be discussed in the *Test Specification Chapter*.
Next will be presented a few of the proposed TC for the parts of the system, the entire list can be found in Appendix G.

Table 4 - Test Cases

No.	Action	Reaction	Use Case
1.	Employee selects punch/break option, enters credentials	Check if punch option selected, check if only allow characters are in the input	Punch In/Out Break Registration
2.	Employee submits a request for work/break time registration	Check if the appropriate message is displayed (*Punched In / Punched Out/To Break/ From Break / Error Message)*	Punch In/out Break Registration
3.	Manager deletes a registration	Check if the registration is not displayed in the list of registration	Manage Time Stamps
4.	Manager edits a registration	Check if the registration was modified	Manage Time Stamps
5.	Manager registers a new Employee	Check if the new employee's information is displayed in all employee view	Manage Employees
6.	Manager edits an employee	Check if the modified information is displayed correctly	Manage Employees
7.	Manager deletes an employee	Check if the employee is not displayed anymore in the list of all employees	Manage Employees

As an example, the action *Manager edits an employee* gives a reaction from the system to display the edited information about the employee in all employees' view. It is important that the TC are rather generic, as they should not influence the Design and Implementation from a *how* perspective, but rather only guide what should be the outcomes of certain functionalities.

2.9. Security

For a system to be able to function as intended and be reliable, it is of major importance to have consideration for the **security policies** and **mechanisms** that will enforce them. Along with that a **threat model** will be made to identify the main risks involved, and in the *Design Chapter,* the mechanisms against them will be described.

2.9.1. Security Objectives

The main objective of the security policies for the system will be to keep the **Confidentiality, Integrity,** and **Availability** (**CIA Triad**) uncompromised.

2.9.1.1. Confidentiality

In a time logging software confidentiality of the information plays an essential role in the well-being of the system and the company itself. Therefore, making sure that there are no leaks as well as easy access to sensitive information is a significant objective.

2.9.1.2. Integrity

Due to the fact that on the information preserved by the system will not only rely on direct users of the system, but other departments of the company – Accounting, Higher Management – or outside the company – Tax Agency, Unions – the safety that the data has not been tampered is essential that cannot be compromised.

2.9.1.3. Availability

The time logging system should work as long as the company is in production, considering that the company works in three shifts – day, evening and night – the system should be available 24/7/365 (with the exception of a few official holidays, and two weeks in the Christmas period). With that in mind, the company and its workers need to

know that the system will be reliable and available at any time without causing more trouble than it is supposed to solve.

2.9.2. Threat Model

To understand, identify and mitigate security risks in software, there is a need for a systematic and structured way of doing it. Threat Modeling is a great tool that aids in providing an overview of potential threats to the system.

2.9.2.1. Goals of the attacker

First, it is needed to understand what are the goals of the attacker. A good methodology that aids in providing a more organized way of identifying the goals of the attacker, developed by Microsoft is **STRIDE**.

As described by STRIDE methodology there are 6 categories of risks that need to be considered in forming a threat model: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privileges.

Spoofing:

When considering spoofing, the attacker's goal is of posing as someone else in the system thus affecting the **authenticity** of the user. When looking at the system, there are two types of users that the attacker would look into spoof:

- Employee;
- Manager;

The means of the attacker, in this case, are **brut-forcing** and setting up a **phishing host**, as well as the attack when tiers are communicating for example **eavesdropping** and **traffic analysis**.

Tampering:

Tampering refers to an unwanted and malicious modification of different information. This affects the **integrity** of the data that the system uses. In the system, the tampering can occur to modify the registration of the work/break time as well as the information about a user, that would affect negatively not only the time logging system but the accounting process, management planning, etc.

The attacker can modify data both in transit and at rest in the system. Some means by which it can be done are **SQL Injections** (data at rest), or **modification of message** (data in transit) through the **ciphertext-only attack**, **known-plaintext attack**, **chosen-plaintext attack**, **chosen-ciphertext attack**.

Repudiation:

The repudiation attack refers to not receiving **confirmation** that certain processes occurred. An example that could affect negatively the system is at the registration of work/break time when an employee would not receive a confirmation that the registration was successful. This attack will result in multiple registrations (because the employee will keep trying until successful) and general malfunctioning of the system.

Information Disclosure:

The information disclosure is a direct attack on the **confidentially** of the data in the system that includes data breaches of the sensitive information (employee's data, registrations, etc.) that could potentially **damage** the well-functioning of the company. The Information Disclosure can occur at each tier as well as when the information is in transit between the tiers.

Denial of Service:

It refers to affecting the network by making the system unreachable by the user thus affecting the **availability**. It can occur by attackers performing **SYN flood**, **DDoS attacks** or directly on the storage**, overflowing** it.

Elevation of Privileges:

The elevation of privileges affects the authenticity of the system, by providing users with unwanted levels of access to sensitive data. In the system, it can occur when users would give themselves manager's privileges to gain access to information otherwise unavailable to them. Along with the means provided in for the spoofing, the attacker can also do a **replay attack** to gain elevated privileges.

2.9.2.2. EINOO

Other things to consider when developing the threat model is where and by whom are the different attacks on the security of the system made.

First, it is important to understand whether the **threat** is coming from the inside (**Internal**) or from the outside (**External**) of the company. The inside attackers can be employees with non-managerial rights trying to obtain some gains or just simple human error that would make the system vulnerable to potential dangers.

Next to consider is where the attack can occur, in this case having three possibilities: **Network**, **Offline,** and **Online**. Due to the architecture of the system (distributed system with three tiers) and the need for the system to be online, to make it available for the remote users, the system increases the public attack surface area giving more possibilities to the attacker.

2.9.3. Risk Assessment

Before defining a Risk Assessment Model, it is important to describe what are the variables that define a risk.

The incident likelihood is defined by the sum of threat frequency and preventive measures;

Threat Frequency – is the possibility (chance) of a threat occurring;

Preventive measures – describe the actions made to decrees the chances of a threat occurring;

The incident consequence is defined by the sum of threat effect and corrective measures;

Threat effect – is formed by the consequence of a threat that occurred;

Corrective measure – describe how easy would be to “fix” the consequences of a threat that occurred;

Next will be presented a Risk Assessment Model, inspired by the *Guide for Conduction Risk Assessments* of the *National Institute of Standards and Technology U.S.* (Rebecca M. Blank, 2012)

Table 5 - Risk Assessment

Threat Event	Threat Source	Threat source Characteristics			Relevance	Likelihood of Attack Initiation	Vulnerabilities and Predisposing Conditions	Severity and Perverseness	Likelihood Initiated attack Succeeds	Overall Likelihood	Level of impact	Risk	Risk Score (1- 5)
		Capability	Intent	Targeting									
System Intrusion and unauthorized system access	External Internal	Moderate	High	Moderate	Possible	Moderate	Possible weak passwords due to a lack of password complexity control.	High	Moderate	Moderate	High	High	8
Tempering of data in the system	External Internal	Low	Moderate	High	Possible	Moderate	Inadequate implementation of protection against SQL Injection	High	Moderate	Moderate	High	High	8
Denial of service attacks	External	Moderate	Low	Moderate	Possible	Moderate	Missing implementation for protection against DoS	Moderate	Moderate	Moderate	Moderate	Moderate	5
Unauthorized elevation of managerial privileges	External Internal	Moderate	High	High	Unlikely	High	Weak passwords, unattended logged-in machines	Moderate	High	Moderate	Low	Low	2
Replay Attack	External	Low	Moderate	Low	Unlikely	Low	Absence of SSL between tiers	Moderate	Low	Low	Low	Low	2

Continuing, int the *Design Chapter* will be described some mechanisms that enforce the security policies to prevent, mitigate and correct possible threats.

3. Design

This chapter will address a more in-depth development of the solution, the result of this chapter should form a more concrete shape of the software. It will focus on providing specific artifacts that will be extensively used in the implementation phase.

Nevertheless, all the development will still be based on the analyses done before as a foundation. The non-functional requirements will influence strongly this phase as they offer important guidance on how the system should function internationally.

3.1. System Architecture

The first step is deciding on an architecture for the system that will present how will the responsibilities be split in the system between the different components as well as how will the components communicate with each other.

