Employee Planner

Project report

Anders Sønderby Sørensen – 264247

Dat Phuc Nguyen (David) – 251771

Niklas Krogh Jensen – 281335

Rokas Barasa – 285047

Supervisors:

Joseph Chukwudi Okika (JOOK)

Nicolai Sand (NISA)

57.555 Characters

Software engineering

3rd semester

04.06.2020

Table of content

List of figures and tables

1 Abstract

The goal of the project was to create an employee and shift planning system, which would make it easy for company managers and employees to have an overview of their coming shifts, and to manage them.

The project was developed as a heterogeneous distributed 3-tier system in Java and C#, using JavaFX as the presentation tier, ASP.NET Web API as the middleware as well as JDBC for the data access and MySQL as the database server for persistence. The system makes use of several design patterns, including the Singleton pattern, Model-View-ViewModel, as well as the Database Access Object pattern.

The system takes advantage of REST webservices, through the middleware API, to provide flexibility and interchangeability. The response format used by the clients is the JSON format.

In total, 10 out of 21 functional user stories were successfully developed and tested using black-box test cases, of which 43 were made in total.

2 Introduction

On the market there are a broad range of employee / shift management systems, which provide various aspects, hence with the primary activity towards increasement of the productivity, business infrastructure and simple worksheets. The numerous employee / shift management systems globally differ depending on features developed, which correlates to a wide variety of similarity due to slight modifications to be unique. (Michelle Rafter, 2019)

Generally, those systems mentioned above are genuine superior and appropriate for larger businesses and organizations environments due to complexity. A business considered small or medium is troubled on the current market because present systems are too feature packed, which causes a lot of unutilized features and aspects. These small or medium businesses seek and pursue a simplified system that provides their needs and disregards the unutilized features. Since unutilized features are inconvenient to investment and time consumption, which can decrease the productivity and business infrastructure.

The relevance of the project is given by providing an option of a simplified and reliant employee / shift management system, that have focus towards the small and medium businesses. Thereby provides a system without unutilized features that causes inconvenience of time consumption and investment.

Development of this project will have certain delimitations:

- Development according to maintenance of the operational system.
- E-mail communication.
- SMS communication.
- Implementation of security measures.
- Notifications for shift changes
- Salary functionality
- Implementation of employee-only functions (Punching in/out, Shift overview, specific view of shift history, internal swapping of shifts, etc.)

The project is limited to:

- Employee management.

Development of an employee management system will be using the programming languages listed below:

- **Java (Tier 1 & 3)** – Used as Client, Server and JDBC.
- **C# .NET (Tier 2)** – Used as Business Logic.
- **MySQL** – Used as Database.

The development process is derived into several phases; hence each aspect of a phase contributes as essential. All phases are thoroughly described and located in the listed below chapters:

- Analysis phase.
- Requirement phase.
- Design phase.
- Implementation phase.
- Testing phase.

3 Analysis

3.1 Requirements

1st step in the analysis is to find requirements for the software. They are foundation of the software and serve as means to gauge the progress team has made throughout the development. The requirements have been made using SMART principle.

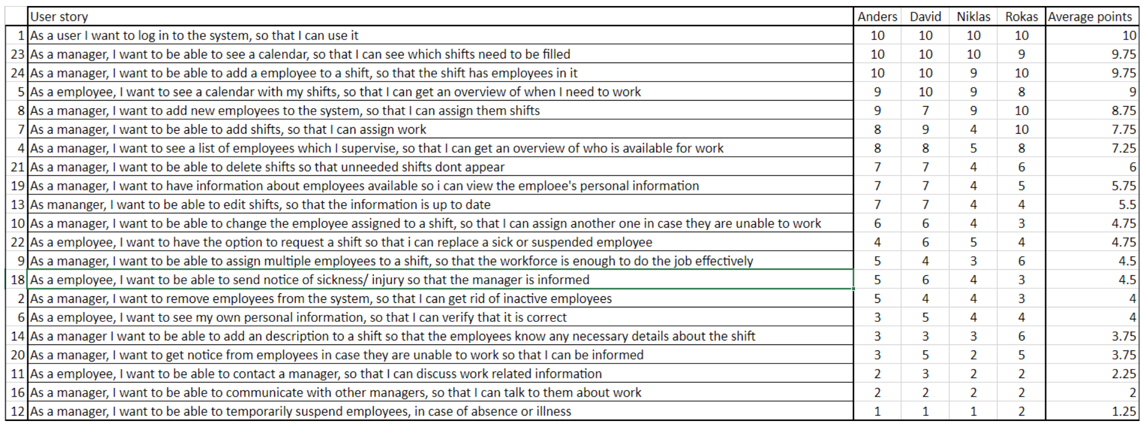
There are 3 actors: Manager, employee and User which includes both mentioned before. Manager is determined as a leading actor and has control over most of the functions in the software.
Employee's main function is to view shifts and communicate about his working status with manager.

3.1.1 Functional requirement

All the functional requirements have been assessed and graded with points 0-10. Here MOSCOW principle is used to determine what is more necessary for the software to have, what should, what could and what is not needed.

- 10-7 points are requirements which are needed as a core of the software and thus are prioritized and need to be developed first.
- 7-5 points are requirements which are important but not vital to the software.
- 5-3 points are nice to have features thus given least importance when developing
- 3-0 points are requirements which are not necessarily needed to be developed and can be give up on. And are developed only if there is excess time during development.

In figure 3-1 you can see 21 requirements graded by development team, product owner with consideration for all above mentioned principles and models.



	User story	Anders	David	Niklas	Rokas	Average points
1	As a user I want to log in to the system, so that I can use it	10	10	10	10	10
23	As a manager, I want to be able to see a calendar, so that I can see which shifts need to be filled	10	10	10	9	9.75
24	As a manager, I want to be able to add a employee to a shift, so that the shift has employees in it	10	10	9	10	9.75
5	As a employee, I want to see a calendar with my shifts, so that I can get an overview of when I need to work	9	10	9	8	9
8	As a manager, I want to add new employees to the system, so that I can assign them shifts	9	7	9	10	8.75
7	As a manager, I want to be able to add shifts, so that I can assign work	8	9	4	10	7.75
4	As a manager, I want to see a list of employees which I supervise, so that I can get an overview of who is available for work	8	8	5	8	7.25
21	As a manager, I want to be able to delete shifts so that unneeded shifts dont appear	7	7	4	6	6
19	As a manager, I want to have information about employees available so i can view the emploee's personal information	7	7	4	5	5.75
13	As mananger, I want to be able to edit shifts, so that the information is up to date	7	7	4	4	5.5
10	As a manager, I want to be able to change the employee assigned to a shift, so that I can assign another one in case they are unable to work	6	6	4	3	4.75
22	As a employee, I want to have the option to request a shift so that i can replace a sick or suspended employee	4	6	5	4	4.75
9	As a manager, I want to be able to assign multiple employees to a shift, so that the workforce is enough to do the job effectively	5	4	3	6	4.5
18	As a employee, I want to be able to send notice of sickness/ injury so that the manager is informed	5	6	4	3	4.5
2	As a manager, I want to remove employees from the system, so that I can get rid of inactive employees	5	4	4	3	4
6	As a employee, I want to see my own personal information, so that I can verify that it is correct	3	5	4	4	4
14	As a manager I want to be able to add an description to a shift so that the employees know any necessary details about the shift	3	3	3	6	3.75
20	As a manager, I want to get notice from employees in case they are unable to work so that I can be informed	3	5	2	5	3.75
11	As a employee, I want to be able to contact a manager, so that I can discuss work related information	2	3	2	2	2.25
16	As a manager, I want to be able to communicate with other managers, so that I can talk to them about work	2	2	2	2	2
12	As a manager, I want to be able to temporarily suspend employees, in case of absence or illness	1	1	1	2	1.25

Figure 3-1 Initial requirements

3.1.2 Non-functional requirements

Non-functional requirements are not graded but need to be developed.

- The system must be developed as heterogenous system
- The system must be developed in 2 or more coding languages
- The system must have 3 tier architecture
- The project will be developed and managed with SCRUM
- The system must consume and expose one or more web services
- The system must use sockets for communication in between one of the tier layers.

These has been derived from course information and presentation from SEP classes.

3.1.3 Use case diagram

From initial requirements, use case diagram was made by compacting similar requirements into a use case. This gives overview of main actors and features which interact with each other. And show them in compact view to show to customer or stakeholders.

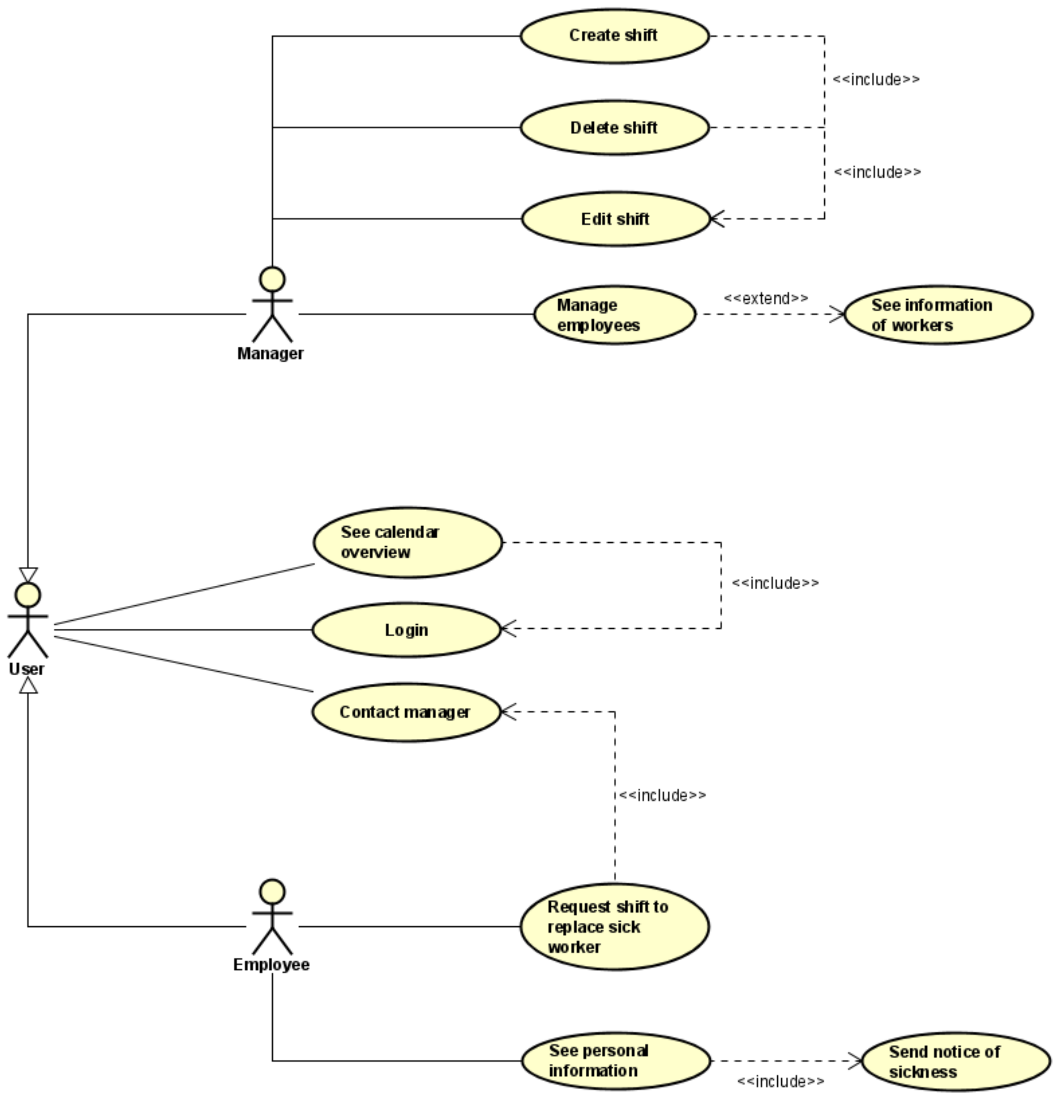


Figure 3-2 Use case diagram

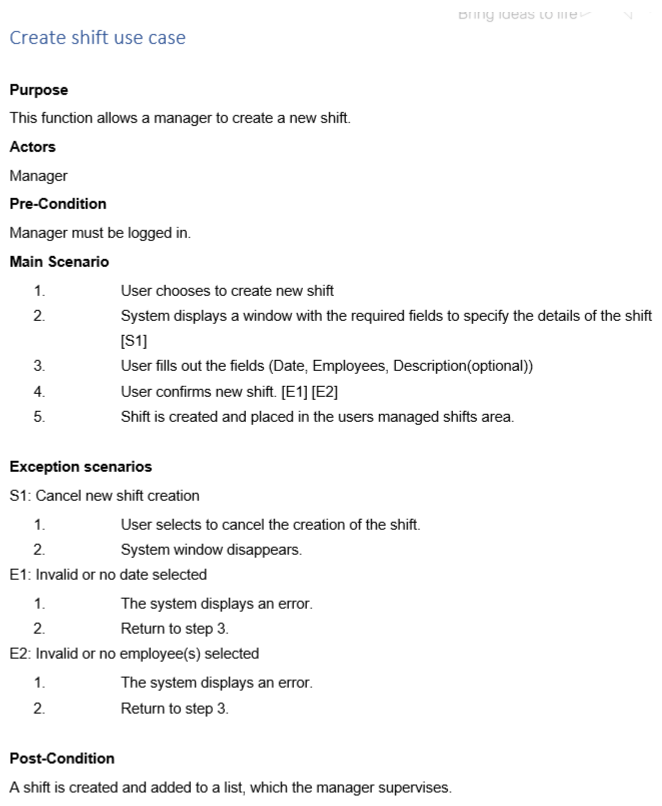
As you can see in figure 3-2 above you can see user representing both manager and employee. User can log in, view calendar and contact its manager.

Manager can create, delete and edit shift. Manager should be able to manage employees (add,

remove, change information about employees and see their information)
Employee should be able to see personal information about themselves and request shift if needed.
Use case diagram can be accessed in Appendix B

3.1.4 Use case description

Use case descriptions here and after UCD were drafted after use case diagram had been made. UCD is made for every use case. They are needed for use case testing. In figure 3-3 below is shown create shift use case which is core requirement for the system. This description provides overview of creating shift by manager.



Create shift use case

Purpose
This function allows a manager to create a new shift.
Actors
Manager
Pre-Condition
Manager must be logged in.
Main Scenario
1. User chooses to create new shift
2. System displays a window with the required fields to specify the details of the shift [S1]
3. User fills out the fields (Date, Employees, Description(optional))
4. User confirms new shift. [E1] [E2]
5. Shift is created and placed in the users managed shifts area.

Exception scenarios
S1: Cancel new shift creation
1. User selects to cancel the creation of the shift.
2. System window disappears.

E1: Invalid or no date selected
1. The system displays an error.
2. Return to step 3.

E2: Invalid or no employee(s) selected
1. The system displays an error.
2. Return to step 3.

Post-Condition
A shift is created and added to a list, which the manager supervises.

Figure 3-3 Use case description for Create shift

The rest of the use case description can be accessed in Appendix C

3.1.5 Domain model

Last part of analysis is to make domain model. Which shows visual representation of conceptual classes or real-world objects in the domain of interest. (Larman, 2005)
Domain model has been made with emphasis on requirements and wishes of the client. Shows behaviour and functionality between actors (user, manager, employee) and objects (shift, request, notice).