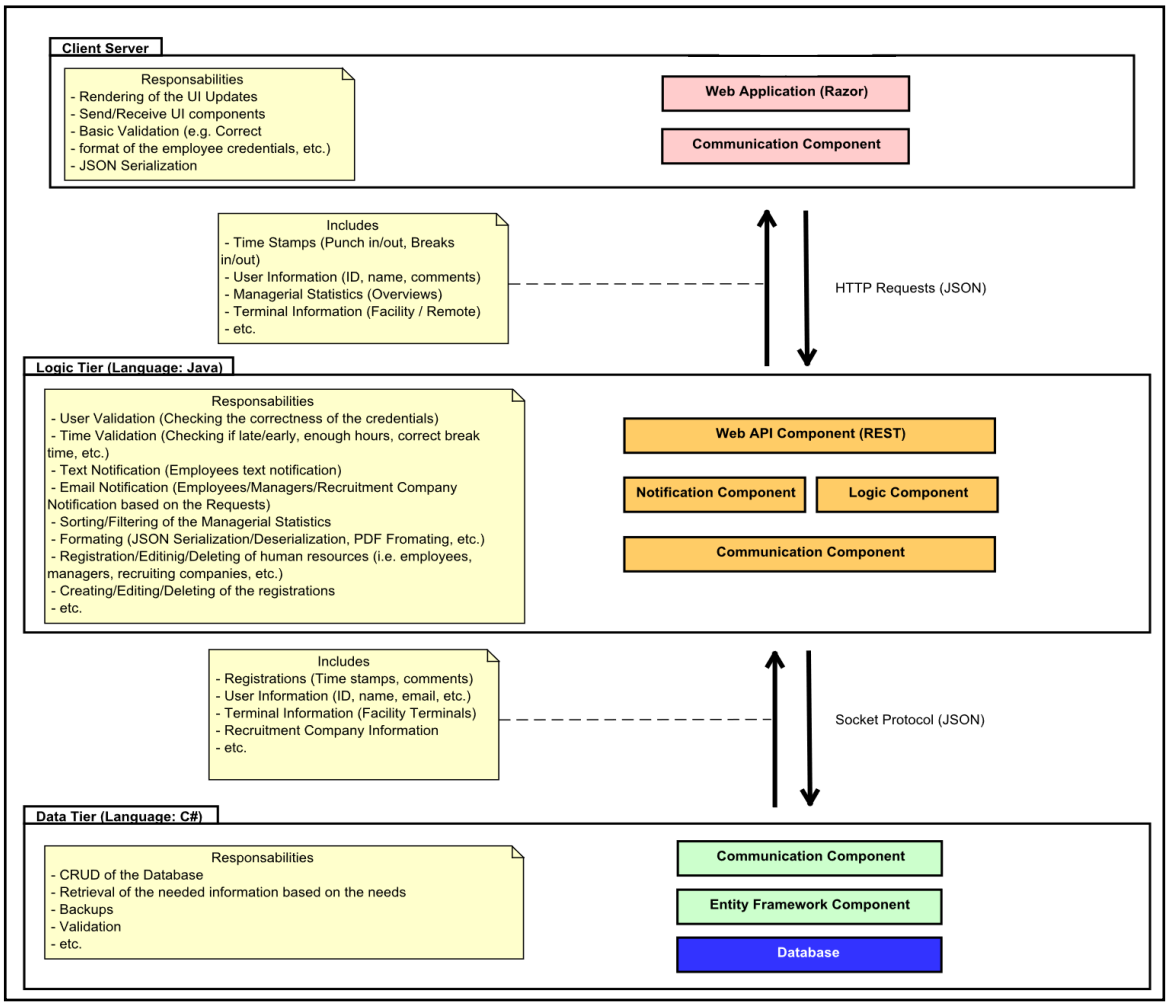
When considering the architecture, the non-functional requirements provide important indications on how should the system components be organized.

It must be a three-tier distributed system that uses both C# and Java as main programming languages. The communication between tiers should be made using web services and socket communication as well as having a relation database.

The architecture diagram in Figure 10 was developed for the system, it respects the imposed requirements as well as presents a few notes in regards to the communication and tier's responsibility.

The first tier will have the responsibility of communication with the clients, through a web-based interface. It will not include any logic, to ensure the separation of responsibilities principle, and will be developed using C# language and Razor technologies. The visual display will be made using HTML, CSS and JavaScript as well as common frameworks designed for it (Bootstrap, Materialize, etc.)

Figure 10 - Architecture Diagram



The second tier will be responsible for the logic for the system and will be a RESTful API developed in Java. It will be in charge of the processing of the information, validation of data and users, data manipulation, etc.
The third tier will be responsible for CRUD[1] functions of data in close connection with a database.

The communication between the first tier and logic tier will be done through HTML Requests, and the communication between the logic and data tier will be done through socket communication.

[1] Create, Read, Update, Delete

3.2. Technologies and Communication

In this subchapter, a more in-depth aspect of the architecture diagram will be presented with a focus on the technologies and communication used providing reasoning for the choices.

3.2.1. Razor Pages

One of the *must* requirements was that access to the time logging system should be available from any location with connection to the internet. A good way to ensure that was to make it web-accessible as it does not require any kind of installation, only a browser.

After consideration of a few technologies that will allow building web applications including (*Razor MVC – C#, Blazor – C#, TeaVM – Java, GWT – Java)* that will provide necessary functionalities for what was desired, it was concluded that Razor Pages will be the best option. (Microsoft, 2019) (Microsoft, 2019) (Microsoft, 2019) (TeaVM, 2019) (GWT, 2019)

The reasoning for choosing Razor Pages is that it provides a simple way for developing web application still having powerful tools that include:

- Similarities with the MVVM Pattern for the internal functionality – enabling two-way binding;
- Inline code;
- Very organized structure;
- Respects the single responsibility principle;
- Established technology with fewer bugs (compared with Blazor) (Microsoft, 2019)

3.2.2. REST API

The decision to make the second (logic) tier available through exposing web services is to provide a more flexible way of communication. On the other hand, it safeguards the future possibility of expanding the client tier on different platforms with little effort.

The web services were exposed using Representational State Transfer (REST) that allowed communication between the client-server and the logic tier through HTTP based requests. (Rouse, 2019)

When choosing between REST and State Object Access Protocol (SOAP), along with other comparations that were made, the fact that REST has a rather simpler way of interacting with, due to using HTTP protocols, made the final decision. (Wodehouse, 2017)
Even if SOAP has stronger security (WS-Security along with SSL) and does the marshaling by default, it was more difficult to implement, and it is bound to XML rather than JSON which is more convenient.

3.2.3. Entity Framework Core

The third tier responsible for database communication and manipulation of data was implemented using .NET technology called Entity Framework Core. It is a data access technology that abstracts from working directly with the database, but rather it serves as an Object-Relational Mapper. (Microsoft, 2016)

It disposes of the need for writing data-access code, but rather allowing working with database based on .NET objects. Moreover, it enables a set of technologies called Language Integrated Queries (LINQ) that provides direct integration of queries capabilities into C#. (Microsoft, 2017)

Java Database Connectivity (JDBC) was considered, but as EF Core provided a more intuitive way of working with the data-access the decision was made towards it. (Java T Point, 2017)

3.2.4. PostgreSQL

When deciding on which database technologies to use, the scales have shifted towards PostgreSQL because of its powerful built-in features as Multi-version Concurrency Control (MVCC), parallel query plans and strong capabilities for protecting data integrity.

MySQL technology was considered, but MySQL is a purely relational database, as compared to PostgreSQL being an object relational-database, the decision was against it. (Krasimir Hristozov, 2019)

3.2.5. Communication Protocols

The communication between the first and the second tier used HTTP Protocol where data is converted into JSON format, to have a common format across the entire system. Between the second tier and the data tier, the communication is made using socket technologies that allow direct TCP communication between the two servers. The data was again converted into the JSON format, this time encapsulated into another layer that established a common agreed-upon protocol (request based) format.

3.3. Design Patterns

Even if the project is still at the early stage of development, therefore many features are not yet designed, there was still a need to solve some problems by applying common developed solutions.

3.3.1. Singleton

The singleton pattern is used in the third tier, to ensure that only one thread at the time gets access to write/read to/from the database. It needed to prevent corrupt and/or overwritten data when working with the database.

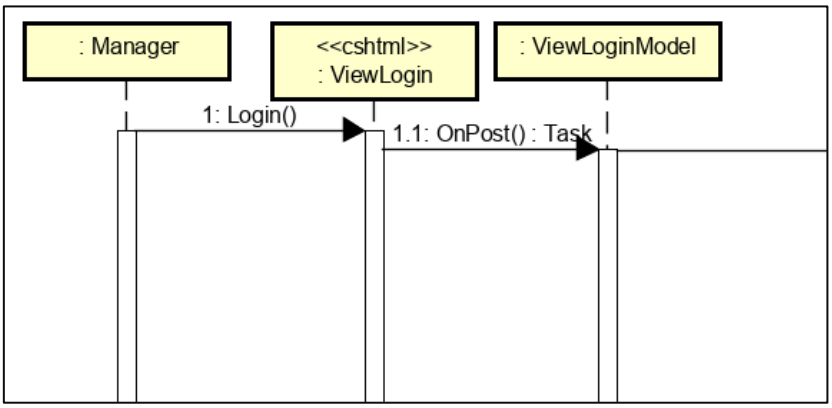
3.3.2. MVVM Pattern

The MVVM pattern is present in Razor Pages technologies, that allows wide manipulation of the views without from the models, through data-binding. The convention naming pattern needs to be respected so that the framework can successfully identify ViewModels and Models and allow communication between them.

3.4. Sequence Diagram

The sequence diagram was developed to observe and understand the flow of information through the entire system. It allowed pointing out interactions between objects of the system. The entire diagram can be found in Appendix J.

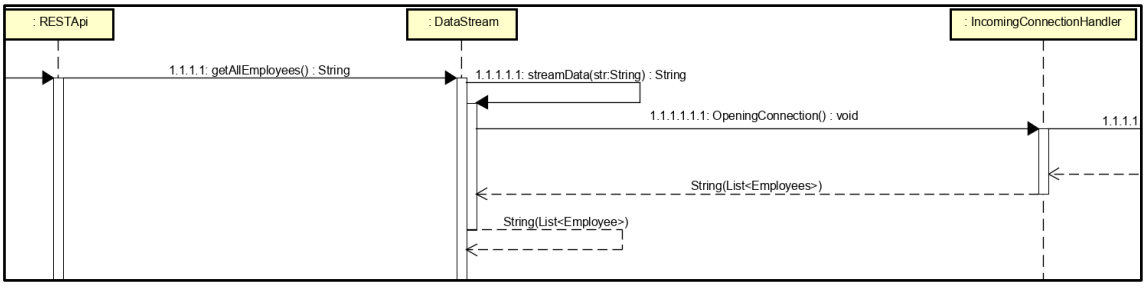
Figure 11 - Sequence Diagram - Part I



The entire process begins with *Manager's* desire to log in, after entering his credentials and submitting for login, the front-end (cshtml) is triggering post requests in the Model.

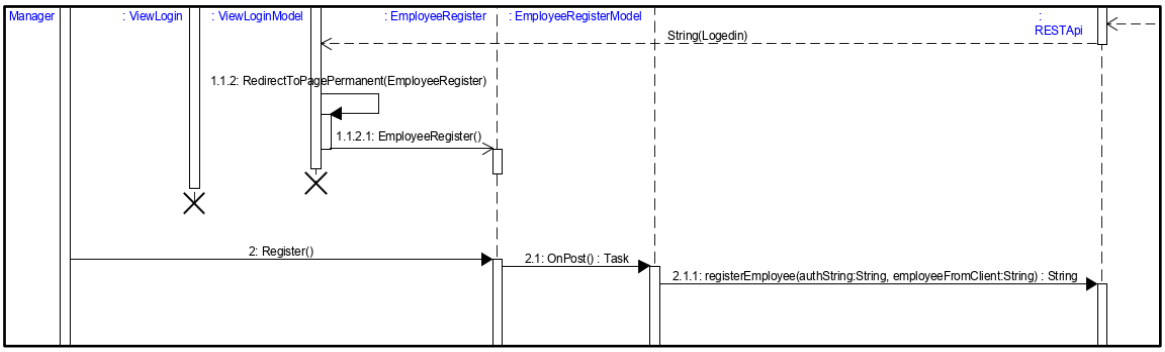
The request is sent to the second tier through an HTTP Request. The logic tier subsequently asks the data tier, through socket communication, for the current list of employees in order to do the validation.

Figure 12 - Sequence Diagram - Part II



After validating the user, the response is sent back to the Razor Page's Model, which redirects the view to a new page in order for the Manager View to perform the needed tasks.

Figure 13 - Sequence Diagram - Part III



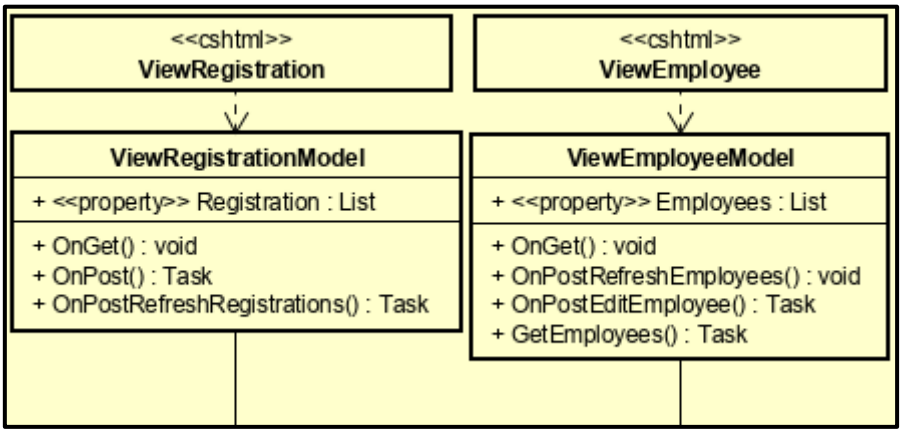
3.5. Class Diagram

Following the architecture of a distributed system, the class diagram was developed in a three-tier format to respect the imposed requirements as well as to leave space for scalability, expandability, and independence of tiers.

The class diagram is providing clear definitions so that the system can be implemented based on it. The entire diagram can be found in Appendix K. There are a few points that should be presented for a better understanding of the system.

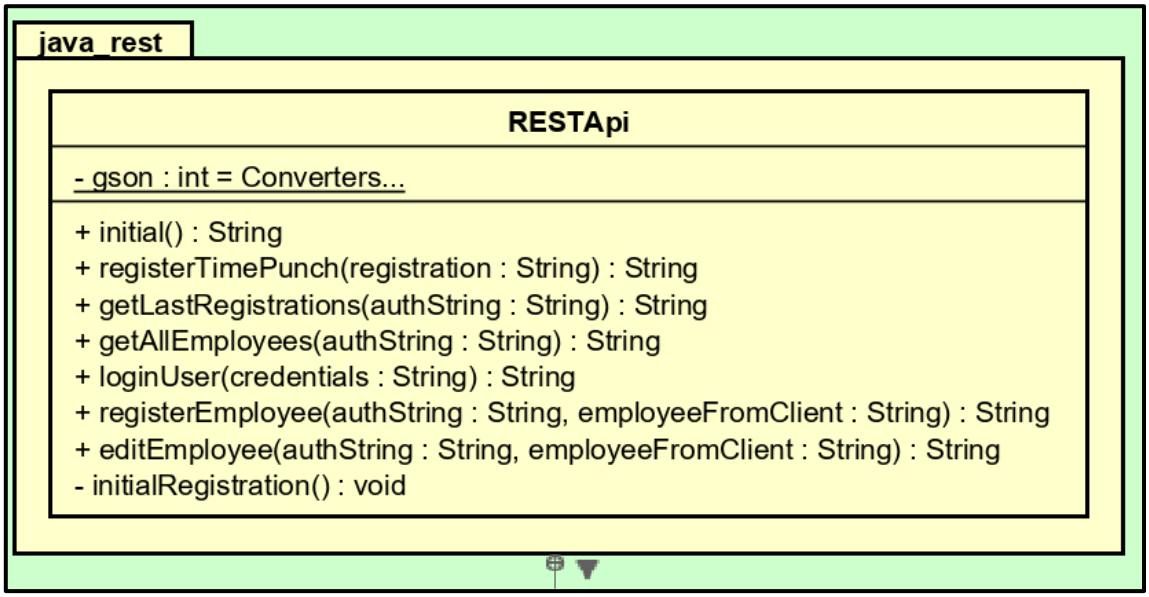
As mentioned before, the first tier utilizes the Razor Page technologies, which require certain constraints when it comes to naming patters of the classes. Below can be seen as an example of organizing the Razor pages, as well as the naming conventions.

Figure 14 - Class Diagram - Razor Pages



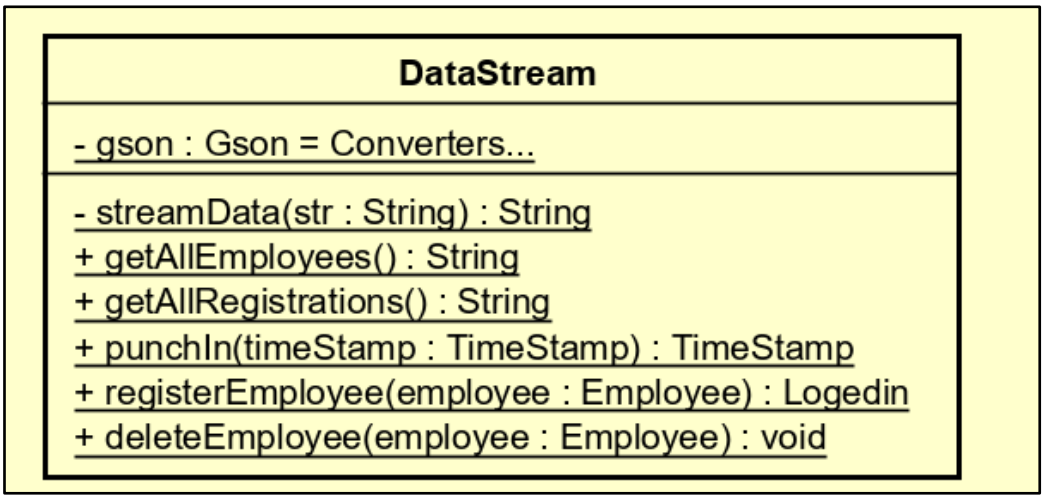
The default methods of the Model classes can be observer (*OnGet(), OnPost())* as well as some customized method handlers – *OnPostRefresEmployees();*
In the second tier, one of the most important classes that are responsible for receiving HTML Requests is the RESTApi class.

Figure 15 - Class Diagram - RESTApi



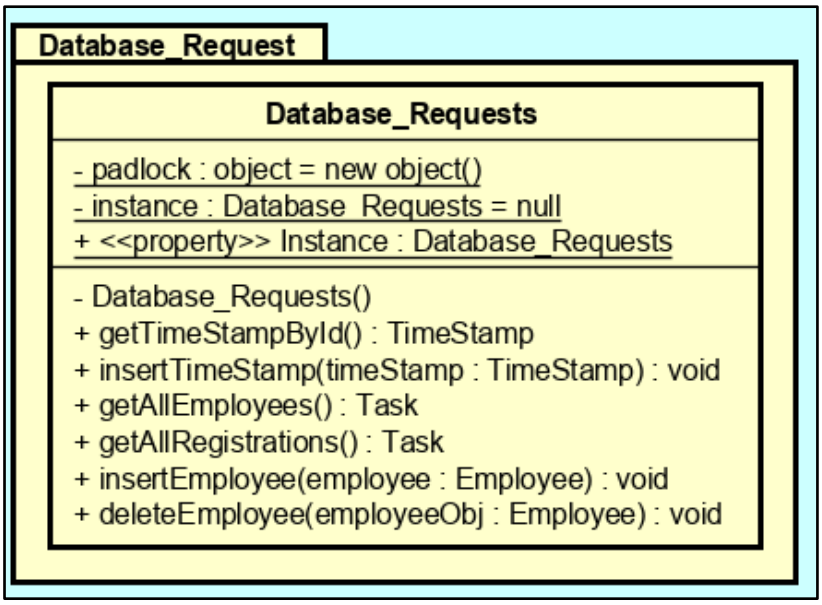
Each method when implemented will have a corresponding routing path so that it can be accessed through HTTP Requests.
Another essential class, in charge of communication between the logic tier and the data tier, is *DataStream,* it provides functionality thus does not require an instance to be created for it to operate.