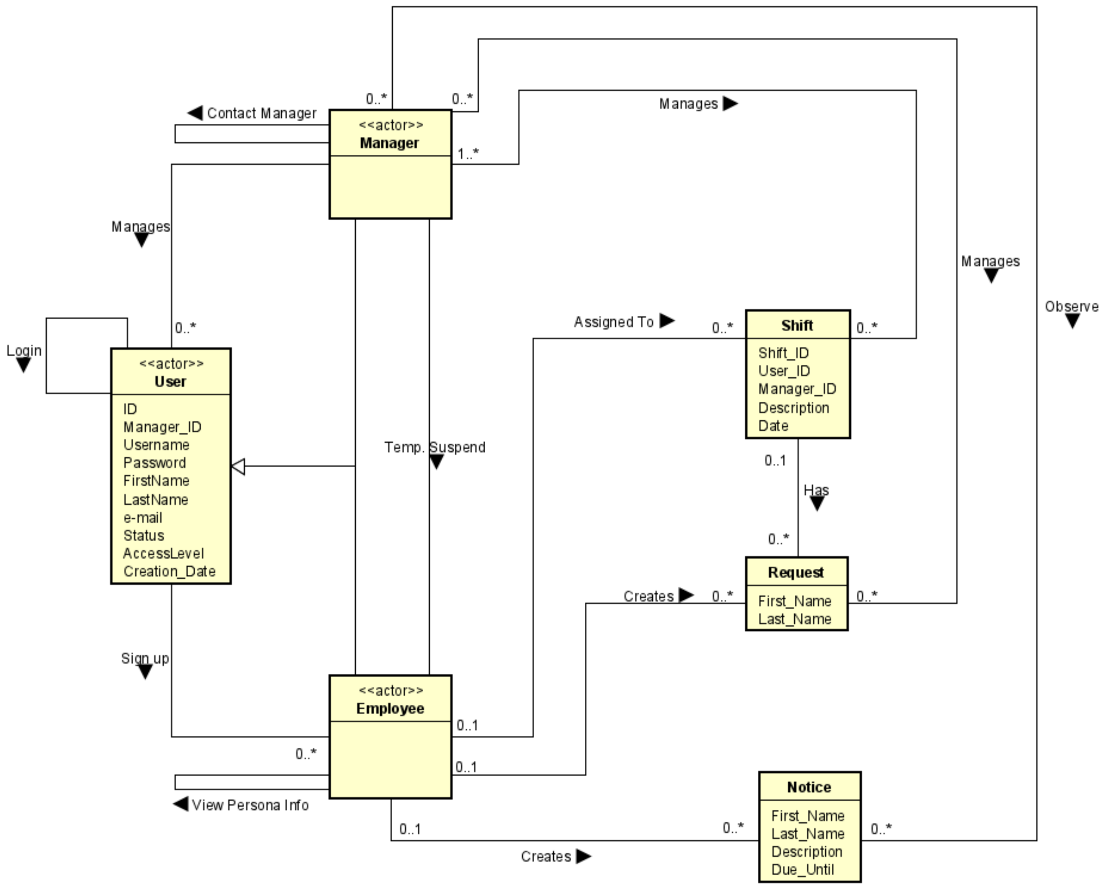


Figure 3-4 Domain model

Domain model can be accessed in appendix B

3.2 Security

3.2.1 Threat model

The system would have threats of:

- **Spoofing identity**: A man in the middle attack is where the attacker would send a message to the server, and the server would think it is a person in the system.
- **Tampering**: Modifying data which is being sent through unauthorized means.
- **Repudiation**: A client saying it have sent something and then denies the entry. Could be a real concern if the system is known for lack of tampering security. Will be delimited since it is closely tied to tampering.
- **Denial of service**: An attacker floods the server with requests such the server must take care of all of them, before executing actual user request, which is blocking actual users from using the system.
- **Elevation of privilege**: Attacker gains more rights in a system then they are supposed to have. Would probably happen while tampering if not properly managed. Will be delimited since it is closely tied to tampering.

The system will not be dealing with sensitive information, that includes messaging between managers and employees. Thereby will the system not be protected against passive attacks. The type of attacks that can be worrying towards the system are those called active attacks, which are listed below:

- **Replay attacks**: The attacker repeats an old message, like a login.
- **Blocking**: The attacker stops a message from arriving.
- **Modification**: The attacker changes a message or entirely replaces it.

The attacks would originate from external attackers or insiders:

1. Insiders will be delimitated, assuming the registered user cannot possibly do anything outside of his functionalities.

2. An external attacker would mainly attack the system through network attacks.

Offline attacks would be very uncommon, because the attacker would not have the option to access files from the client. If the attacker somehow gains access to the servers or database the attack would be devastating, letting the attacker tamper with information on the database and even deleting the stored information and schema.

An online attack would preferably target the server, where the system is hosted instead of a direct attack to the software. This attack would only be bad for the system if the attacker could somehow break into the database and start impersonating the host of the database. An online attack is very unlikely and would be difficult because the database's security such as username and password are needed.

The intention of the attacker would be to tamper with the organization's ability to manage their employees. If tampering happened to the organization, it would cause a security breach and possible deletion of internal database data.

3.2.2 Security objectives

The main objective is to prevent tampering of data by attackers.
The second objective would be to prevent offline attacks to the database.
The third objective would be to minimize denial of service attacks.

3.2.3 Risks and risk assessment model

The first problem would be tampering which would not be uncommon, since it is an easy way of attacking the system. A way to prevent this would be to set up a MAC which would authorize the messages and prevent tampering, when using a MAC, it authenticates the messages to make sure they are made in the user and not changed. The only way this security system would fail is if a key that the MAC uses gets compromised. The attacker could change the message in a way that would use the affected user's functionalities, which would be disastrous. A way to fix it after it have been tampered, would be to change the keys and perhaps revert the users changes to a certain date.
The total risk for this threat is moderate as it is low likelihood it will happen.

Secondary risk:
There are several layers of authentication preventing an offline attack on the database and the frequency of this happening is highly unlikely, but effect of this threat could be catastrophic. The only correct measure would be to either encrypt the entire database which results in slow accessibility or add password hashing and backup the database frequently. The second option would prevent the sensitive information from getting into the wrong hands and would minimize losses of content by having a copy of the database.
The risk of this happening is low.

Third risk:
This attack could be minimized by configuring network security, using a firewall or focusing on network architecture such as balancing the load on the servers. Another approach would be to use a filter that prevents spam. As mentioned above a MAC security method could lower the impact of this attack, by not processing the requests completely, rather discarding them if the encryption does not match. The frequency of these attacks would be below average as they require sophistication to achieve. The consequences of these attacks would be minimal as they would only prevent users from accessing the system.
The risk of this threat is moderate.

4 Design

The aspects described in the analysis section are used to outline a structure for the system in the design section.

4.1 Use of two languages

The project was created as a heterogeneous system, because of the beneficial approaches when using different programming languages. A heterogeneous system generally uses different languages, different operating systems. Java provided a simplified approach when creating a GUI through the graphical toolkit JavaFX and Scenebuilder. Java provided a simple database access point, due to familiarity with the use of a Java Database Connection (JDBC). C# provided a simple and good API management framework using ASP.NET Core.

A 3-tier architecture allows the system to be expandable and reduce load on either tier by adding more of the same tier. A 3-tier architecture divides the system into tiers of functionality, typically composed of a presentation, business logic and database.

- The presentation tier is only used to show the information it receives from the layers below, business logic and database.
- The business logic is used is used to process logical operations when the client makes requests from the database.
- The database tier is just an access point for retrieving or sending information from the database, which in the projects case has a socket connection.

Taking this into consideration, the system was devised such that Java was used in the presentation tier, C# in the business logic and Java as the server connection to the database.

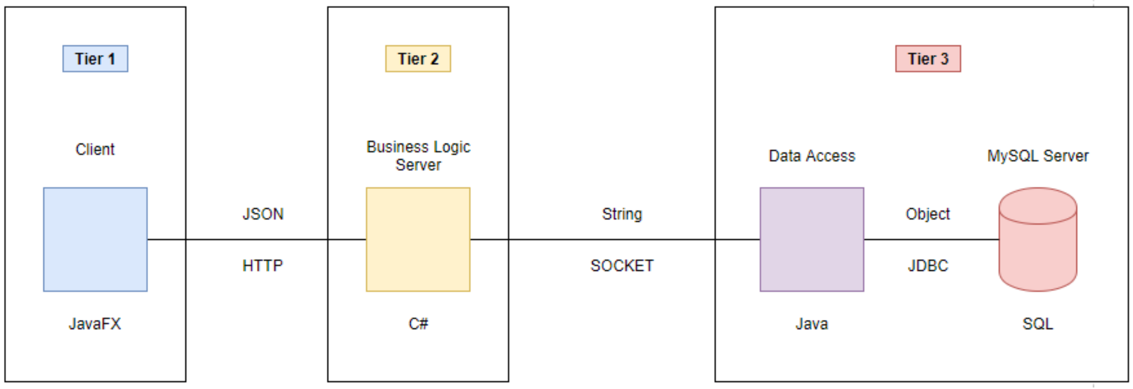


Figure 4-1 Architecture model

4.2 GUI design and user functionality

The GUI is operated using JavaFX. Upon starting the system, the user is greeted by a login view. When a valid user has logged into the system, then the user is provided with a main view. The main view is divided into two sections, a menu and calendar. The calendar is controllable by selecting the month and year to display, as default it displays current month and year. This allows the user to see past shifts and upcoming in a calendar format. The calendar format was chosen because it is a convenient and simplified layout. The main view, consisting of the calendar, provides an overview towards which days during the week and month their shifts is due. Furthermore, an alternative approach with a table view is given in the menu field, which supports the ability to view a whole month's worth of shifts.

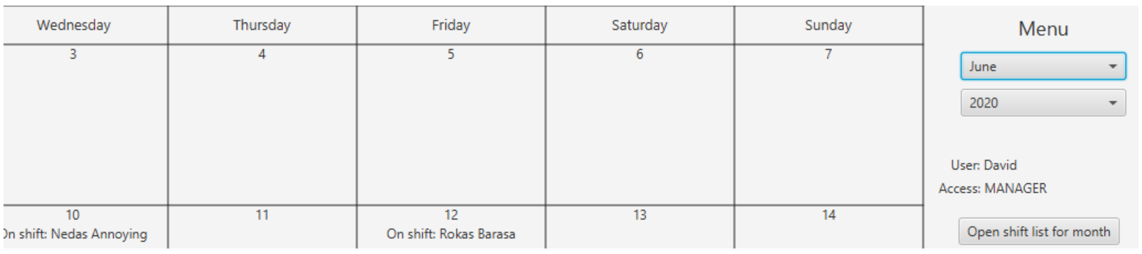


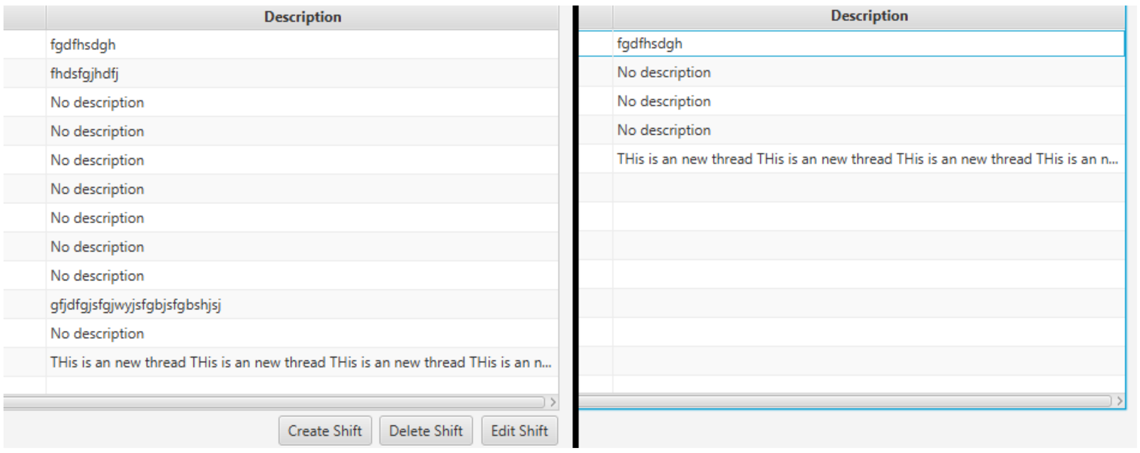
Figure 4-2 GUI design for Calendar view

The menu is a placeholder of all functionality the user can perform along with calendar controls. The buttons can be disabled to accommodate different functionality profiles.

The GUI's functionality is supported by a split between two users. The user referred to as a "manager" has all the features of the program available. The main features of the manager are adding shift, deleting shift, list of users their managing, adding users, deleting users, and the ability to view all shifts, either in the calendar or shift list.

The user referred as an "employee" has limited functionalities. The user cannot view the list of employees, including the features it holds. The employee can only view their own shifts on the calendar and the monthly shift list.

Comparison between access level:

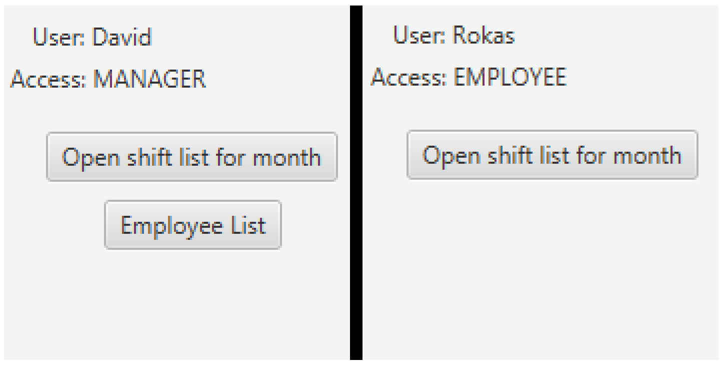


Description
fgdfhsdgh
fhdsfgjhdfj
No description
No description
No description
No description
No description
No description
No description
gfjdfgjsfgjwyjsfgbjsfgbshjsj
No description
THis is an new thread THis is an new thread THis is an new thread THis is an n...

Create Shift Delete Shift Edit Shift

Description
fgdfhsdgh
No description
No description
No description
THis is an new thread THis is an new thread THis is an new thread THis is an n...

Figure 4-3 Shift list comparison between employees and managers



User: David
Access: MANAGER

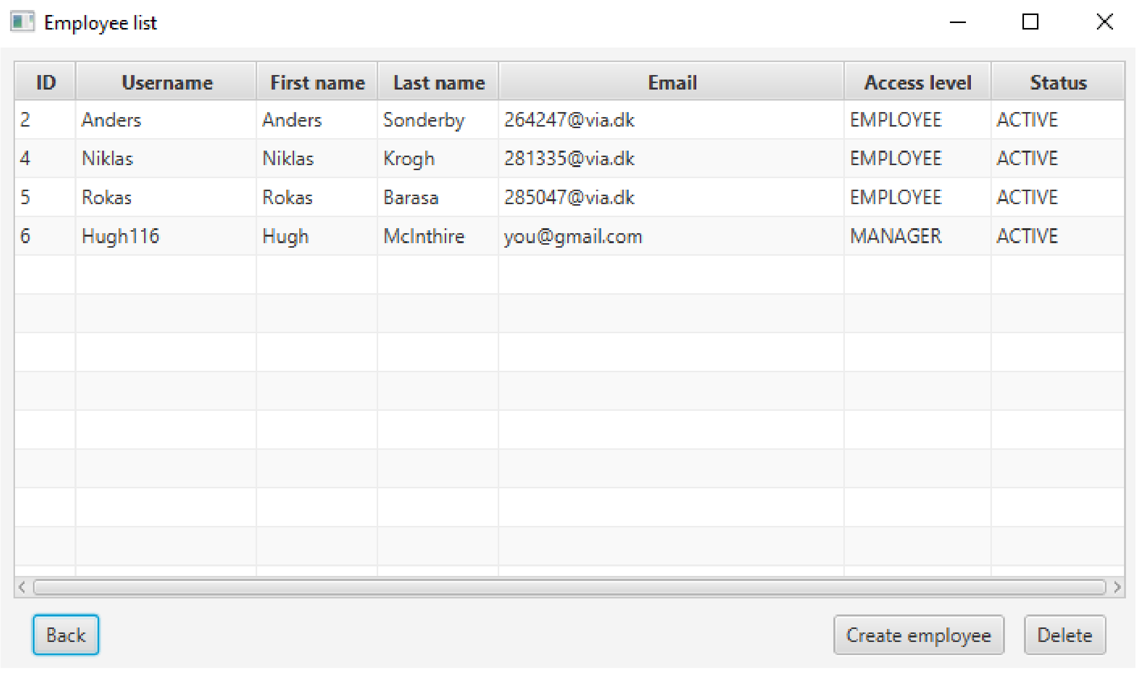
User: Rokas
Access: EMPLOYEE

Figure 4-4 User functionality availability comparison between employees and managers

The GUI was designed using a split between different users, as seen in figure 4-4. Some issues arose during the creation of the calendar, due to trouble implementing the correct date on a day, which in the end became possible through an alternative implementation method. The shift list acts as a supplement, which makes it easier to select which shift to perform different functionalities.

4.3 User functionality

The system is designed to have a hierarchy of users. Ideally the system would be configured to have an admin user for the owner of the business. This admin user, which is referred as the manager can create other managers or employees, the mentioned managers can create and manage as fitted their created managers and employees. As seen in figure 4-5 the entire list of employees the logged in manager manages.



ID	Username	First name	Last name	Email	Access level	Status
2	Anders	Anders	Sonderby	264247@via.dk	EMPLOYEE	ACTIVE
4	Niklas	Niklas	Krogh	281335@via.dk	EMPLOYEE	ACTIVE
5	Rokas	Rokas	Barasa	285047@via.dk	EMPLOYEE	ACTIVE
6	Hugh116	Hugh	McInthire	you@gmail.com	MANAGER	ACTIVE

Figure 4-5 Employee list available to managers

4.4 MVVM

The client (presentation tier) uses MVVM, an architectural pattern that divides the client into layers of functionality, as seen in figure 4-6. View, ViewModel and Model are the different layers, which makes it a reliable pattern for maintenance and expandability.

- The view layer contains the user interface and only represents the GUI.
- The ViewModel layer contains the formatting for the GUI.
- The Model normally contains logic for the client, but since the project is split into a 3-tier architecture, the logic will be contained in the business logic.

All the layers have a factory class that is lazily instantiated, such that a layer is only initiated when it is needed. The MVVM structure is supplemented to the client of the 3-tier architecture, such that the code is split by its functionality. The client layer handles the networking of the presentation tier that connects to the business logic.

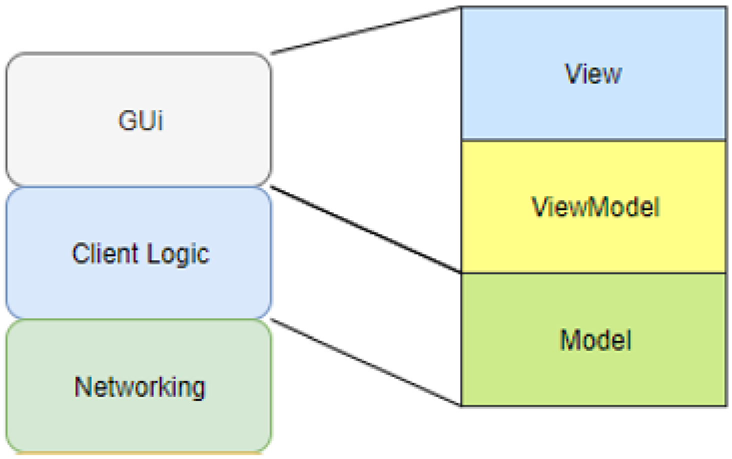


Figure 4-6 MVVM structure

4.5 Networking

The connection between the client and the business logic server is handled by an HTTP protocol. A decision was made to use restful webservices, since it does not take much time to implement requests and it can use JSON instead of only having XML available as a format. The system is not complex enough to use the alternative of SOAP. The client uses plugin class of

RestEasy to handle the connection. RestEasy was chosen over the Java spring framework as there was no need for all the features the spring framework was providing. To serialize the messages to JSON the system uses the JSON serializer plugin Gson.

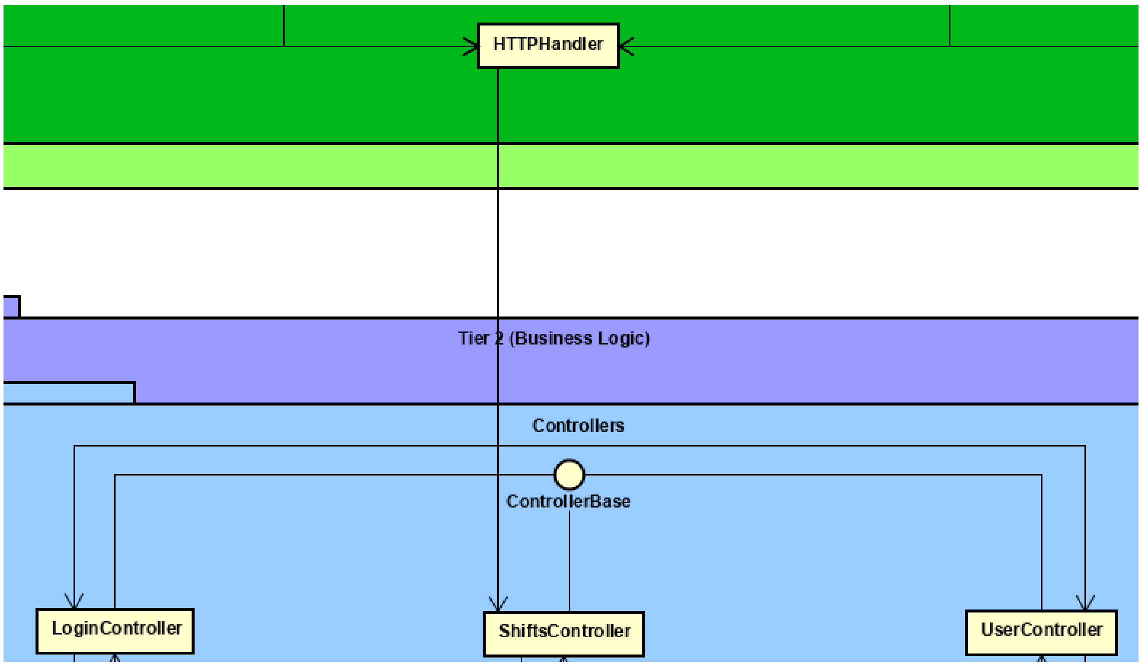


Figure 4-6 A snippet of the class diagram showing the connection between the client HTTPHandler and the API controllers

The HTTP connection is not constant. The client only connects to it to perform a function and it is kept alive until the function is complete. This saves performance for both the business logic and the client.

The business logic server does not need to keep a large number of users constantly connected and waste processing power pinging every single one to keep the connection alive. Another benefit is that this also removes the need for threads, as they are created dynamically when a client performs a request.

The client performance saves come from the same lack of need to keep a connection as well as a lack needing to loop and try to reconnect if a connection goes down. This prevents interference to the user if a connection is lost, if the user does not load something new into the system. If the connection is available, the user would find out by clicking a functionality button again. Sockets would not be a good choice for the communication of these two layers.

For the business logic and database, a socket connection is a fine choice because it would not have may connections to databases compared to clients. The business logic should not work while the database it is connected to is not working.

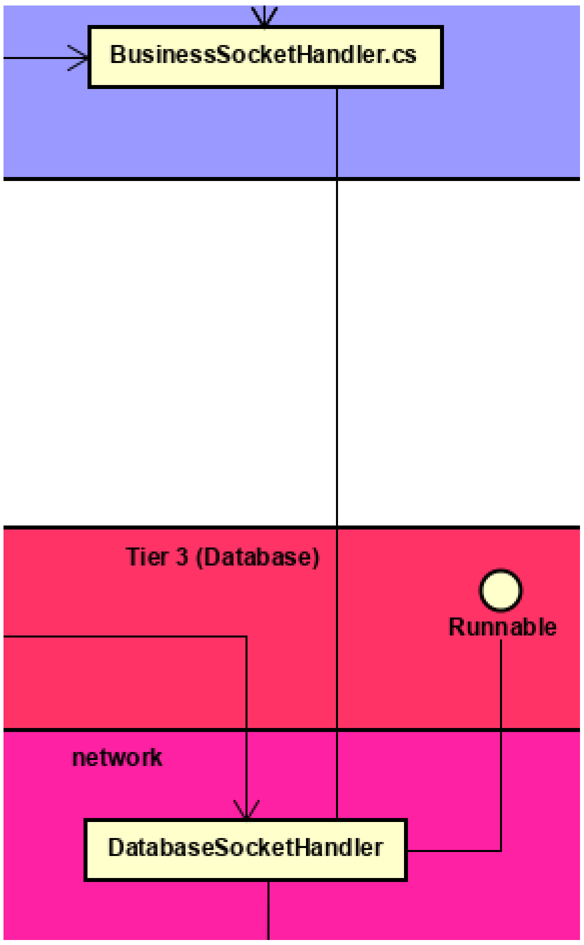


Figure 4-7 A snippet of the class diagram showing the socket connection between the API and the data access server

The connection between the business logic and the database is handled by a socket connection. It uses TCP for a reliable datastream and is mostly based on a “Command;JSONData;” format approach to request processing in the data access server.

The business logic socket handler is not a separate thread. A decision was made to make the sending of data through the socket accessible by any thread at any time.

The receiving of data is different, immediately after a request is made the thread steps into a response method in the socket handler which has a lock making only one thread access it at a time this way preventing response mix-up between threads. The message being sent is a string that is converted into bytes and length in bytes before sending.

4.6 Singleton pattern

An issue was noticed, there was no way to pass parameters into the API controller constructor when it is instantiated, and a way was needed for all models to have the same business socket handler instantiated in order to send information through a single socket.

A singleton pattern was used as the solution when creating the socket handler, since it is a class that can only have one instance of itself available throughout the whole tier.

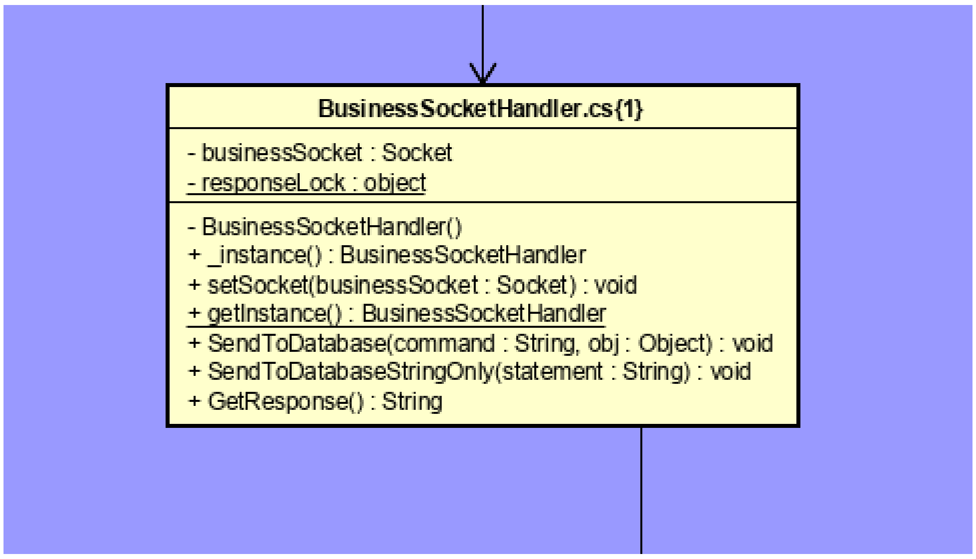

BusinessSocketHandler.cs{1}
- businessSocket : Socket
- responseLock : object
- BusinessSocketHandler()
+ _instance() : BusinessSocketHandler
+ setSocket(businessSocket : Socket) : void
+ getInstance() : BusinessSocketHandler
+ SendToDatabase(command : String, obj : Object) : void
+ SendToDatabaseStringOnly(statement : String) : void
+ GetResponse() : String

Figure 4-8 A snippet of the class diagram which shows the singleton pattern applied in the web API

4.7 Enhanced Entity relationship diagram

The relationship is very simplified because it only consists of two entities:

- User
- Shift

In the user entity a field is named "accessLevel" which determines whether the user is a manager or an employee, as seen in figure 4-10.

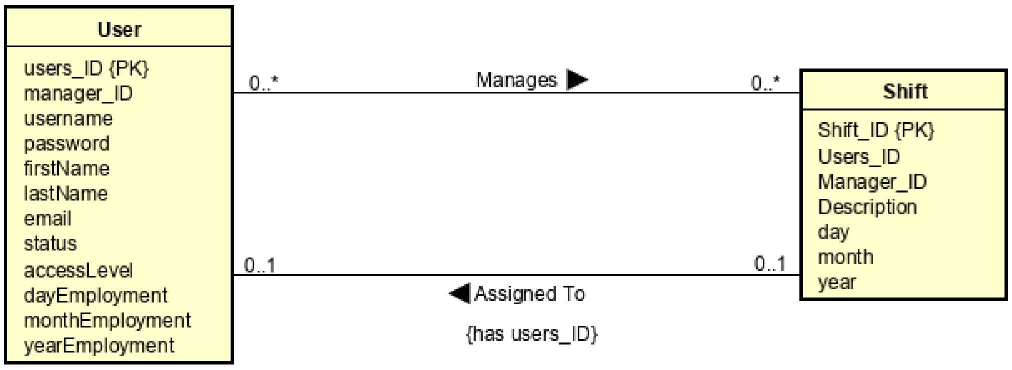


Figure 4-9 EER diagram

4.8 Database

For the database server, MySQL was chosen. The main reason for choosing it was to gain experience in a new database management system and because MySQL was deemed more mainstream than PostgreSQL, which was the main focus in the 2nd semester.

To connect to database, JDBC was used. Prepared statements were used to query the database. This was done mainly to avoid mix-ups when using execute queries, for security reasons, as well as to make each function single responsibility.

4.8.1 Database access point and Data access object pattern

To keep database functionality separated, Data Access Objects, from here on referred to as "DAO" were created to handle the CRUD functionalities of adding, retrieving, updating, and removal from the database.

Since the DAO provides a single responsibility, while specifying operations without exposing crucial details from the database. This can be seen in figure 4-11, which shows an overview of the DAO and database connection.