Figure 16 - Class Diagram - DataStream



For the third tier, an important class to look at is the *Database_Requests* class which is responsible for providing the functionality of communication between the program and the database. The requests are called from the *IncomingConnectionHandler* and are performed by use of the database context classes.

Figure 17 - Class Diagram - Database_Requests



All in all, the class diagram provides the necessary information for a developer to be able to implement the current system. In addition, it shows how the architecture of the system is respected by splitting responsibilities accordingly between the different tiers.

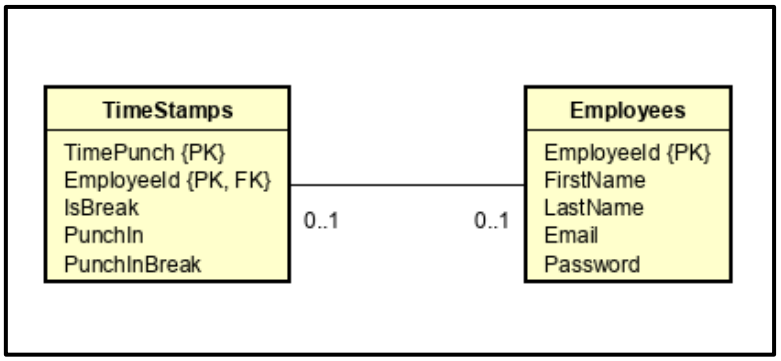
3.6. UI Design

The design of the User Interface was meant to provide a simple to navigate interface that still has all the necessary details to be able to be useful for the user. When designing the interface, the decision agreed upon is to use common web development frameworks (*Bootstrap)* that has building components. The initial sketches for the UI were improved through an iterative feedback process with the Product Owner.

3.7. Database

Due to the fact that in the third tier uses *Entity Framework* technology, the database design is made through object-mapping from .NET objects in the third tier. Moreover, due to the nature of the software, the database is rather simple. As follows:

Figure 18 - ER Diagram



3.8. Security

A few mechanisms that will prevent threats and reduce the overall risk for the system will be presented next. Some of them will be implemented in the system.

- TLS/SSL – Especially because the system is accessed through a web-based interface exposed on the web it is a priority to have a Secure Socket Layer signed by a Certificate Authority;
- Authentication protocols – is needed in communication between the client and logic tier;
- Diffie-Helman key exchanged paired with Digital Signatures – to ensure the safe key exchange and symmetric encryption;
- Hashing of the passwords;

4. Implementation

When the design phase was completed, the next step was to start implementing the system based on it. With a significant amount of the knowledge provided from the *Analysis* and *Design* the implementation part was focused more on deciding how certain designs should be implemented.

In the end, a short subchapter will describe different libraries that were used and the process of managing them.

The source code can be found in Appendix N.

4.1. Software Implementation

To be able to correlate with the details described in the previous chapter, this chapter will go through the implementation of similar processes (*Sequence Diagram)* so that clear lines can be traced from the design to implementation.

Figure 19 - ViewLogin

```
<!--Login Input Form-->
<div class="login-form">
    <form method="post">
        <h2 class="text-center">Log in</h2>
        <div class="form-group">
            <input name="EmployeeId" type="text" class="form-control" minlength="4" maxlength="4" pattern="\d*" placeholder="Username" required="required">
        </div>
        <div class="form-group">
            <input name="Password" type="password" class="form-control" placeholder="Password" minlength="6" required="required">
        </div>
        <div class="form-group">
            <button type="submit" class="btn btn-primary btn-block">Log in</button>
        </div>
        <div class="clearfix">
            <label class="pull-left checkbox-inline"><input type="checkbox"> Remember me</label>
            <a href="#" class="pull-right">Forgot Password?</a>
        </div>
        @if (Model.Display)
        {
            <div class="alert alert-danger" role="alert">
                Wrong Credentials
            </div>
        }
    </form>
</div>
```

The main section of the *ViewLogin* implementation is the login form, that uses the Razor's build-in recognition of post requests described in the upper section *<form method = "post">*. The inputs have a basic form of validation provided by the functionalities of HTML5.

Figure 20 - ViewLoginModel

```
public async Task<IActionResult> OnPost()
{
    var client = new HttpClient();

    //Login Information retrieval
    Login = Request.Form["EmployeeId"];
    Password = Request.Form["Password"];

    //Login post request
    HttpResponseMessage response = await client.PostAsync("https://www.api.el-aasi.com/api/thehappypig/login"
    var responseContent = await response.Content.ReadAsStringAsync();

    //Checking the success state
    logedin = JsonConvert.DeserializeObject<Logedin>(responseContent);

    if (logedin.logedin)
    {
        return RedirectToPagePermanent("/Manager/ViewRegistration");
    }
    else
    {
        Display = true;
        return Page();
    }
```

The backend control of the form is done through a *PageModel,* that retrieves the credential information using *Request.Form* method and sends a post request to the second tier with the information.

Figure 21 - RESTApi - Login

```
@POST
@Path("/login")
@Produces(MediaType.APPLICATION_JSON)
public String loginUser(String credentials)
{
    //Creating the "credentials employee"
    Employee employee = gson.fromJson(credentials, Employee.class);

    ArrayList<Employee> empployeeList = DataStream.getAllEmployees();

    //Returns a false or true (in a LogedIn Object for better serialization) for whether the client was logged or not
    return gson.toJson(new Logedin(Checks.checkCredentials(empployeeList, employee)));
}
```

The post request is taken over by the *loginUser()* method which deserializes the body of the request, retrieves the current list of the employee and checks if the credentials are valid for the user, sending back a custom Boolean object. The validation is done through function classes, that use hashed passwords.

The socket connection with the third tier is done using the *DataStream* class presented below.

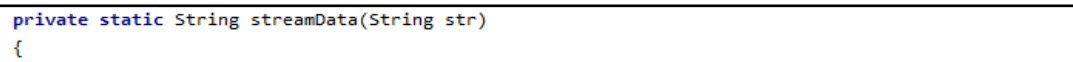


Figure 22 - DataStream

```
private static String streamData(String str)
{
    Socket socket = null;
    String clientIn = null;

    try
    {
        //!!!NEEDS FUNCTIONALITY FOR DYNAMICAL CHANGING THE IP AND PORT Marcel @version 4.5.1.
        socket = new Socket( host: "35.246.226.193",  port: 5000);

        Charset charset = Charset.forName("ASCII");

        final OutputStream outToServer = socket.getOutputStream();
        final DataOutputStream out = new DataOutputStream(outToServer);
        out.write(str.getBytes(charset));

        final InputStream inFromServer = socket.getInputStream();
        final BufferedReader input = new BufferedReader(new InputStreamReader(inFromServer, charset));

        clientIn = input.readLine();

        socket.close();
```

The class uses a basic TCP socket which sends data on a given IP address and port number.

On the other end, data tier, a TCP Client listens on the given port, accepting incoming data from the logic tier.

Figure 23 - OpeningConnection

```
public async void OpeningConnection()
{
    Console.WriteLine("Waiting for clients");
    TcpClient client = listener.AcceptTcpClient();

    Console.WriteLine("Client Accepted");
    NetworkStream stream = client.GetStream();

    //Receiving the request
    byte[] bytesFromBusinessLogic = new byte[1024];
    int bytesRead = stream.Read(bytesFromBusinessLogic, 0, bytesFromBusinessLogic.Length);
    string request = Encoding.ASCII.GetString(bytesFromBusinessLogic, 0, bytesRead);

    //Checking the requests
    var requestFromClient = JsonConvert.DeserializeObject<Request>(request);
    Console.WriteLine("Reading the request");
```

Based on the request, the class uses different methods provided by the *Database_Requests* class that through context classes performs CRUD actions on the database.

Figure 24 - Database_Requests - getAllEmployees()

```
1 reference
public async Task<List<Employee>> getAllEmployees()
{
    return await context.Employees.Select(p => new Employee(p.FirstName, p.LastName, p.Email, p.EmployeeId, p.Password)).ToListAsync();
}
```

4.2. Dependencies

When it comes to using external libraries and building dependencies for the project, the two technologies (.NET and Java) have different approaches in solving them.

4.2.1. .NET - NuGet

For .NET the dependencies were managed through a free open-source packet manager called *NuGet* through which the dependencies were found and added to the project's external library.

Some of the used packages are:

- Newtsoft – Json.Net 3.0 package – for Json Serialization/Deserialization;
- Npsql – PostgreSQL 3.0 package – PostgreSQL Data Provider;

4.2.2. Java - Maven

When it comes to managing packages the best solution for Java was to use Maven packages builder which was operated through the pom.xml file where the dependencies were added, afterward imported into the project.

Some of the used packages are:

GSON – Google's Json Serializer 2.8.6

JODA – Joda Time for having similar to C#'s DateTime's objects 2.10.5;

FatBoy – Gson-Joda added functionality for serialization of the DateTime Objects 1.8.0;

5. Test

At the end of the project, testing of what was currently implemented was made, to observe how useable is the system (*Usability Testing)* from a UI point. To observe the general functionality of the system, the test cases developed previously were used.

5.1. Black Box Testing

In this section will be described tests made while treating the system as a *black box*, usually, the testers are not experts in the field, in order to provide impartiality for the testing process.

5.1.1. Test Cases – Process and Results

In the *Analysis* chapter, some test cases were put in place to determine how well the software is functioning according to the described Use Cases and Requirements. All the results of the test cases can be found in Appendix G.