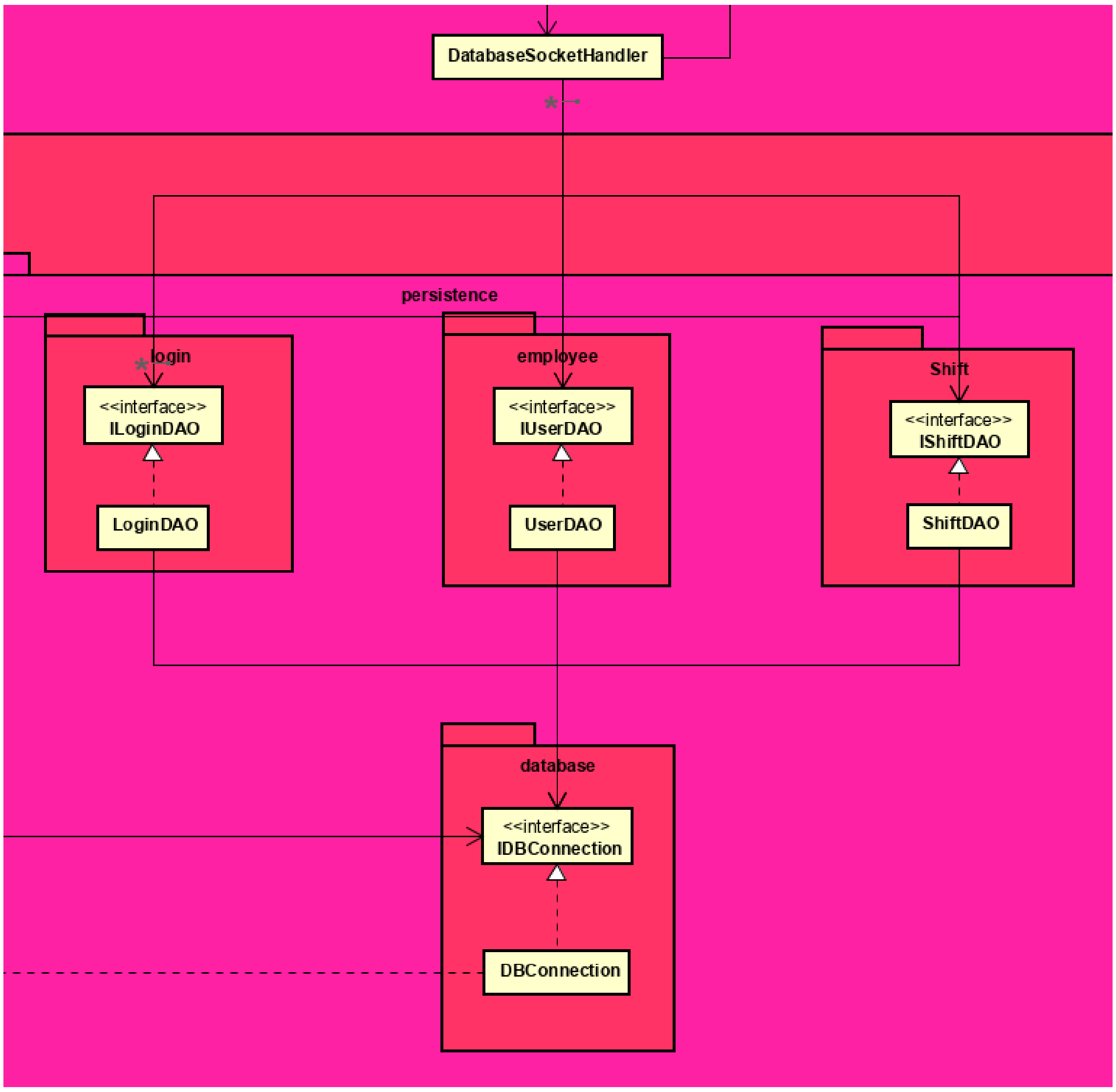


Figure 4-10 A snippet of the class diagram which the connection between the JDBC Data Access Object classes

4.8.2 Database queries and tables

As seen above in the EER diagram, the system has 2 main actors whose access levels are differentiated by the field 'accessLevel' by having either the value 'EMPLOYEE'or 'MANAGER'. Both are stored in user table, which has both employees and managers.
Next up is the shift table, where 'shiftID' is primary key and serves as differentiation between shifts. Date of shift and date of creation of user is stored in 3 different integer fields to make time formatting easier.

The ER diagram and EER diagram can be accessed in Appendix B.

4.9 Security in system

The first security objective of preventing tampering is taken care of by using a message authentication code (MAC). The system would prepare the message, copy it and add a timestamp to avoid replay attacks and then encrypt it using a symmetric key. The server would use the same symmetric key to do the same process and if the result is equal it would allow the message to be processed. This would work both for the http connection and for the socket connection. If the key is compromised, the system could issue an update to all users and replace the symmetric key.

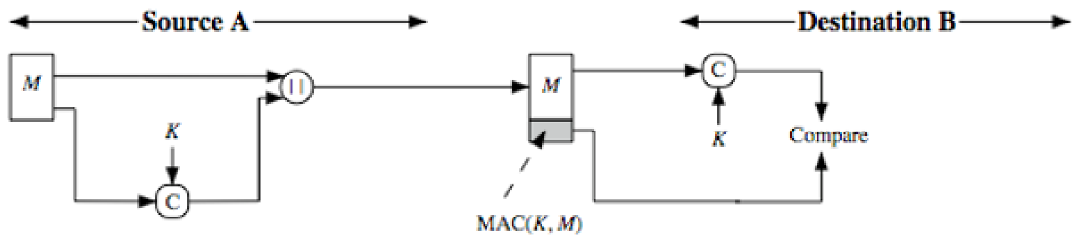


Figure 4-11 A visual display of a Message Authentication Code

Solving the second objective of offline database attacks is done by both hashing the password and doing backups of the database. Hashing the password would happen at the client level when a user is created and when trying to log in. The hashed password would be stored in the database, where it would be compared to login attempts. Backups of the database would

happen daily and each back up would be held for 1 month before being discarded if no emergency happens.
Solving the third objective is not as simple, there are only things that help manage it. The first is the 3-tier architecture, which has the advantage of being easily expandable and can be used to divide load of the system, should a denial of service attack happen.

A lot of the load of this attack would be taken care of by the message authentication code, when it would check if the codes match and If not would discard the message, quickly cutting the time that the attack messages are processed.

5 Implementation

The implementation of add shift is the essential part of the system therefore in the upcoming section there will be a focus on how it travels through the system. The sequence diagram shown below in figure 5-1 can be accessed in Appendix B.

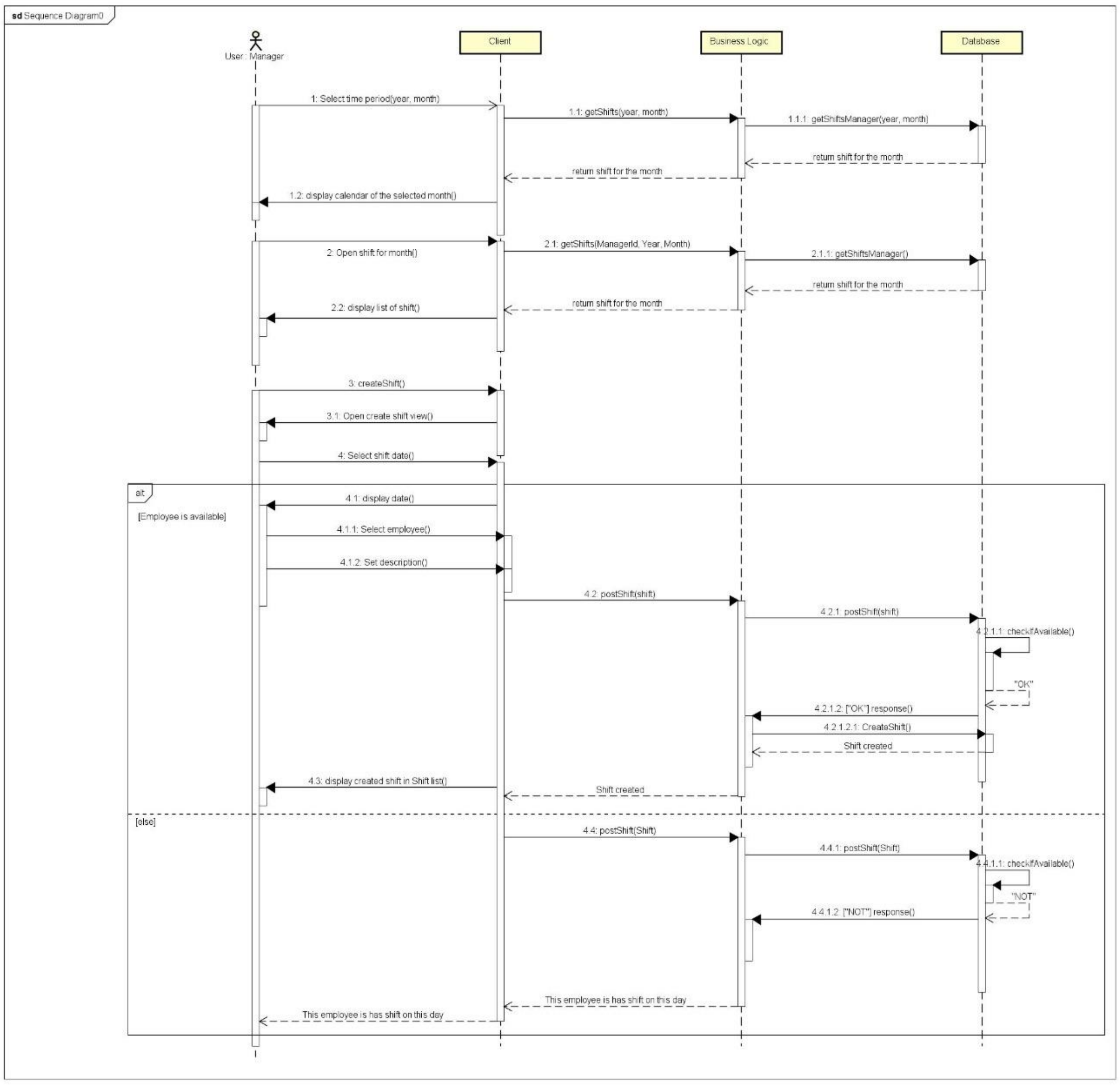

sd Sequence Diagram0
User : Manager
Client
Business Logic
Database
1: Select time period(year, month)
1.1: getShifts(year, month)
1.1.1: getShiftsManager(year, month)
return shift for the month
return shift for the month
1.2: display calendar of the selected month()
2: Open shift for month()
2.1: getShifts(ManagerId, Year, Month)
2.1.1: getShiftsManager()
return shift for the month
return shift for the month
2.2: display list of shift()
3: createShift()
3.1: Open create shift view()
4: Select shift date()
alt
[Employee is available]
4.1: display date()
4.1.1: Select employee()
4.1.2: Set description()
4.2: postShift(shift)
4.2.1: postShift(shift)
4.2.1.1: checkIfAvailable()
"OK"
4.2.1.2: ["OK"] response()
4.2.1.2.1: CreateShift()
Shift created
4.3: display created shift in Shift list()
Shift created
[else]
4.4: postShift(Shift)
4.4.1: postShift(Shift)
4.4.1.1: checkIfAvailable()
"NOT"
4.4.1.2: ["NOT"] response()
This employee is has shift on this day
This employee is has shift on this day

Figure 5-1 Sequence diagram

Shown in ascending order:

- AddShiftController – *View – Tier 1*
- AddShiftViewModel – *ViewModel – Tier 1*
- AddShiftModel – *Model – Tier 1*
- ShiftClient – ClientHandler – Tier 1
- HTTPHandler – Network – Tier 1
- ShiftController – Controllers – Tier 2
- ShiftModel – ServerModel– Tier 2
- BusinessSocketHandler – *Network – Tier 2*
- DatabaseSocketHandler – *Network – Tier 3*
- ShifDAO – Data Access Object– Tier 3
- DBConnection – Data base connection– Tier 3

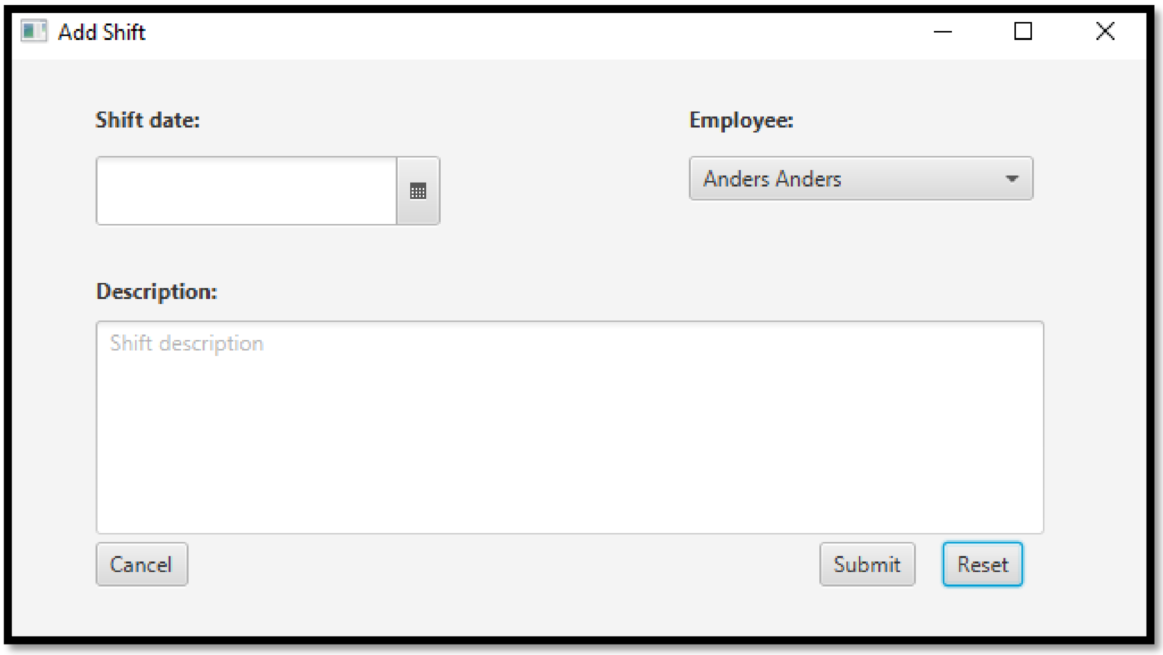
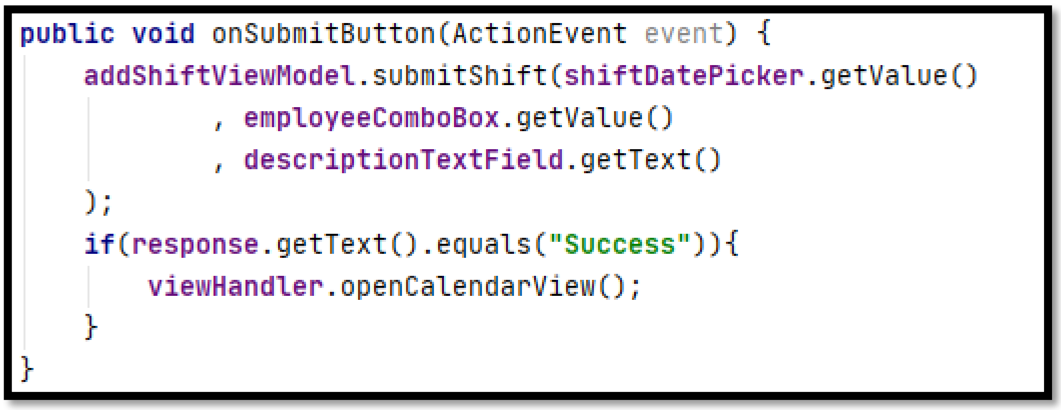