Table 6 - Test Case Results

19.	Manager leave the input empty when editing a registration	Check if an informing error message is displayed	Not implemented
20.	Manager registers a new Employee	Check if the new employee's information is displayed in all employee view	Functioning
21.	Managers adds an employee with the same Employee Id as an existing one	Check if an informing error message is displayed	Partially functioning

It can be seen that generally, the results of the Test Cases are positive, most of the cases providing the expected result (as described in the *Reaction* section). There were marked as *Partially Functioning* which meant that either to have the desired result there was a need for more effort, or the result was only partially achieved.

In the cases of *Not Implemented* meant that the functionality was totally absent or unavailable giving no possibility to observe the reaction of the test cases.

Test Cases provided great insight into the general stage of the currently developed features for the system. In addition, during the design and develop test cases (along with other results from the provided analyses) offered a guiding point. Sometimes for design

or/and implementation, a partial *Test-Driven* software development process was followed.

5.1.2. Usability Testing – Process, Results, and Critique

To test how well the future users will be able to use the system, as well as to provide some constructive feedback for future development, *Usability Testing* was made on the currently implemented features.

The usability testing consisted of 5 non-expert individuals follow some instructions described in the scenarios that also specified what was needed to achieve. The usability testing, along with testing the functionality of the system, provides great insights on how the user interface is built. Some of the scenarios were a continuation of the previous one, meaning that the experience/information gained in the first scenario can be used in the next one.

There was a passive spectator, that observed how well the users were able to follow the given instructions, as well as a short discussion after each test to gain some more personal (qualitative) opinions.
All of the scenarios can be found in Appendix P.

Figure 25 - Scenario Testing Example

Scenario **Punch in work time**

You are a facility employee of The Happy Pig Company. You just came at work, want to register for work time and then go at your work place, having the following information (not all information is required to be used, you may add or modify the information as you wish) as follows:

- *First and Last name – John Doe;*
- *Email – john@thehappypig.com;*
- *Employee Number – 1111;*
- *Password – changeit;*
- *Work Station – Packaging;*

Please punch in for work.

The results of the testing were generally positive, some of the respondents needed a bit more time to accommodate with the general feel of the software. All the results of the Usability Testing can be found in Appendix Q.

Table 7 - Usability Test Results

SCENARIO	RESULTS	ADDITIONAL COMMENTS
PUNCH IN WORK TIME	3 – Successful 1 – Neutral 1 – Partially successful	All of the respondents managed to complete the task. Some of the respondents were confused that they needed to first check the punch/break. One respondent was looking for specific "punch in";

5.1.3. White Box Test

Testing of the code was provided in a more unstructured way due to the nature of the code. The fact that there are not very complex classes that perform complex tasks formal testing (Unit Testing) was not necessary and only a small part of the software would have been possible to test. Because of that, it was decided for a more unformal test process of the code that includes:

- Checking of the inputs/outputs;
- Logic check;
- Checking the SOLID Principles;

6. Results and Discussion

The current state of the software can be classified as incomplete. Even if the software provides the necessary functionality for having a proof of concept, it still lacks many base functionalities that will classify it at least a minimum viable product.

Currently, the software is able the successfully communicate between the tiers, even when deployed on separate machines and running on different types of server technologies:

- Client Tier – Razor Pages – using a Kestrel with a reverse proxy to Apache (for limiting the exposed public surface area and simplifying the secure communication configuration);
- Logic Tier – Java Logic – running as a Java Servlet on a Tomcat Server;
- Data Tier – running as .NET console application;

The communication between the servers is secured with an SSL certificate provided by *Let's Encrypt*.

At this point in development, the software is able to provide registration of both work and break time based on the registered employees in the system. The time registration functionality provides logics for determining whether the user is punching in/out or registering to/from break time. A user guide can be found in Appendix R.

On the administration side, the software provides possibilities for logging into a *Manager Dashboard* where the managers can add/edit employees, see the currently registered employees, as well as see the timestamps.

The logic tier provides the capacity of securing the password through a hashing process that uses *Password-Based Key Derivation Function 2 With Hash-based Message Authentication Code SHA1* Algorithm.

One last point to mention is that the data is stored into a PostgreSQL database through the data tier that is build using *Entity Framework* technology.

7. Conclusions

To understand the progress of this project, it is important to look back at the problems described for it, in the Project Description.

The main problem stated:

The time logging system used by The Happy Pig company is inefficient, prone to error, hard to store/access/maintain and easy to cheat, thus leading to a variety of negative outcomes for the company.

Currently, the system provides a more efficient way, that is less prone to errors and much easier to store/access/maintain the system for registering work and break time. Even at this stage, the system presents advantages over what the company is working with right now.

On the other hand, the system still needs many improvements to make it usable in a real work environment. These improvements will be described in the next chapter.

Next, it is important to observe how well are the sub-problems answered through the project.

- *What are the different access levels for the information to which type of data?*

When considering the different access levels for information a login system was implemented, and at this stage, only managers have the right to retrieve any kind of data. With this in mind, it can be stated that there are two levels of access to information:

Employees – no access;

Managers – access to all information;

- *How should different type of workers (fix vs flexible schedule, facility vs remote location, full-time vs part-time) log their work-time?*

It was decided that all type of employees will log their time in a unified way through a web-based application.

- *How and where should the information be stored, accessed and retrieved so that it is private and secure?*

The information is stored in a database, with the very sensitive information – password – not being stored in original but hashed. The connection between the tiers is encrypted with an SSL thus improving the security of the communication.

- *Which type of reports or statistics must be created for different end users?*

Currently, the system provides only an overview of all the registered employees in the system as well as a view of the time registrations made by the employees. Even if limited it offers the minimum amount of information needed for the manager. Other types of statistical reports were less prioritized in the requirements, therefore, is left for future development.

- *What information in regard to employees, time and statistics should be stored?*

To provide the base functionality for the system, personal information about the employees is stored in the system (First and Last name, e-mail, employee id, etc.) as well information in regards to registration (start/finish work/break).

- *What are the different types of exceptions and how should the exceptions be handled (e.g. overtime, human error)?*

The system offers a small amount for handling exception, in regards to making registrations:

- Cannot register break without being punched in;
- Cannot register finish work, while on break;
- Automatic registration for start/finish based on the previous state;
- Checking for empty fields;

There is still space for more improvements in this direction that will be addressed in the next chapter.

- *What measures can be made to prevent and/or identify attempts to cheat the system?*

Unfortunately, due to the nature of the system, currently, there is no functionality satisfying this problem.

To conclude this project, a line through the project will be drawn. The project began with the Project Description, where the background and the main problems were stated. Together with the Product Owner a series of requirements were defined and prioritized, from there Use Cases were developed that serves as a base for the Activity Diagram, culminating with the Domain Model as a finale for the Analysis.

In the Design phase, the architecture for the system was established, following a Sequence and Class diagram which server as the foundation for the implementation.
The project overall does not satisfy all the requirements, but as proven in testing, the ones that were fulfilled are functioning as intended.

8. Project future

As mentioned before through the project, currently the software provides the base functionality for the system, therefore many features are left to be implemented in the future.

First, it is a priority to develop and implement a feature that will satisfy all the requirements that were presented at the beginning of this project.

Other possibilities can include an Android/iOS app that will give employees an overview of their working time as well as managers a more flexible way of administration. As the logic tier is built as an API expanding on the client side does not provide many difficulties.

Having an application will allow the usage of NFC technologies present in many of today's smartphone, thus providing an even easier way of registering the time. Otherwise other kind of NFC technologies could be implemented to facilitate employees in their time registration.

As the biometric technologies advanced, using a biometric terminal (Fingerprint) will improve the security of the system as well as diminish the cases of cheating of the time registration.

Last but not least, a communication protocol could be made to make it easier to port data from the system to an accounting system, to reduce the amount of work of the managers as well as the accountants.

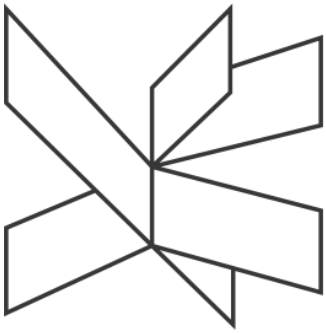
9. Source of information

- Altexsoft, 2019. *What is SOAP: Formats, Protocols, Message Structure, and How SOAP is Different from REST.* [Online]
  Available at: altexsoft.com/blog/engineering/what-is-soap-formats-protocols-message-structure-and-how-soap-is-different-from-rest/
  [Accessed 2019].
- Crain, A., 2002. *The Relational Edge.* [Online]
  Available at: http://www.therationaledge.com/content/jun_02/t_drUseCase_ac.jsp
  [Accessed 2019].
- Cristancho, M., 2019. *ECM.* [Online]
  Available at: https://theecommmanager.com/smart-requirements/
  [Accessed 2019].
- DAFC, D. A. a. F. C., 2019. *Danish Pig Meat Industry.* [Online]
  Available at: https://agricultureandfood.dk/danish-agriculture-and-food/danish-pig-meat-industry
- Geogre Coulouris, J. D. T. K. G. B., n.d. *Distributed Systems Concepts and Design.* Fifth ed. s.l.:Addison-Wesley.
- GWT, 2019. *Overview - GWT.* [Online]
  Available at: http://www.gwtproject.org/overview.html
  [Accessed 2019].
- Java T Point, 2017. *Java JDBC.* [Online]
  Available at: https://www.javatpoint.com/java-jdbc
  [Accessed 2019].
- Krasimir Hristozov, 2019. *MySql vs PostgreSQL -- Choose the Right Database for Your Project.* [Online]
  Available at: https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres
  [Accessed 2019].
- Larman, C., 2004. *Applying UML and Patterns: An Introduction to Object-Orientated Analysis and Design and Iterative Development.* Third ed. s.l.:Addison Wesley Professional.
- Marketline, A. I., 2019. *Beyond Meat, Vegan Burgers are going Public,* s.l.: MarketLine.
- Microsoft, 2016. *Entity Framework Core.* [Online]
  Available at: https://docs.microsoft.com/en-us/ef/core/
  [Accessed 2019].