Figure 5-2 Add shift view

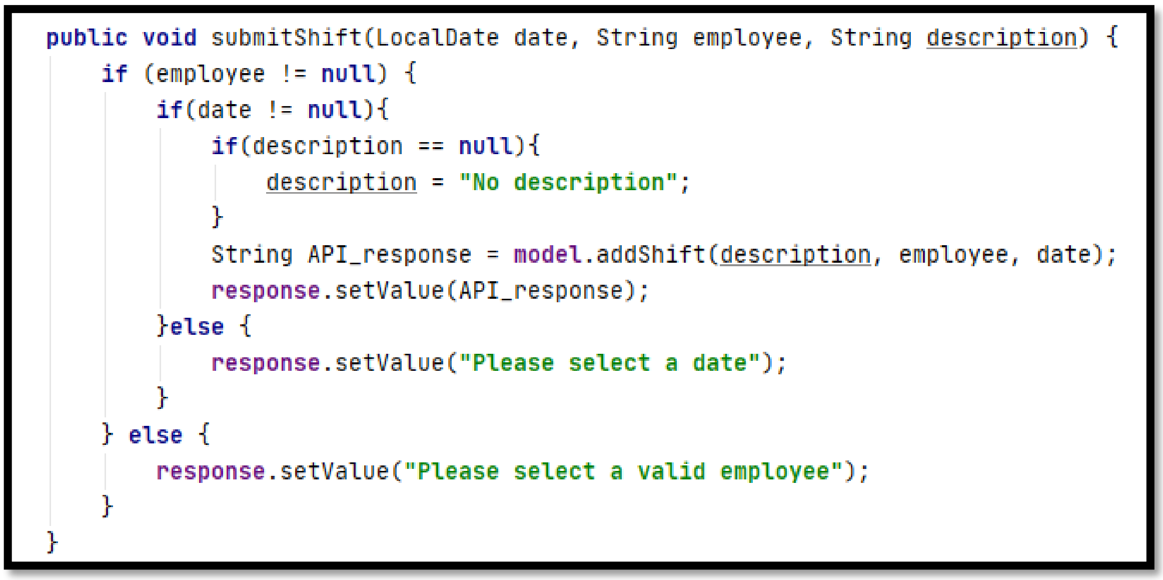
Before the manager can start adding there is a setup process that is needed to understand upcoming sections. When the view is initiated it calls a method that gets all users managed by the current manager id and fills the drop-down menu of employee selection. The employees are shown in a format of “Name Username” this will be explained further on. After this setup, the window is ready for use. The shift needs to have values of employee and date. After the manager presses the submit button the system checks if any of the values are empty, if they are, it asks for the user to input them. The description of the shift is optional, if it is not set it is shown as “No description”. If the mandatory fields of employee and date are not filled the system displays a message asking the user to enter missing details to the fields. Once submitted the details of shift are collected by the onSubmiButton method and sent for checking to the



```
public void onSubmitButton(ActionEvent event) {
    addShiftViewModel.submitShift(shiftDatePicker.getValue()
            , employeeComboBox.getValue()
            , descriptionTextField.getText()
    );
    if(response.getText().equals("Success")){
        viewHandler.openCalendarView();
    }
}
```

Figure 5-3 AddShiftController method onSubmitButton

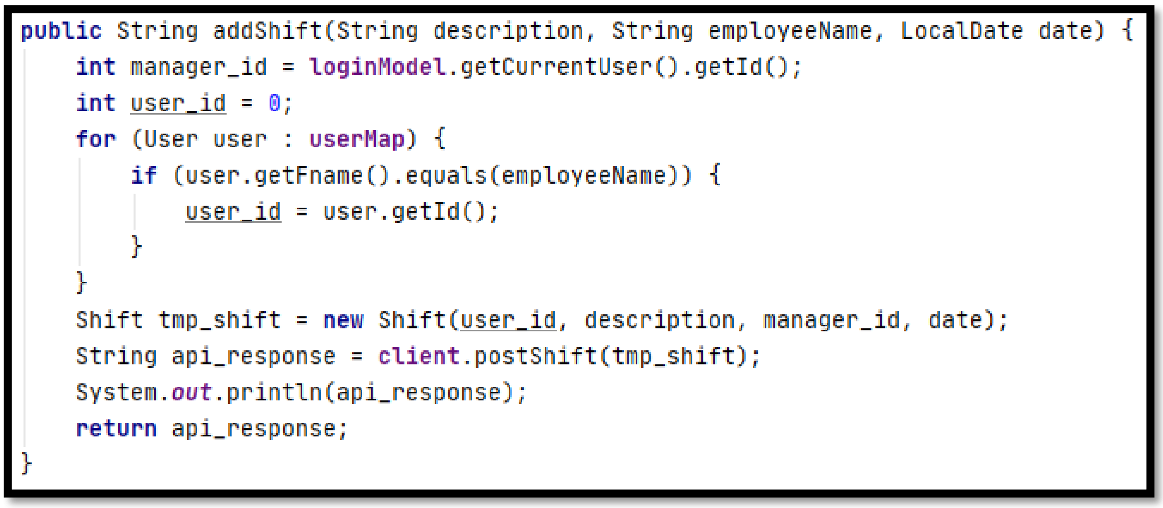
Once submitted the details of shift are collected by the onSubmitButton method and sent for checking to the AddShiftViewModel that is the before mentioned checking happens.



```java
public void submitShift(LocalDate date, String employee, String description) {
    if (employee != null) {
        if(date != null){
            if(description == null){
                description = "No description";
            }
            String API_response = model.addShift(description, employee, date);
            response.setValue(API_response);
        }else {
            response.setValue("Please select a date");
        }
    } else {
        response.setValue("Please select a valid employee");
    }
}
```

Figure 5-4 AddShiftViewModel method submitShift

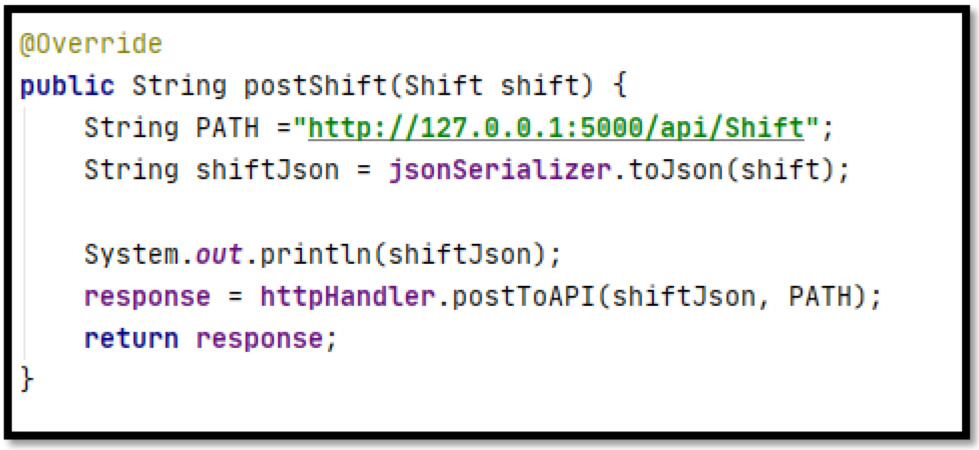
The system then passes the details of the shift to the AddShiftModel class for shift creation. Before the shift is created the model gets the user id from the employee username, that was picked by the manager. The reason for using the username in the selection of employee along with name is because multiple people can have the same name. Username is unique and a good way to find the correct user. The name is needed to easier identify who that employee is by the manager. The user is created and sent to the class of AddShiftClient, the model starts waiting for a response from the business logic server and will send a response to the window as soon as it arrives.



```java
public String addShift(String description, String employeeName, LocalDate date) {
    int manager_id = loginModel.getCurrentUser().getId();
    int user_id = 0;
    for (User user : userMap) {
        if (user.getFname().equals(employeeName)) {
            user_id = user.getId();
        }
    }
    Shift tmp_shift = new Shift(user_id, description, manager_id, date);
    String api_response = client.postShift(tmp_shift);
    System.out.println(api_response);
    return api_response;
}
```

Figure 5-5 AddShiftModel method addShift

The client uses a pre-defined URL that has an attached manager id to act as a link to the business logic server. Before sending objects trough http the system serializes them into JSON.

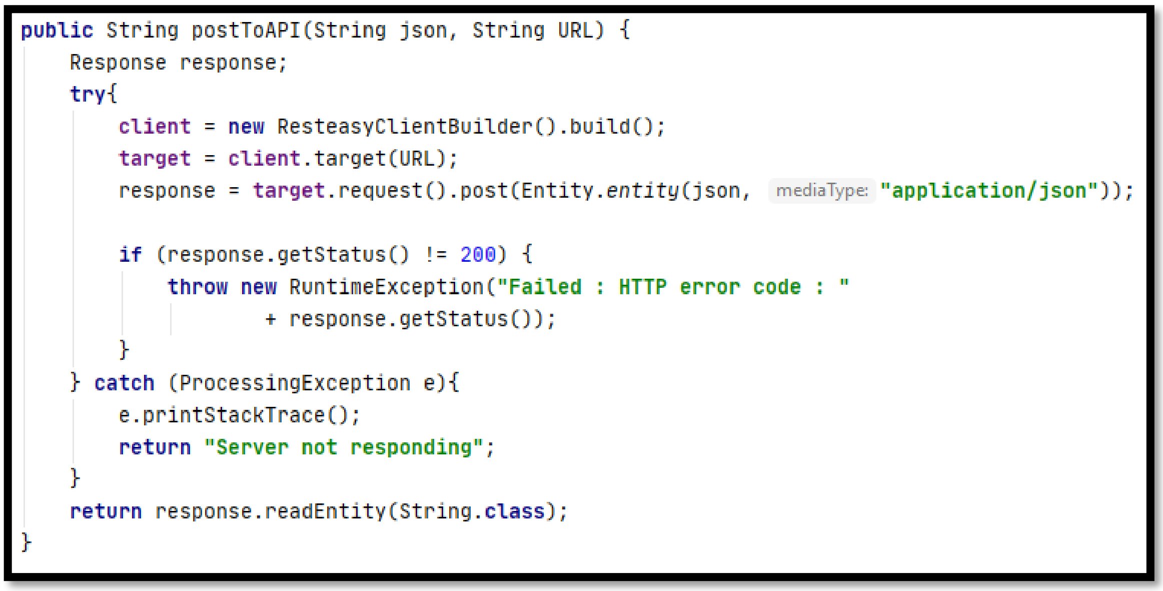


```java
@Override
public String postShift(Shift shift) {
    String PATH ="http://127.0.0.1:5000/api/Shift";
    String shiftJson = jsonSerializer.toJson(shift);

    System.out.println(shiftJson);
    response = httpHandler.postToAPI(shiftJson, PATH);
    return response;
}
```

Figure 5-6 AddShiftClient method postShift

To send the message the HTTPHandler class is called. It uses the RestEasy to post the request. From this point the created shift travels through the second tier of the system, business logic.

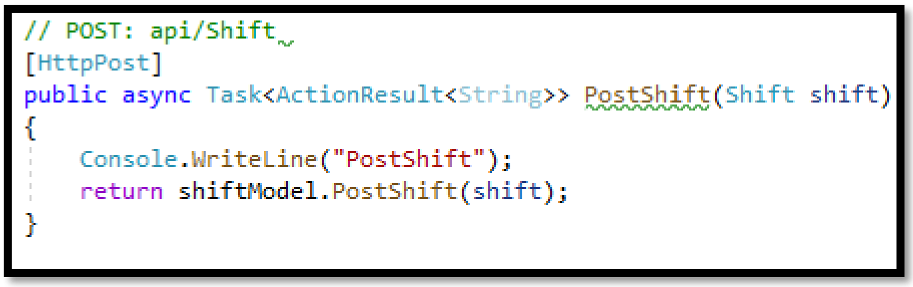


```
public String postToAPI(String json, String URL) {
    Response response;
    try{
        client = new ResteasyClientBuilder().build();
        target = client.target(URL);
        response = target.request().post(Entity.entity(json,  mediaType: "application/json"));

        if (response.getStatus() != 200) {
            throw new RuntimeException("Failed : HTTP error code : "
                    + response.getStatus());
        }
    } catch (ProcessingException e){
        e.printStackTrace();
        return "Server not responding";
    }
    return response.readEntity(String.class);
}
```

Figure 5-7 HTTPHandler method postToAPI

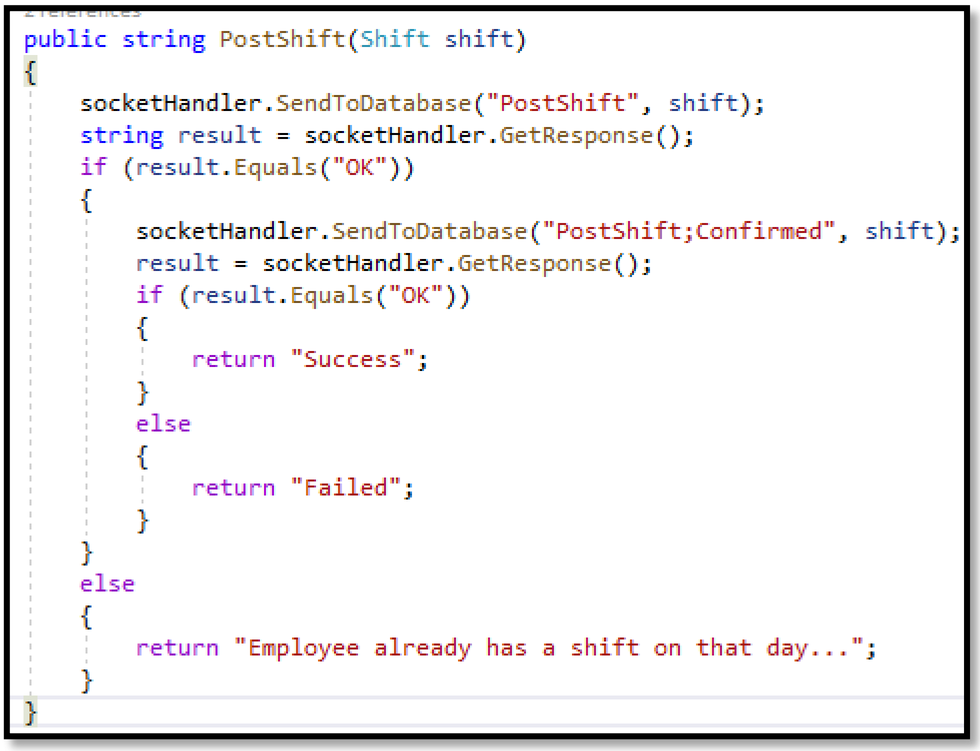
ShiftController determines which function the request should go to. When the shift JSON is received it is taken form the body of the request and deserialized into almost the same class that is defined in the java client of shift. There is only one difference, Java type of LocalDate is not interoperable with any of the C# data types. Therefore, had to create our own LocalDate type which has much less features but can correctly hold the values like the java class of LocalDate. The information is passed to a ShiftModel class.



```
// POST: api/Shift
[HttpPost]
public async Task<ActionResult<String>> PostShift(Shift shift)
{
    Console.WriteLine("PostShift");
    return shiftModel.PostShift(shift);
}
```

Figure 5-8 ShiftController method PostShift

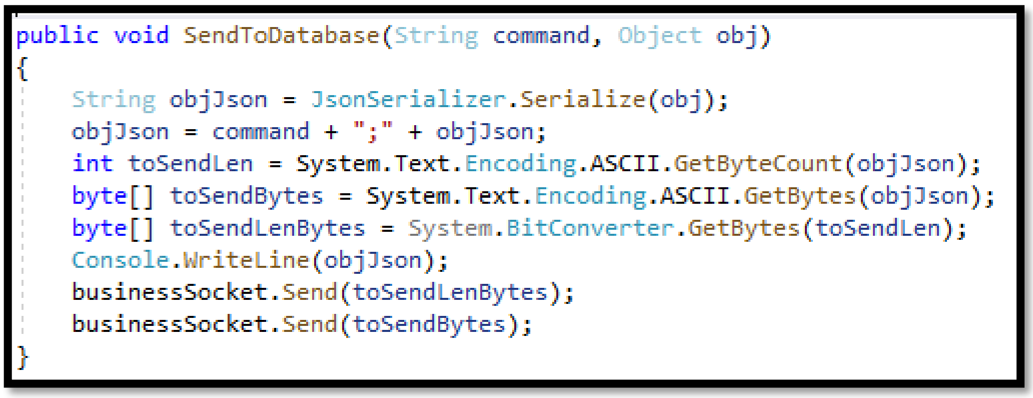
The model then performs a procedure where it makes two requests to the database – check and post. Check, the first message being sent, sends the shift along with a command name to the database and checks if the shift is already there, how this is done will be show in the database later. Once the response returns, if it is "NOT", the client will receive a response that the database already has this shift, if the response is "OK", it means it has the ability to continue with posting. Post is the second if statement which after receiving ok to post sends the same message again but this time ads a confirmation message. If it fails inserting it will return "NOT" as in not successful, else if it is "OK" it will return ok to the client and will change window after a successful insert into database.



```
2 references
public string PostShift(Shift shift)
{
    socketHandler.SendToDatabase("PostShift", shift);
    string result = socketHandler.GetResponse();
    if (result.Equals("OK"))
    {
        socketHandler.SendToDatabase("PostShift;Confirmed", shift);
        result = socketHandler.GetResponse();
        if (result.Equals("OK"))
        {
            return "Success";
        }
        else
        {
            return "Failed";
        }
    }
    else
    {
        return "Employee already has a shift on that day...";
    }
}
```

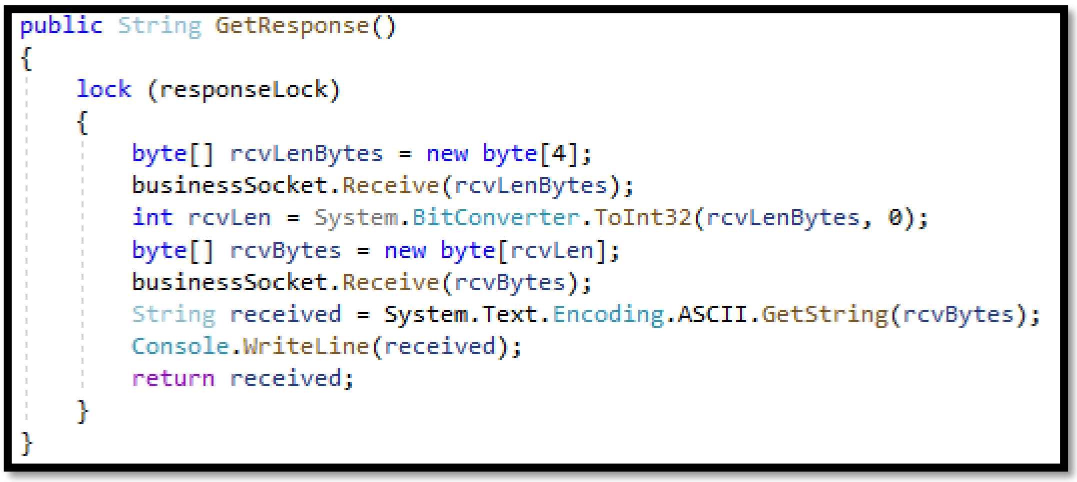
Figure 5-9 ShiftModel method PostShift

The BusinessSocketHandler takes care of all sending and receiving of information to and from the database. It turns the command and object into a JSON and then turns that into bytes before sending it. Receiving is a reversed send with an addition of a lock method, which locks the inside of the method when a thread begins using it so no other thread can use it and releases it when a thread is done with it.



```
public void SendToDatabase(String command, Object obj)
{
    String objJson = JsonSerializer.Serialize(obj);
    objJson = command + ";" + objJson;
    int toSendLen = System.Text.Encoding.ASCII.GetByteCount(objJson);
    byte[] toSendBytes = System.Text.Encoding.ASCII.GetBytes(objJson);
    byte[] toSendLenBytes = System.BitConverter.GetBytes(toSendLen);
    Console.WriteLine(objJson);
    businessSocket.Send(toSendLenBytes);
    businessSocket.Send(toSendBytes);
}
```

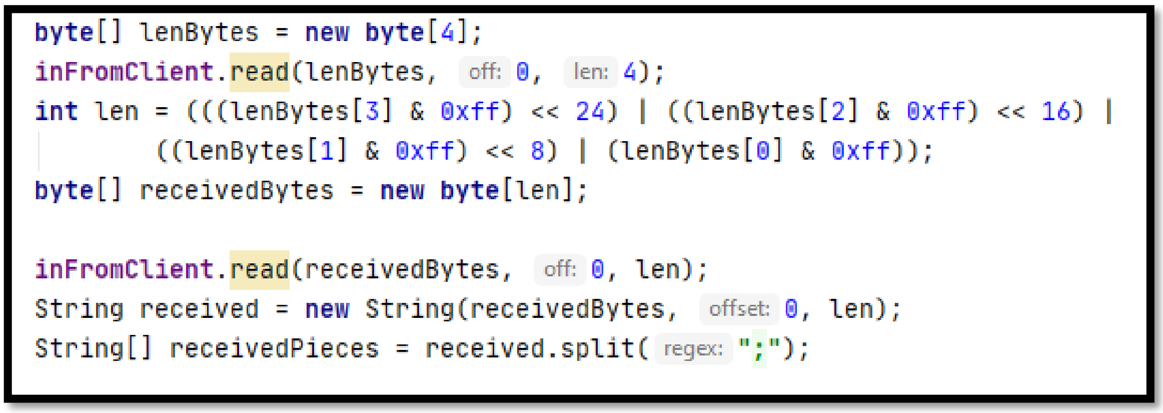
Figure 5-10 BusinessSocketHandler method SendToDatabase



```
public String GetResponse()
{
    lock (responseLock)
    {
        byte[] rcvLenBytes = new byte[4];
        businessSocket.Receive(rcvLenBytes);
        int rcvLen = System.BitConverter.ToInt32(rcvLenBytes, 0);
        byte[] rcvBytes = new byte[rcvLen];
        businessSocket.Receive(rcvBytes);
        String received = System.Text.Encoding.ASCII.GetString(rcvBytes);
        Console.WriteLine(received);
        return received;
    }
}
```

Figure 5-11 BusinessSocketHandler method GetResponse

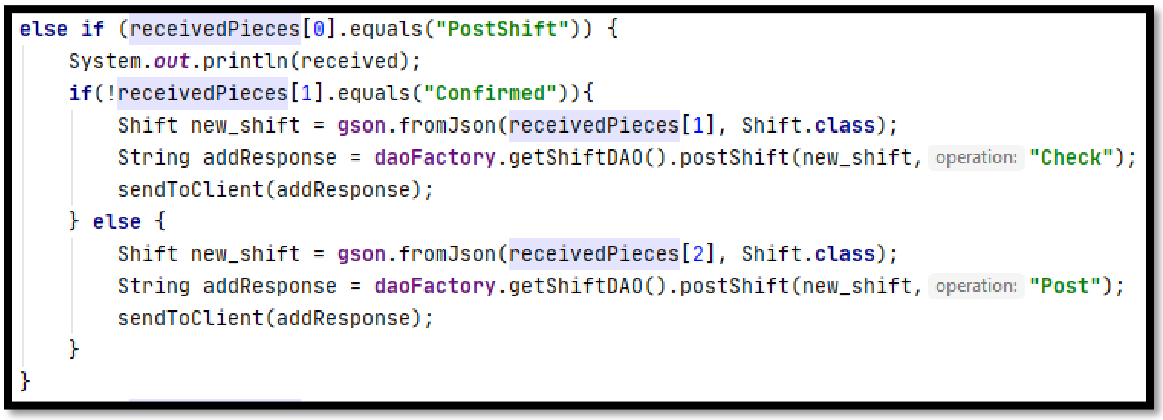
When the DatabaseSocketHandler receives a message it first turns it into a string. The string is split it into commands and content then it is sorted by the command name. If the request is valid it executes the code in the chosen command if statement. In this case the post request has an object attached to it that the method deserializes first and passes it to a DAO object that deals with shifts. When the if statement gets something back it sends a string back the same way it received it.



```
byte[] lenBytes = new byte[4];
inFromClient.read(lenBytes, off: 0, len: 4);
int len = (((lenBytes[3] & 0xff) << 24) | ((lenBytes[2] & 0xff) << 16) |
        ((lenBytes[1] & 0xff) << 8) | (lenBytes[0] & 0xff));
byte[] receivedBytes = new byte[len];

inFromClient.read(receivedBytes, off: 0, len);
String received = new String(receivedBytes, offset: 0, len);
String[] receivedPieces = received.split( regex: ";");
```

Figure 5-12 DatabaseSocketHandler method Run part one



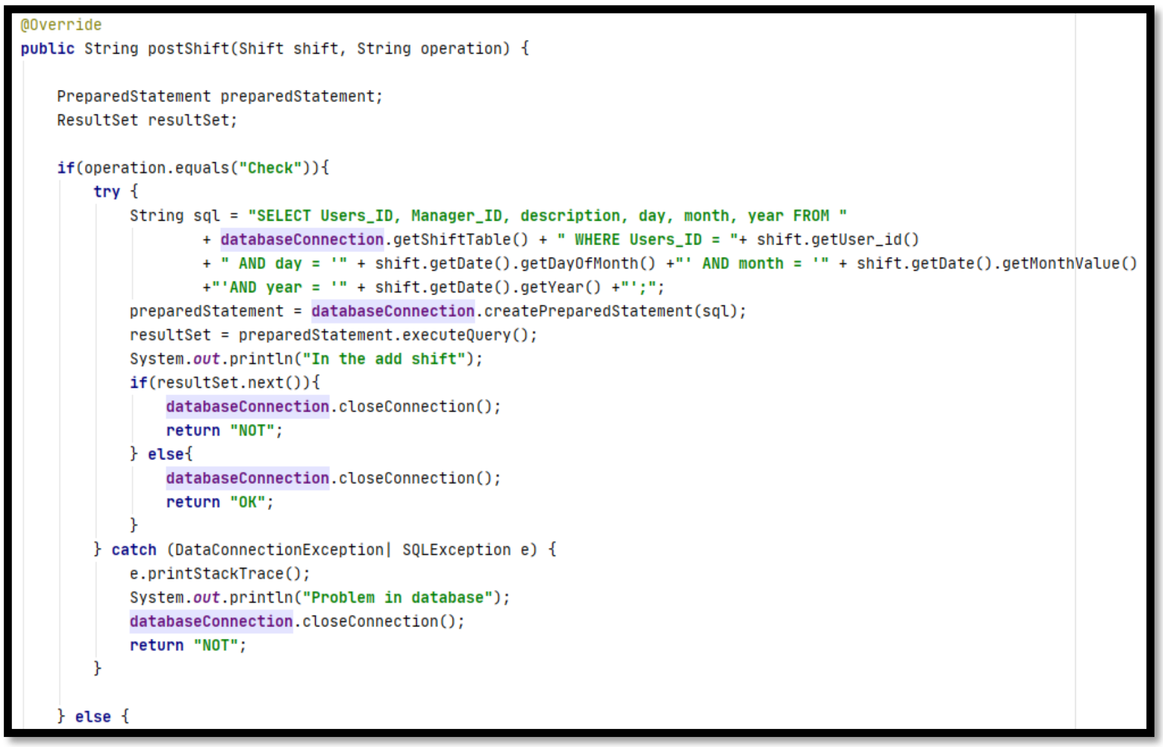
```
else if (receivedPieces[0].equals("PostShift")) {
    System.out.println(received);
    if(!receivedPieces[1].equals("Confirmed")){
        Shift new_shift = gson.fromJson(receivedPieces[1], Shift.class);
        String addResponse = daoFactory.getShiftDAO().postShift(new_shift, operation: "Check");
        sendToClient(addResponse);
    } else {
        Shift new_shift = gson.fromJson(receivedPieces[2], Shift.class);
        String addResponse = daoFactory.getShiftDAO().postShift(new_shift, operation: "Post");
        sendToClient(addResponse);
    }
}
```

Figure 5-13 DatabaseSocketHandler method Run part two

The ShiftDAO has two responsibilities in it and violates the solid principles. First if it gets “Check” operation it performs checking if the shift is there in the database already and second, if it gets anything else but a “Check” it inserts it to the database.

The check command finds a shift that has those parameters and instead of entering into a loop to check all of them if they are equal with the given shift it goes into an if statement that if there are results returns a “NO” back. This is done because database select statement is a good enough method to check if something is already existing. The most important values of the shift are the user id and date. A user cannot have more than one shift per day. Otherwise returns OK notifying to continue.

The post method executes the insert statement with values taken from the provided parameters of the shift. If the insert was "OK" a new shift appears in the calendar and shift list for the manager.



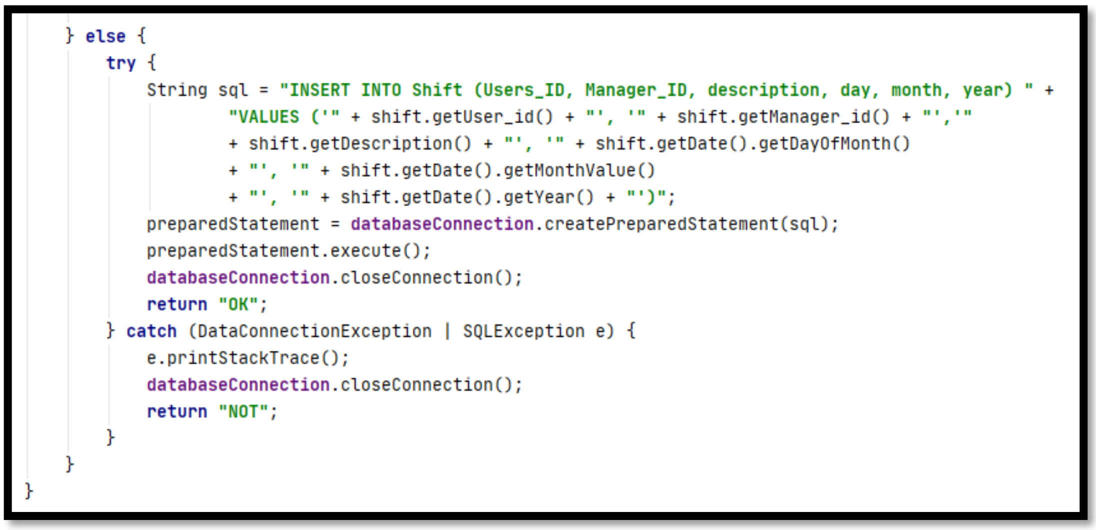
```
@Override
public String postShift(Shift shift, String operation) {

    PreparedStatement preparedStatement;
    ResultSet resultSet;

    if(operation.equals("Check")){
        try {
            String sql = "SELECT Users_ID, Manager_ID, description, day, month, year FROM "
                    + databaseConnection.getShiftTable() + " WHERE Users_ID = "+ shift.getUser_id()
                    + " AND day = '" + shift.getDate().getDayOfMonth() +"' AND month = '" + shift.getDate().getMonthValue()
                    +"'AND year = '" + shift.getDate().getYear() +"';";
            preparedStatement = databaseConnection.createPreparedStatement(sql);
            resultSet = preparedStatement.executeQuery();
            System.out.println("In the add shift");
            if(resultSet.next()){
                databaseConnection.closeConnection();
                return "NOT";
            } else{
                databaseConnection.closeConnection();
                return "OK";
            }
        } catch (DataConnectionException| SQLException e) {
            e.printStackTrace();
            System.out.println("Problem in database");
            databaseConnection.closeConnection();
            return "NOT";
        }

    } else {
```