- Microsoft, 2017. *Language Integrated Query (LINQ).* [Online]
  Available at: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/
  [Accessed 2019].
- Microsoft, 2019. *Get started with ASP.NET Core Blazor.* [Online]
  Available at: https://docs.microsoft.com/ro-ro/aspnet/core/blazor/get-started?view=aspnetcore-3.1&tabs=visual-studio
  [Accessed 2019].
- Microsoft, 2019. *Introduction to ASP.NET Web Programming Using the Razor Syntax (C#).* [Online]
  Available at: https://docs.microsoft.com/en-us/aspnet/web-pages/overview/getting-started/introducing-razor-syntax-c
  [Accessed 2019].
- Microsoft, 2019. *Introduction to Razor Pages in ASP.NET Core.* [Online]
  Available at: https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-3.1&tabs=visual-studio
  [Accessed 2019].
- Mintel, 2018. *It's getting Vegan in here.* [Online]
  Available at: https://downloads.mintel.com/private/zgXKu/files/688043/
- NISO, 2010. *Scientific and Technical Reports -,* Baltimore: National Information Standards Oganization.
- Radigan, D., 2019. *www.atlassian.com.* [Online]
  Available at: https://www.atlassian.com/agile/kanban
  [Accessed 2019].
- Rehkopf, M., 2019. *Agile Coach: Kanban vs Scrum.* [Online]
  Available at: https://www.atlassian.com/agile/kanban/kanban-vs-scrum
  [Accessed 2019].
- Rouse, M., 2019. *RESTful API (REST API).* [Online]
  Available at: https://searchapparchitecture.techtarget.com/definition/RESTful-API
  [Accessed 2019].
- sestek., 2019. *advantages and disadvantaages of biometric authentication.* [Online]
  Available at: https://www.sestek.com/2016/11/advantages-disadvantages-biometric-authentication/

- Soco Tax, 2019. *THE MAJORITY OF SMALL BUSINESSES STRUGGLE WITH TIME TRACKING ERRORS.* [Online]
  Available at: https://socotax.com/majority-small-businesses-struggle-time-tracking-errors/
- TeaVM, 2019. *Docs - TeaVM.* [Online]
  Available at: http://teavm.org/docs/intro/overview.html
  [Accessed 2019].
- thepigsite.com, 2019. *Significant fall in the Danish pig population.* [Online]
  Available at: https://thepigsite.com/news/2019/05/significant-decline-in-the-danish-pig-population
- TSheets, 2017. *The Ultimate List of Time and Attendance Statistics.* [Online]
  Available at: https://www.tsheets.com/resources/time-attendance-stats
- VIA Engineering, in preparation. *Confidential Student Reports,* s.l.: s.n.
- VIA University College, 2019. *Semester Project: Heterogeneus System (IT-SEP3).* [Online]
  Available at: https://studienet.via.dk/sites/uddannelse/ict/Horsens/studymaterial/Pages/Course-Descriptions.aspx
  [Accessed September 2019].
- volkerdon, 2019. *Volkerdon.* [Online]
  Available at: https://www.volkerdon.com/pages/moscow-prioritisation
  [Accessed 2019].
- Wodehouse, C., 2017. *SOAP vs. REST: A Look at Two Different API Styles.* [Online]
  Available at: https://www.upwork.com/hiring/development/soap-vs-rest-comparing-two-apis/
  [Accessed 2019].

Appendices



VIA University College

Semester Project: Heterogeneous System

Process Report – The Happy Pig Company Time Logging System

Supervisors:

Jakob Knop Rasmussen

Jan Munch Pedersen

Students:

Gais El-AAsi – 279910

Marcel Notenboom – 279963

Number of characters incl. spaces, images and footnotes 27,186 characters

Third Semester: Software Engineering

VIA University College – Horsens Campus 20th of December 2019

Table of content

List of Figures and Tables

1. Introduction

The project was introduced to us at the begging of the semester on the 4th of September, where we were provided with the information on what is expected from this project, milestones and formal requirements. On the 16th of September after forming after deliberation for a few different ideas, we decided that for this project we will be working with a time logging system.

Next, different milestones followed up, including defining the project description, preparing the basic requirement of the future software and proving that the product is viable. The Software Development of Distributed Systems (SDJ3) class provided us with valuable knowledge needed to meet those milestones in regards to system architecture.

We decided that the best software development framework for us would be Kanban (Radigan, 2019) that allowed a good structured and organized way of developing the project, at the same time, less pushing and complicated than SCRUM (Rehkopf, 2019).

During the Semester, we would work for the project on the assigned days – Wednesdays, though to keep up with the milestones, and ensure that our wishes in regards to the project are satisfied, more work needed to be put in. Therefore, many times we used other days as well to work on the project.

Being a heterogeneous system, .NET development classes (DNP) provided us with great resources of understanding the .NET platform, and how to utilize it to our advantage in the semester project. Many tools and technologies ending up to be crucial for the functioning of the system.

Supervision was carried on during the entire period, many times teachers assisting us in our semester project even during their other classes (DNP and SDJ3), this helped us stay on track and ensure that we are working our best for this project.

2. Group Description

For the new project, all the members of the group have worked together in the previous semesters. This fact helped us find a common language much easier, and therefore many of the hops encountered in the previous project were successfully avoided.

2.1. Group Contract

Even if in the last semester we have all worked together, this semester the Group Contract was completely reworked. The contract was redone, to ensure that it will become more liberal, to encourage and engage all the members of the group into productive communication. It still provided the necessary structure, rules, and work-ethics to ensure that there was a guideline to follow for the members. The group contract was very much inspired by Valve's *Handbook for New Employees* (Valve Corporation, 2012). The group contract can be found in Appendix A.

2.2. Project Experience

The fact that all members have had experience in working in this kind of projects as well as the fact that we have had experience working with each other, helped us establish common unspoken work rules. Moreover, it helped us gain a common understanding of what outcome we expect from this project, how much work is needed to be put into the project and how dedicated should everyone be.

2.3. Group Roles

As stated in the *Group Contract's Section 3 Management, Article 3.3* there are no official roles for any member of the group, though, in case of different roles naturally emerging during the project, the roles were more than welcomed as long as it contributed to the well-being of the project. The roles were not fixed or/and imposed, in addition, they could morph through the project. With that in mind there were certain responsibilities that the group members took over to ensure that the project was developing as intended:

- Levi was responsible for ensuring that the team stays on track with what are the formal requirements of the project. He was as well a motivator and support member of the team many times ensuring that there are no obstacles (not

necessarily related to the project) that could harm the progress of the semester project;

- Marcel had the responsibility with quality control, and critique thinking, his assessment of the overall work many times helped us spot issues that would have occurred in the future. He was in charge of the overall direction of the design and implementation;
- Gais was responsible with the management of the work and team, many times acting as a proxy for the team when communication with supervisors, as well as a role of coordination and work distribution;

In general, we have had a great collaboration for this project, with occasional obstacles that were easily solved by communicating and expressing our concerns.

2.4. Member dismissal

As stated in the Group Contract *Section 4 Conflict resolution and member dismissal Article 4.1* a member cannot be dismissed by the group, only in case when the member himself/herself has the decision to leave. In our case, this was the situation when during the period after the Project Description was approved, Levi had to inform the team that he will not be able to continue the work with us because of personal reasons.

After that, the expectation for the project outcome was reworked with the remaining members, as well as the work amount, which was redistributed to ensure that all the milestones were hit.

3. Project Initiation

After we were introduced to the semester project, the first milestone to hit was to decide on a project topic. For finding out what we wanted to work with for the next months, it was decided that every member was allowed to make two proposals. With those in mind, 3 of them were democratically picked and chosen to be elaborated. The proposals can be found in Appendix B.

After a meeting with the supervisors, another voting was made, the decision being for a time logging system. The arguments for this proposal were:

- Can easily be implemented in a distributed system with three tiers;

- Did not have much unnecessary overhead that did not contribute to the development of the semester project or learning process;
- Was realistic – meaning that it was achievable – which can be a great motivator during the semester period;

4. Project Description

As our first main milestone for the project, we had to make a Project Description. The project description had the role to provide a more structured way of viewing what we wanted to do. Some of the details that previously were not worked out were put in place, and after finishing it, we could see a clearer image of what is the problem that we wanted to solve, what we will not focus on, what tools we will use, the schedule, the risks, etc.

The first draft of the Project Description had some issues in the Time Schedule chapter, as we were both overestimating and underestimating some of the milestones that needed to be achieved. Other than that, the project description has been a guiding tool through the entire project development. It helped us stay focused on what was needed to be done. Project Description can be found in Appendix C.

5. Project Execution

During the project execution period, this semester we took the decision to use Kanban Software Development Framework as it fitted better our team and style of work.

5.1. Kanban

Kanban is a Software Development Framework, very similar to SCRUM but with a few differences that make it great for small teams that prefer to work in a less structured way.

5.1.1. Reasoning

The main reasons that made us try Kanban as our framework are:

- A rather small group – 3 people initially, 2 during most of the time;
- Goal-based tasks – the tasks did not have a particular size (hours, points, etc.) rather they were determined by a clear goal;

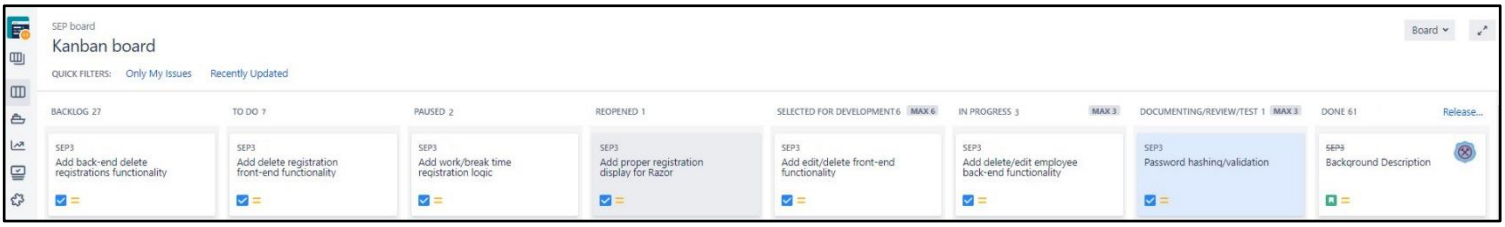
- No roles – there were not any kind of roles needed (ex. SCRUM Master, etc.), the focus was that everyone was contributing to the management of the team, through proper management of themselves;
- Less strict iterations – an iteration ended when a substantial amount of value was added to the system;
- Less amount of work in the management and more into work;

5.1.2. Kanban board

To ease the planning and usage of the Kanban framework *Jira Software* from Atlassian was used as the main Kanban dashboard. The board was made of 8 sections as follows:

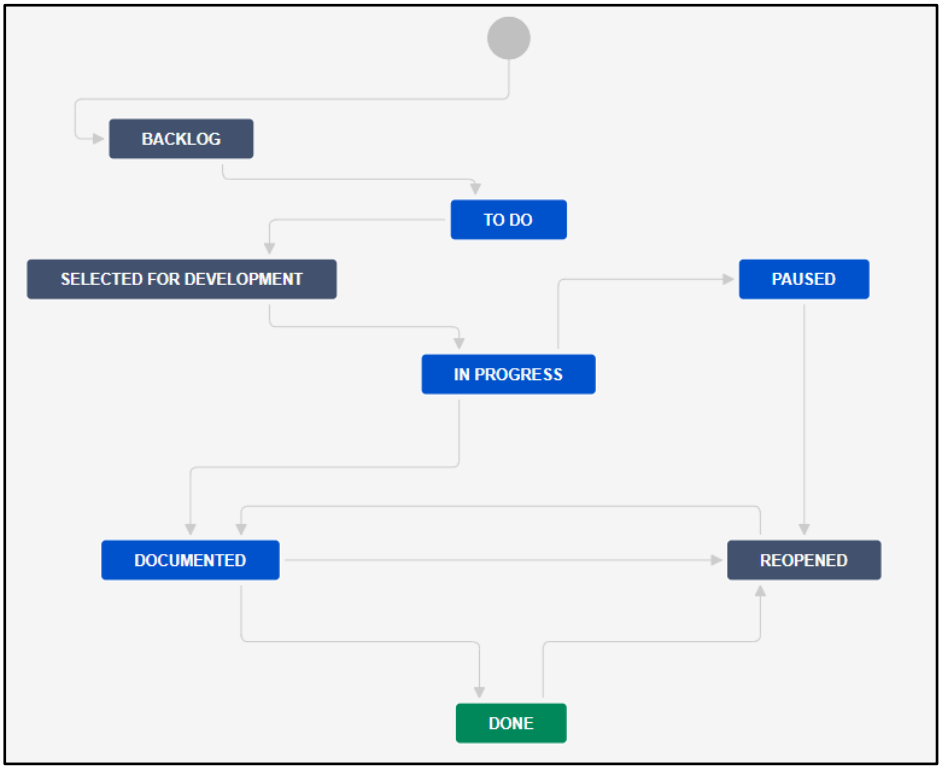
- Backlog;
- To do;
- Paused;
- Reopened;
- Selected for development – maximum 6 tasks;
- In progress – maximum 3 tasks;
- Documenting/Review/Testing – maximum 3 tasks;
- Done;

Figure 1 - Kanban Board



A certain workflow was established, to ensure that a proper flow of the tasks is in place as presented in the figure below.

Figure 2 - Task flow



Jira Software helped us a lot in better organizing our work, with a little setup at the beginning of the project, we used it throughout the entire period.

5.1.3. Backlog planning

A planning session was made every time a low number of tasks (2 or less) were left in the backlog of the Kanban board. The meeting was either online or physical (depending on the situation) where we will take next inline user stories and would split them into more manageable tasks that were added to the backlog.

Sometimes, when a task turned to be more complex than expected, it would be split into smaller subtasks, so that it can be more achievable in a shorter period, to make it easier to track the progress and test.

5.1.4. Iterations

An iteration was considered complete when all of the tasks that were obtained from a single user story were completed. It was important after each iteration to "stop" and assess our progress, look retrospectively and try to draw some critique.

Because the iterations were not time-bounded, but rather task bounded, we did not have a fixed schedule for them, and would usually have the meeting on the next day, giving us some time to individually reflects on the "goods" and "bads" of the iteration.

5.1.5. WBS and WIP

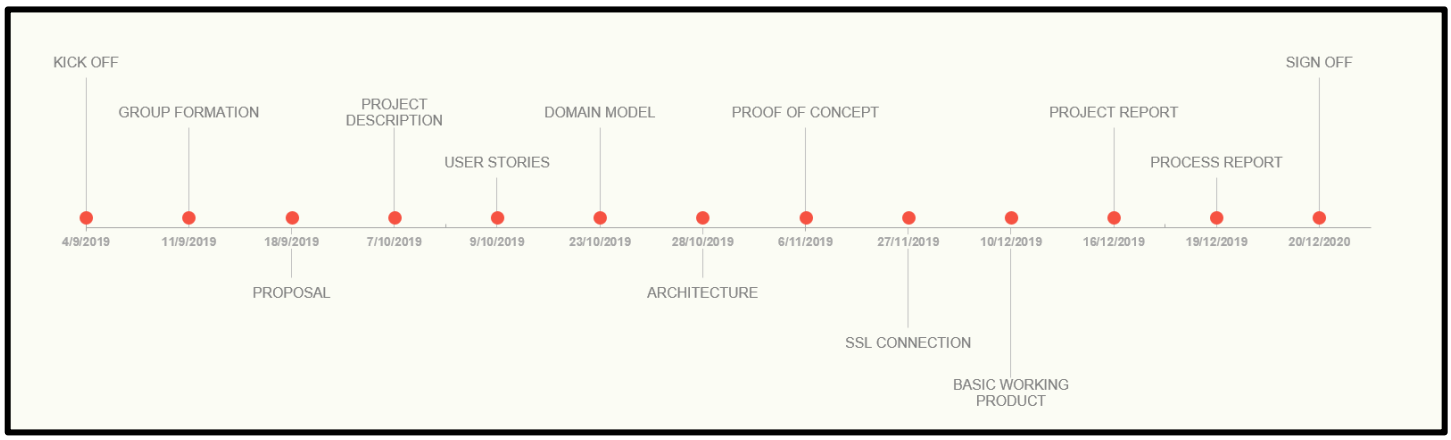
Work breakdown structure principle helped us manage even difficult tasks, by splitting them into smaller and smaller pieces and solving them one by one. Even if, one could say that it should be natural to do so, sometimes having a structure for splitting tasks is a great tool for handling hard to finish tasks.

Work in progress principle limited the number of tasks that we were allowed to work at the same time (In progress column from the Kanban) to 3 tasks at the time. This helped us to guarantee that each task is completed before the next one is started.

5.1.6. Time Schedule

To ensure that we stay on track a time schedule was made with the main milestones that were made in the Project Description along with others developed during the project.

Figure 3 - Schedule



Along with the milestones, a time tracking table/chart was made, *to keep an eye,* on how the work was progressing. The hours were updated at each milestone and it offered a good overview of how much work is needed, how much time is left, what is the next milestone, etc. The Excel file can be found in Appendix D.

Figure 4 - Time reference table

Milestone		Days until submit	Hours Worked (total)	Hours Left	Hour left / Member	User Storie Completed	Milestone hit
Name	Date						
Kickoff	4-Sep	107	0	825	275	0	Yes
Group Formation	11-Sep	100	24	801	267	0	Yes
Proposal	18-Sep	93	24	777	259	0	Yes
Project Description	7-Oct	74	96	681	227	0	Yes
User Stories	9-Oct	72	16	665	222	0	Yes
Member Dismissal	10-Oct	71	4	470	157	0	
Domain Model	23-Oct	58	48	422	141	1	Yes
Architecture	28-Oct	53	16	406	135	3	Yes
Proof of concept	6-Nov	44	48	358	119	3	Yes
SSL Connection	27-Nov	23	96	262	87	5	Yes
Basic Working Proudct	5-Dec	15	64	198	66	6	Yes
Improved working product	10-Dec	10	64	134	45	7	No
Project Report	16-Dec	4	80	54	18	8	Yes
Process Report	19-Dec	1	48	6	2	8	Yes
Submit	20-Dec	0	8	-2	-1	8	Yes
Total (remaining members)			588			18	

We were keeping track of how many days are left until the submission date to ensure that know we are on the timeline. Having milestones helped us ensure that we end up with the base requirements (for the project, not software) completed. Hours left was giving an approximation of how much work is needed to be put as well as how much was done (motivational).