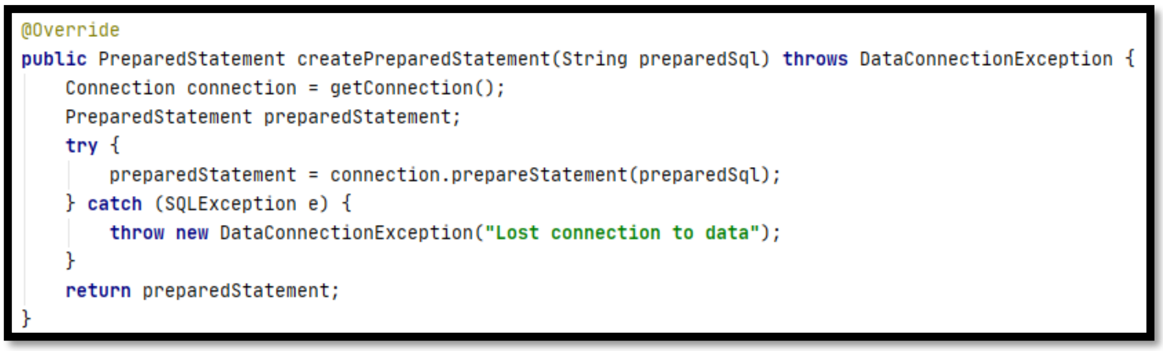
Figure 5-14 ShiftDAO method postShift checking part



```
    } else {
        try {
            String sql = "INSERT INTO Shift (Users_ID, Manager_ID, description, day, month, year) " +
                    "VALUES ('" + shift.getUser_id() + "', '" + shift.getManager_id() + "','"
                    + shift.getDescription() + "', '" + shift.getDate().getDayOfMonth()
                    + "', '" + shift.getDate().getMonthValue()
                    + "', '" + shift.getDate().getYear() + "')";
            preparedStatement = databaseConnection.createPreparedStatement(sql);
            preparedStatement.execute();
            databaseConnection.closeConnection();
            return "OK";
        } catch (DataConnectionException | SQLException e) {
            e.printStackTrace();
            databaseConnection.closeConnection();
            return "NOT";
        }
    }
}
```

Figure 5-15 ShiftDAO method postShift posting part

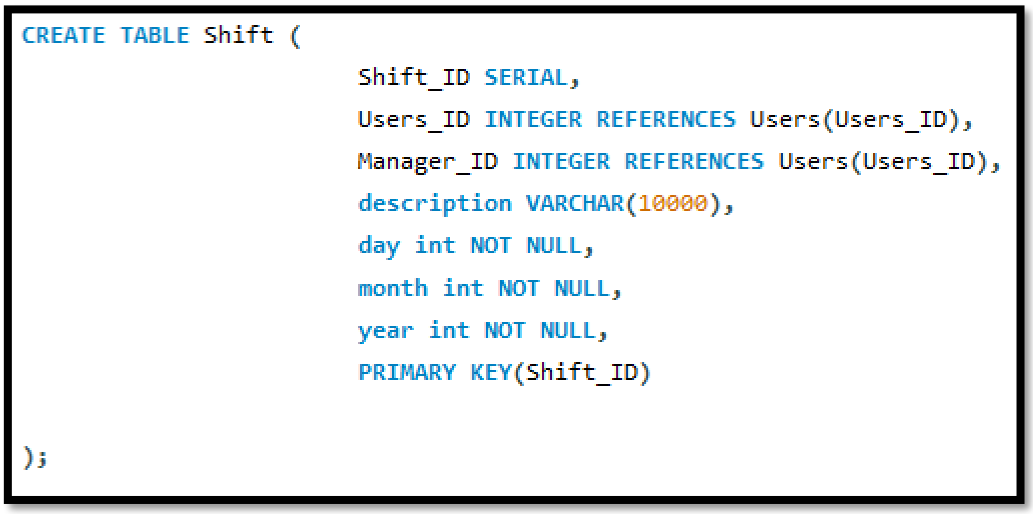
All the database executes are called through a prepared statement that is bound to a MySQL database.



```
@Override
public PreparedStatement createPreparedStatement(String preparedSql) throws DataConnectionException {
    Connection connection = getConnection();
    PreparedStatement preparedStatement;
    try {
        preparedStatement = connection.prepareStatement(preparedSql);
    } catch (SQLException e) {
        throw new DataConnectionException("Lost connection to data");
    }
    return preparedStatement;
}
```

Figure 5-16 DBConnection method createPreparedStatement

And the final step in this process happens in the database. A shift id is never set while going through the tiers. This step would create needless complexity of keeping track of what numbers are taken in each entity, so instead when the database creates a user it assigns a unique among shifts serial key.



```
CREATE TABLE Shift (
                        Shift_ID SERIAL,
                        Users_ID INTEGER REFERENCES Users(Users_ID),
                        Manager_ID INTEGER REFERENCES Users(Users_ID),
                        description VARCHAR(10000),
                        day int NOT NULL,
                        month int NOT NULL,
                        year int NOT NULL,
                        PRIMARY KEY(Shift_ID)

);
```

Figure 5-17 Shift table entity

6 Testing

The tests for the software were done using test cases in a black box testing environment. The black box environment means that the software was tested in a way that did not require any knowledge of the code or the architecture of the program. This means that the program could perform functional tests, using realistic test data, like a user would.

Practically, the tests were included as tasks for the SCRUM framework and were done at the end of each user story.
This was done to ensure that the functionality worked as it was intended to, and to re-assure the group moving on to the next sprint without the user stories of the previous sprint being left unfinished.

6.1 Test Cases

The test cases were made, in order to organize testing in an easy-to-understand method, so that it can be done by mostly anyone.
Test cases make it easy to execute tests, since they provide step-by-step instruction with test data.

6.1.1 Test case: Successful log in

Pre-Condition

None.

Step	Action	Reaction	Test Data
1	Open the system	The program starts	
2	Enter valid username		David
3	Enter valid password		test
4	Click log in	The user is logged in to the system	

Post-Condition

The user is logged in to system.

Test execution:

Working as intended.

6.1.2 Test case: Failed log in w/ wrong password

Pre-Condition

None.

Step	**Action**	**Reaction**	**Test Data**
1	Open the system	The program starts	
2	Enter valid username		David
3	Enter wrong password		system
4	Click log in	The program displays an error stating that the login failed	

Post-Condition

The user is not logged in to system.

Test execution:

Working as intended.

6.1.3 Test case: Adding a shift to the system with valid information

Pre-Condition

Must be logged in as manager and have connection to the calendar window.

Step	Action	Reaction	Test Data
1	Click on "Add Shift" button	Program opens add shift window	
2	Enter date		4/6/2020
3	Choose employee		Anders
4	Enter description		SEP Deadline
5	Click submit	Program displays message and returns to calendar window.	

Post-Condition

A new shift is added to the chosen employee in the calendar window.

Test execution:

Working as intended.

6.1.4 Test case: Adding a shift to the system with date only

Pre-Condition

Must be logged in as manager and have connection to the calendar window.

Step	Action	Reaction	Test Data
1	Click on "Add Shift" button	Program opens add shift window	
2	Enter date		4/6/2020
3	Click submit	Program displays error message.	

Post-Condition

A new shift cannot be added to the chosen employee in the calendar window.

Test execution:

Working as intended.

6.1.5 Test case: Cancel adding a shift

Pre-Condition

Must be logged in as manager and have connection to the calendar window.

Step	Action	Reaction	Test Data
1	Click on "Add Shift" button	Program opens add shift window	
2	Click "cancel"	Program exits add shift window and open the calendar window.	

Post-Condition

Exits to the calendar window.

Test execution:

Working as intended.

6.2 Test case results

- In total, 43 test cases were made and executed
- **All** the test cases passed

The remaining test cases can be found in Appendix G.

7 Results and Discussion

7.1 Fulfilled requirements

The overall results with reference to the user stories, which had been realized during our sprints can be seen below:

User story	Test results

“As a user I want to log in to the system, so that I can use it”	PASSED 7/7 TEST CASES
“As a manager, I want to be able to see a calendar, so that I can see which shifts need to be filled”	PASSED 4/4 TEST CASES
“As an employee, I want to see a calendar with my shifts, so that I can get an overview of when I need to work”	PASSED 4/4 TEST CASES
“As a manager, I want to be able to add shifts, so that I can assign work”	PASSED 6/6 TEST CASES
“As a manager I want to be able to add a description to a shift so that the employees know any necessary details about the shift”	
“As a manager, I want to add new employees to the system, so that I can assign them shifts”	PASSED 10/10 TEST CASES
“As a manager, I want to be able to delete shifts so that unneeded shifts do not appear”	PASSED 3/3 TEST CASES
“As a manager, I want to remove employees from the system, so that I can get rid of inactive workers”	PASSED 3/3 TEST CASES
“As a manager, I want to see a list of employees which I supervise, so that I can get an overview of who is available for work”	PASSED 3/3 TEST CASES
“As a manager, I want to have information about employees available so I can view the employee's personal information”	

In total, **10 out of 21** user stories were implemented.

7.2 Program features

The program makes it possible for both employees and managers who use the system to keep track of their shifts, and managers to manage said shifts.

In order to identify the users of the program, a log in function was made. If provided with valid credentials, the function authenticates the user through a web API in a database server. The password is hashed in the client, so that the other system layers have no knowledge of the user-credentials.

Past user-authentication, different user types are given different permissions and different information displayed to them.

Employee level users, from here on known as “employees”, have access to a calendar overview of which shifts they themselves are assigned to.

Manager level users, from here on known as “managers”, also have access to the calendar, as well more features.

Those features include:

- Seeing shifts for every employee which they manage
- Access to a list of shifts for employees which they manage, from which they can delete shifts from the system
- Creating more shifts by specifying a date, assigned employee and optionally specifying a description
- Creating new user accounts by specifying username, password, first name, last name, e-mail, status as well as access level
- Access to a list of employees, from which they can view employee information
- Deleting existing user accounts from the beforementioned list

The product of the project is mostly a success, since the core functionality has been successfully implemented.

Despite that not all the user stories were carried out, the 10 most important ones were implemented and are working as intended according to the test cases. Therefore, it is reasonable to believe, that given more time and more sprints, the remaining user stories could also be successfully implemented.

8 Conclusions

The goal of the project was to create an employee and shift planning system, which would make it easy for company managers and employees to have an overview of their coming shifts, and to manage them.

The project was made based on 21 user stories, which were created through the perspective of the system users in order to decide the required program functionality.

In the analysis section, the user stories were created. Alongside that, it features a use case diagram, which shows how the system actors relate to the user stories.

According to each user story from the use case diagram, use case descriptions were made to describe in further detail how the user would interact with the program in each scenario. A domain model was made in order to identify early conceptual classes and the associations between them, which helps transition into system design.

Lastly, the analysis section includes a thorough assessment of security, including a threat model and an analysis of security risks.

The system design starts off explaining the overall program architecture, GUI choices as well as the overall user functionality.

The section then goes into detail as to which software design patterns were used, including MVVM for the client and Data access objects for the database. It also includes an assessment of system security.

In the implementation section, the most interesting classes and pieces of implementation are explained in detail, and further expanded upon. It also includes a sequence diagram of the implementation of the “add shift” function, which helps visualize how the system classes interact with each other throughout the program.

In the testing section, the test methodology for the system is explained. The section also features test cases, which explain step-by-step how the tests were executed for each user stories. The results of the test cases show that all 43 passed.

Finally, in the “results and discussion”, the system tests are compared to the user stories, to see whether they have been successfully implemented. The results show that 10 out of 21 user stories were completed, which given the timeframe and total amount of user stories, suggest that with more time and sprints, the remaining user stories could be finished.

9 Project future

As of now software has been produced with its core functionality. There are 11 unfinished functional which would be implemented if given more sprints to develop.
First what should be changed is label for displaying date and shift in calendar view. That would be changed to list view where individual shift could be shown and accessed.

In the future there might be a possibility to add salary, which would be automatically calculated, to ease up work for manager. Full time workers would have set salary and designated shift and part time workers would get their final salary at the end of the month which would be calculated from worked hours and hourly pay.

GUI should be improved. As of now the GUI is not user friendly, the focus was functionality over aesthetic GUI design. This would increase overall value of the product for the customer.

Last, but not least, an improvement could be to add an interface for a mobile app. This will give more freedom to user to choose which platform they would prefer to use. And would be handy if manager or employer were to send some important message for everyone to view.

10 SDJ Reflections

10.1 Anders Sønderby Sørensen

10.1.1 Distributed systems

Distributed systems were a big part of the SDJ3 curriculum. They are systems whose elements reside on different networked machines (machines ranging from anything from physical computers to software-level containers). This divides the different systems into independent machines, who work together to provide a feature or functionality.
SDJ3 introduces several types of distributed systems, but the ones which directly relate to SEP3 are described below:

10.1.1.1 Client/Server

One of the most common distributed systems found today is the client/server structure. Functionality for features and data are hosted on machines referred to as “servers”, whereas the machines who request the features (for example a client can request a server for data used in a client presentation tier) are “clients”.

10.1.1.2 3-tier architecture

The 3-tier architecture takes its start in the client/server structure and further expands it with another tier. This allows for a dedicated presentation tier, a dedicated application/business logic tier as well as a data tier. The 3-tier architecture is common with web applications and web services.

10.1.1.3 SEP3 Architecture

The architecture used for SEP3 is a 3-tier architecture as described above. It uses different technologies, with different programming languages to show the flexibility of a 3-tier architecture, which in this project is divided in the following way:

- Presentation tier = JavaFX (Java)
- Application logic tier / Middleware = ASP .NET (C#)
- Data tier = JDBC (Java)

The presentation tier / client requests functionality through the business logic server, which then independently retrieves data from the data tier.
Since the tiers are fully separated, the client knows nothing of what goes on in the middle tier, except in which format to expect responses (in our case the format chosen was JSON).

3-tier architecture vs. client/server structure

The downside of the project being developed as a 3-tier architecture rather than a 2-tier architecture (client/server) is, among other things, added complexity and development time, as well as latency.
If the project had been made as a client/server system, it would still have required all of the 3 aspects, however the application logic would have to be split between the client and the server. The program would in theory be faster, since it wouldn't have to go through the middleware.
Practically, the difference could be minimal, depending on "outside" factors such as internet speed and physical server locations.
One of the major upsides to having made the project with a 3-tier architecture is easy scalability. In a professional setting, upgrading or adding more servers is significantly easier, because of the forced separation of GUI logic and business logic.

10.2 Dat Phuc Nguyen (David)

Heterogenous system was requirement set by course description and the software was built around that. The software has to be developed in more the 1 programing languages. And it needs to be able to communicate in between the different languages. The advantage of making heterogenous system is that the software is open which means that it can be reimplemented in various ways and its mobile which means that it can be transferred from one computer to another. The middleware in this system is in C# (business logic) it’s a connecting layer which accommodates to other parts of the system and helps them communicate in between each other.

Three tier architecture was chosen as software architecture for this software. It consists of 3 tiers where each tier has its own function. Client where GUI/views are, Business logic where all the specific processing is being made and data tier where the application stores all the data. This has been chosen because of its scalability. And same goes for data tier where more space can be allocated/purchased. With 3 layers we can run each layer on different platforms.