The *Member Dismissal* milestone, of course, was not planned and later introduced in the table, as it was important because we needed to rework the hours, to ensure that we are working what it is expecting from us.

5.1.7. Critique of Kanban Framework

Working with Kanban Framework was interesting and quite different from what we were used to with SCRUM. Because mostly we worked only two members it felt nice not to have all the responsibilities that involve doing SCRUM and it felt a bit more natural.

On the other hand, because last project working with SCRUM, felt like a *sprint* we were afraid that not having a constant push would make us *slack*. But it turned out to be the other way around. Because there were not clear achievements for each iteration, we constantly had the sensation that we are falling behind. Because of that sensation, we constantly were rushing, which was quite exhausting, burning us up in the end.

All in all, we feel like the best option for a development framework would be a combination of those two (SCUM and Kanban) which we will experiment in the next project.

5.2. Unified Process

The unified process helped us ensure that we added a fully developed feature to the software. Working in an iterative way, provide many benefits (easier to adjust to the needs, harder to lose focus, easier to predict) and helps stay agile in the software development process

An iteration would go through all the stages (Elaboration, Construction, Transition) providing a finished, tested and documented feature in the end.

5.3. Version Control

Even if last semester we touched upon using version control, this semester we tried to use it from the beginning and until the end. It helped us a lot to be able to share everything between us without having constantly send files and compare versions. At the same time, it was very useful when adding new features that would break something to be able to go back and understand what has broken.

All in all, it was a very useful tool, that provided great support for our software development process. One problem was when we tried using *branching*. Because it is a

rather small project, using branching was an *overkill* meaning that it caused more problems than solved. But still, it was great to gain experience of the system behind.

5.4. Critique of the project

There are a few points that are not good as we wished in regards to this project, point that we will try to keep in mind and not repeat in the next projects. Because of the learning process, it is a priority to mention them to ensure that they are documented and can be learned from.

5.4.1. SOLID Principles

Because a lot of importance was paid to the architecture of the software, ensuring that it respects the requirements of a distributed system, we paid less attention to the SOLID principles. As a consequence, the system is much more fragile, closed and immobile than we would have wanted it to be, at the end affecting the software's possibility for scalability and expansion. It was our mistake, not paying more attention to the principles when designing the software and rather focusing on the overall architecture which in the end is negatively affected by us violating the SOLID Principles

5.4.2. User Stories

In the beginning, phase, when working on the user stories, we expected to complete at least 2/3 of them. But as we progressed through the project, we realized that we will never be able to reach that amount. In addition, a member of the group had to leave, which meant that even less would be done. We wanted this project to be more round, in the end, providing at least a glance of a solution that could actually be used. But due to the time needed to analyze and design, as well as implement the skeleton, we fell short.

5.5. Group Reflections

All in all, we enjoyed this project better compared to the last semester, because of the fact that it felt much closer to a *real project* due to its nature (Heterogeneous System). Working with different technology options for the problems, made our critical and engineering thinking to improve (compared to when one has only one option). Deciding

on what to use in order to solve a certain problem, and how to organize everything so that it made sense was, enjoyable.

Moreover, this semester not having a class that was solely responsible with how the semester project should be made (Software Engineering Classes of the last semesters) gave us a freer hand on deciding how to work, which felt a bit, *unsafe* through the project, but was a great learning tool for us.

6. Personal Reflections

In the next chapter personal reflection on the project and process around it will be presented.

6.1. Gais El-AAsi

The third semester was a good one. I particularly loved the semester project *Heterogeneous System* as it was quite a step forward form the last semester. Having the possibility to work with different technologies, and try to combine then in a way that the result makes sense in a meaningful way was of great pleasure.

This semester we reduced the group size from five to three to ensure that everyone has the same expectations from the outcome of this project, as well as is willing to contribute the same amount of energy. Because of that, and the fact that we previously worked together we had quite a good synergy.

Unfortunately, one member had to leave which meant a bit of restructuring of the workload, expected deadline, but most importantly the expectations of the outcome. We quickly realize that we were already doing our best, and cutting 1/3 of the men's power, would have influenced the result of the project.

Nevertheless, I consider than each member of the group did their best, and lived up to the expectations of this project. Even if a bit below what I wanted, we are happy with the outcome of the project, as well as the process of getting in there.

One thing that bothered me throughout the entire semester, was that I constantly felt like we have *no time*. This can be attributed to the fact that we were short one member, or that the projects became larger or both of the above.

6.2. Marcel Notenboom

In this semester the project group had been changed significantly, we have gained a new member since the first semester, but also reduced the group size to three as opposed to a five-member group in the second semester. The reduction in the size of the group has been very beneficial in terms of smaller project scope and ease of communication and planning.

Despite this change in group structure, the chemistry between the group members and the vision of the group was still as solid, also because we already knew each other from previous semesters which made the start of the project smoother. In addition, we did not need to test the waters with each other and as a result, the issues that we encountered were quickly and civilized resolved.

The group did have one significant problem that needed to be solved this semester mid-project which we did not expect and have not experienced before. This was that we lost a group member reducing our already small group size of three into two. As a result, this meant that the work that was originally planned and split into three now needed to be restructured and split between two people. Luckily, Gais who has been acting as the project lead planned in a talk to resolve this issue and apart from a bigger workload per person this did not leave us with a lasting problem.

At the beginning of the project, we decided to not have structured meetings and instead adopt a more fluid project structure. Because of this fluid-structure, we decided to use an online instant messaging *#Slack* platform for near-constant communication. This made sure that even though we did not have weekly physical meetings every member was still in contact with each other and knowledgeable of where the project stands.

For the project overview, we used a digital Kanban software which a project member had sourced a free student version for. This helped the group to keep track of which features and tasks where already done, still needed to be done or the ones which were being

worked on already. Filling out the tasks on this Kanban board was difficult at first because we made each task too big and undefined, later we figured out that we could use the requirements of the project and break them down in small testable sections of work. When we structured the Kanban in this matter we could easily make, test and then implement a section that a group member made.

In order to facilitate the fact that the group needed to work on the same project without physically being together, we made a git-repository which worked out well during the project although we spend more time then we wanted on figuring out how the branches worked since they gave us some problems when we wanted to merge them together with the main branch.

A factor that significantly helped us in this project, in my opinion, was the mid-semester deadline to show a working skeleton of the project. This forced us to focus our attention on the backbone of the project on which the rest was based and therefore also was the very time-consuming part of this project. Having this done in the early stages of the project made it much easier to make the different features and enabled the group to work simultaneously.

7. Supervision

During this project, we have had two supervisors, Jakob and Jan that provided guidance during the entire period. Both teachers, provided great directions and advice for the semester project with matters as organization and structure, as well as the technologies to be used.

During the semester (classes) many times they have been helpful enough to even offer guidance during their non-semester project classes (DNP/SDJ3). We constantly relied on their support when we were in a bottleneck or did not know how a certain technology works.

Even so, during this semester, the provided guidance was more *on-demand* meaning that we needed to ask for it, rather than expect being taken care of, which was a bit uncertain in the beginning.

All in all, I think that supervision provided great help for our software development, in all the stages.

8. Conclusion

After contemplating the semester project, and the development process there are a few points that should be highlighted before making a general conclusion. This point is related to group work, time management, frameworks and technology, and the overall feeling of the semester project.

- **Software Development Framework** – either it is SCRUM, or Kanban or any other framework it is important to figure out what works for the group/project and use one to aid the team. Even if sometimes a taboo, realizing that everyone needs help for self-structuring as well as work-structuring is powerful enlightenment which can help overcome many obstacles on the road;
- **Version Control –** it should be though and used from the first semester. It helps at every point of the software development process and cannot be dismissed in any way;
- **Packet Manager –** even if it is not too much related to the .NET platform as it has one build in. When it comes to working with Java, using a packet manager (Maven in our case) was of great help to utilize different packets and libraries useful for the project development. By using it there was less headache with Java versions, with finding and manually adding packets, and so on;
- **Understanding that Semester Project is a learning tool –** as in the previous two semesters, it was of great focus that we already had the understanding of the use of the semester project. It is not mean to develop great solutions that will change people's lives and *make the world a better place* it is a learning process, for the students, and the better we understanding this fact, much easier is to work with it, gaining a lot of motivation from the learning process;

All in all, the semester project was a great experience, where we got to gain new knowledge about architecture, technologies, languages. We had the possibility not only to learn about them but to utilize them in a project that offered the possibility of

combining and adding everything together in the hope that it will make sense in the end. We consider that this project semester offered the greatest knowledge rewards so far, and doing it has improved us as software engineers and programmers.

9. Source of information

- Larman, C., 2004. *Applying UML and Patterns: An Introduction to Object-Orientated Analysis and Design and Iterative Development.* Third ed. s.l.:Addison Wesley Professional.
- Radigan, D., 2019. *www.atlassian.com.* [Online]
  Available at: https://www.atlassian.com/agile/kanban
  [Accessed 2019].
- Rehkopf, M., 2019. *Agile Coach: Kanban vs Scrum.* [Online]
  Available at: https://www.atlassian.com/agile/kanban/kanban-vs-scrum
  [Accessed 2019].
- Valve Corporation, 2012. *Handbook for New Employees.* 1st ed. Bellevue: Valve Press .
- VIA University College, 2019. *Semester Project: Heterogeneus System (IT-SEP3).* [Online]
  Available at:
  https://studienet.via.dk/sites/uddannelse/ict/Horsens/studymaterial/Pages/Course-Descriptions.aspx
  [Accessed September 2019].

Appendices

Appendix A – Group Contract
Appendix B – Project Proposals
Appendix C – Project Description
Appendix D – Time Reference Table