Distributed system was chosen because the employee planner is supposed to have more users who should be able to access their shift or get notification on different locations via different networked computers. The communication is being managed by our business logic server. And client request services based on what is needed for current user. The system sends messages in between each computer in the network.

Since the software is sending messages in this case JSON between each tier. RESTEasy web services were used as mean of transferring request and responses between Client and server. Its chosen over Spring because the simplicity of the requests. Client request over HTTP and awaits response from server. The server has the methods for operation (GET, POST, DELETE, …)

Last requirement for this software was to have sockets. Which are allocated in between business logic server and data tier. It is a TCP connection based which allows to have multiple connections between server and the data tier.

Maven was used to get dependencies without need to alter libraries and import files into the project structure. Such as RESTEasy, Gson, MySQL dependencies. This avoided unnecessary problems with versions if used on another computer.

10.3 Niklas Krogh Jensen

In the project a 3 – Tier Architecture have been constructed as the fundamental backbone, for conducting a distributed system with different programming languages. When engaged with multiple programming languages, it is referred as a heterogeneous system. A heterogeneous system is generally more complex, due to difficulty of maintenance and communication between different platforms and programming languages, furthermore such system can be supported properly with a 3 – Tier architecture.

The 3 – Tier Architecture is distributed beyond different layers; A presentation, Business Logic and Data Access layer, which can be referred to the Client (presentation), Business Logic and Data Access in the Semester Project.

Java was selected as the Client due to the incorporation of RESTEasy, which is explained into further details in the section 'RESTEasy'. C# was selected as the middleware due to an integrated API management framework using the ASP.NET Core, since it supports RESTful services, additional initiation is explained in the section 'Busisnees Logic'.

During creation of the Client and Data Access layer, Maven was chosen as the project's build structure, so a POM.xml file was prebuilt for the use of dependencies. Maven's provided POM file was used for simplifying instantiation of RESTEasy and Gson, because when a dependency is implemented into the POM file Maven automatically downloads the library. This meant that the libraries did not need to be incorporated into the project manually, and every user of the project would have the ability of usage.

10.3.1.1 Client

The Client provides a graphical UI representation with the use of JavaFX and MVVM architectural pattern for separation, such the different classes in the Client becomes independent and therefore uses single responsibility and interface segregation. Separation of layers within the client provides an easier implementation, while having an overview of where different methods are located.

10.3.1.2 RESTEasy

Connection between the Client and Business Logic used an HTTP protocol, through the RESTEasy (also referred as an innovated JAX-RS for clients) web services for sending and retrieving serialized JSON objects. The RESTEasy was chosen for the project due to the simplicity, intuitive and straight forward methods it provides. In comparison to another well used API as Spring, RESTEasy only needs a specification in the dependencies to access implementation and uses standard annotations which is already a part of Java EE. Where Spring is a complete RESTful implementation while constructing a new project, additionally Spring does not use standard annotations, but custom.

Secondly, the project would only be sending a retrieving limited and simple serialized JSON objects using the POST and GET annotations. Therefore, a complete implementation of RESTful would not be needed including the custom annotations. With that said RESTEasy would be a more suitable candidate as the web service.

10.3.1.3 Business Logic

When handling the requests from the Client, controllers are essential for retrieving and sending the serialized JSON further to the Data Access layer. The controller classes were preconstructed with a starter controller when derived from a ControllerBase class. Sockets was used as the connection between the Business Logic and Data Access since sockets uses a TCP connection across networks.

10.4 Rokas Barasa

The system that was developed is heterogeneous. A heterogeneous system uses different languages, different operating systems, or different protocols of communicating. A 3-tier architecture was also used in our system. A 3-tier architecture has:

- A presentation tier. Focuses on handling the user and the view.
- A business logic tier. Handles application specific processing.
- A database tier. Storage of system data and database management

The benefits of having a 3-tier system: the system is divided into layers of functionality making the system less complicated to change, development can be done separately, since it provides scalability and reusable. All of this was combined to make a system that used java for the presentation tier, C# for the business logic tier, and java for the database. Having a system in such a structure lets us use the strengths that each coding language provides. Java has a good way of doing a user interface, while C# has a very good API support that makes it easy to set up. The system can be transferred from one computer to another without problems. The negatives of having a 3-tier system is because it has noticeable delays and has to be secured at the two different connection points connections.
The system is a distributed system. As it passes messages through a network to coordinate functionality. It has a server and a client that share resources.
On one of the networks, the client connecting to the business logic server, http webservices were used. The http protocol is a TCP connection that operates on requests that are made through URL's. Information is transported over JSON, xml or other data interchange formats. The connection is alive when a client is sending a request to the server, until the client receives feedback form the server. The two main types of request that are used most are post and get. If no request is being made the client and server are not connected to each other. There are two different http protocols available, rest and soap. Rest was chosen because it is light weight and did not have a learning curve. Soap would have been too big to handle for a system that does not need much complexity. The http connection between business logic and client was placed there, due to the lack of connection between the server and client when no request is happening. It lowers the load on the server when it does not have to maintain connections to

many clients when they are idle. For this RestEasy was used. RestEasy was chosen because it lightweight and easy to use. It was our second choice as a http client. First it was considered using spring, but after realising that it would not need any other features spring provides it was decided to keep the system simple and go for something that works reliably and easily.
A MVVM architectural pattern was used on the first tier, the presentation tier, of the system to organise it by functionality.

11 Sources of information

Banger, D., 2014. A Basic Non-Functional Requirements Checklist « Thoughts from the Systems front line.... Available at: https://dalbanger.wordpress.com/2014/01/08/a-basic-non-functional-requirements-checklist/ [Accessed January 31, 2017].

Business Analyst Learnings, 2013. MoSCoW : Requirements Prioritization Technique — Business Analyst Learnings. , pp.1–5. Available at: https://businessanalystlearnings.com/ba-techniques/2013/3/5/moscow-technique-requirements-prioritization [Accessed January 31, 2017].

Dawson, C.W., 2009. *Projects in Computing and Information Systems*, Available at: http://www.sentimentaltoday.net/National_Academy_Press/0321263553.Addison.Wesley.Publishing.Company.Projects.in.Computing.and.Information.Systems.A.Students.Guide.Jun.2005.pdf.

Gamma, E. et al., 2002. *Design Patterns – Elements of Reusable Object-Oriented Software*, Available at: http://books.google.com/books?id=JPOaP7cyk6wC&pg=PA78&dq=intitle:Design+Patterns+Elements+of+Reusable+Object+Oriented+Software&hl=&cd=3&source=gbs_api%5Cnpapers2://publication/uuid/944613AA-7124-44A4-B86F-C7B2123344F3.

IEEE Computer Society, 2008. IEEE Std 829-2008, IEEE Standard for Software and System Test Documentation,

Larman, C., 2004. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development,

Mendeley.com, 2016. Homepage | Mendeley. Available at: https://www.mendeley.com/ [Accessed February 2, 2017].

YourCoach, S.M.A.R.T. goal setting | SMART | Coaching tools | YourCoach Gent. Available at: http://www.yourcoach.be/en/coaching-tools/smart-goal-setting.php [Accessed August 19, 2017].

12 Appendices

The purpose of your appendices is to provide extra information to the expert reader. List the appendices in order of mention.

Examples of appendices

- Project Description
- User Guide
- Source code – source documentation
- Diagrams
- Data sheets
- Etc.

Appendix A Project Description

Insert the original Project Description here

Employee Planner

Process report

Anders Sønderby Sørensen – 264247

Dat Phuc Nguyen (David) – 251771

Niklas Krogh Jensen – 281335

Rokas Barasa – 285047

Supervisors:

Joseph Chukwudi Okika (JOOK)

Nicolai Sand (NISA)

47.049

Software engineering

3rd semester

04.06.2020

Table of content

1 Introduction

This report is a description and documentation of the process of our study group on our semester project. The project is part of the curriculum of the 3rd semester of the Software engineering education at VIA University College.

The project includes elements of several of our courses during the semester, including:

- SDJ3 (Software Development of Distributed Systems)
- DNP1 (Internet Technologies, C# and .NET)
- NES1 (Networking and Security)

The goal of this semester was to utilize a 3-tier architecture to develop a distributed, heterogeneous system, as well as discuss the security aspects of the project. An employee/shift planning software was chosen by the group to be the product.

The project began in February 2020, with one day per week allocated to planning the project, until the 13th of May, from which 3 full working weeks were allocated to working on implementation and finishing the project.

The process for developing the "employee planner" was well-planned and well-executed, thanks to the agile methodology framework SCRUM, our issue-tracking tools, as well as our experience with agile development from the previous semesters.

2 Group Description

Our study group was created at the start of the 3rd semester. Dat and Niklas were working together before on 1st and 2nd semester. Rokas and Anders have not experienced study group work with any of current students in the group.

This did not prove to be huge challenge because the group has been chosen based on goal which each of the members wanted to achieve. Which in this case was to get good grade and work in efficient team.

2.1 Group members

Anders AS	Danish, 24 years old, background in IT-support, software engineering student at VIA UC
Dat (David) Phuc Nguyen DN	Czech (Vietnamese), 23 years old, background in Marketing management and currently student of Software engineering in VIA UC
Niklas NK	Danish, 24 years old, student of ICT – Software engineering VIA UC
Rokas RB	Lithuanian, 20 years old, student of ICT – Software engineering VIA UC

2.2 Culture

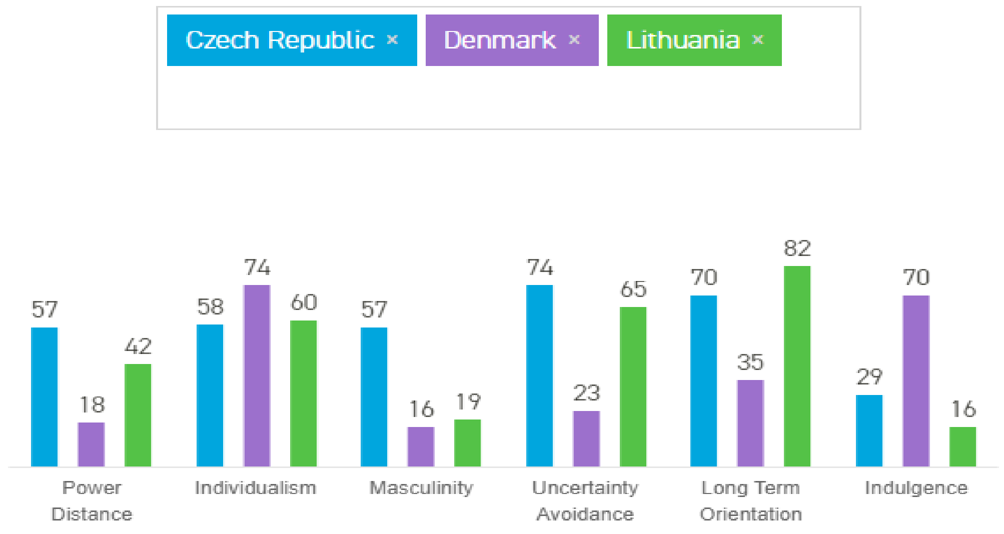


Figure 2-1 Hofstede Insights

The representatives of group originate from 3 countries. Czech Republic, Denmark and Lithuania. As show up above on figure 2-1, there are large differences between Denmark and the two other countries. Only Individualism seems to be similar between represented countries. But these do not represent the team member's personal cultural preferences.

The cultural difference is not that noticeably different between the members of the study group. DN has been living in Denmark for over 3 years and this has helped to cope with cultural differences between him and the rest of the group and last semester he was working in study group consisting of three Danes and him. RB is the most different in comparison to other team members. He is younger than the other team members and lived in Denmark for approximately 16 months. He has proved to be very adaptive but sometimes needs more than one explanation for other members to understand his view/message.

2.3 Member profiles

Personal profile insight has been done in 1st semester and taken as source of personal differences of each individual.

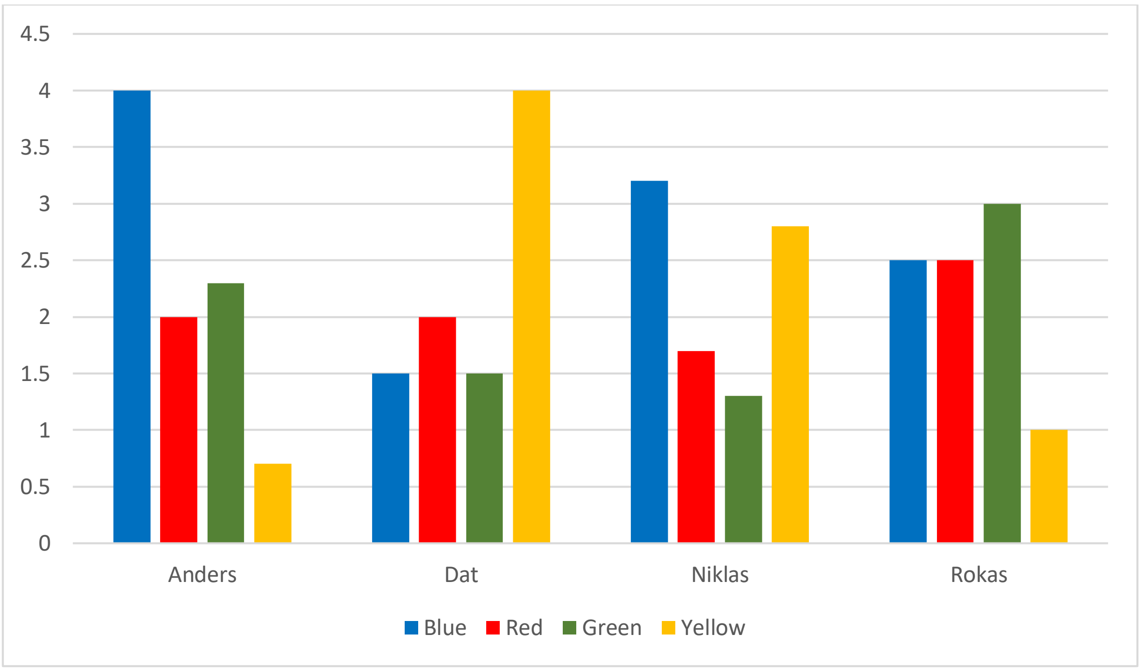


Figure 2-2 Personal profiles

Significant blue: Describes people who protect values, are careful and work accurately and methodically. The color preference is divided into two facets: Reserved and controlled.

Significant red: Describes people who want to win, achieve results and goals, both for themselves and for the group. The color preference is divided into two facets: Self-focus and Power.

Significant green: Describes people who seek companionship and cooperation, focus on others and feelings. The color preference is divided into two facets: focus on others and emotion oriented.

Significant yellow: Describes people who seek attention, are extroverted and innovative. The color preference is divided into two facets: outgoing and innovative.

Figure 2-2 Provides an overview of each group members color profiles.

2.4 Group roles

RB – Scrum master, Developer

DN – Product owner, Developer

NK - Developer

AS – Developer

Group roles has been assigned since beginning of the project period when we decided on which idea for project we wanted to do. Since we decided to go with DN’s idea for project he has been selected as product owner. And Scrum master was given to RB since his experience with it was limited last semester and he wanted to try it. AS and NK were set as developers by default.

3 Project Initiation

When the initiation phase started the project, our supervisors had beforehand provided the group with certain requirements and expectations which had to fulfilled. These requirements were as listed:

- Create a distributed system that adheres a 3-tier architecture (Heterogeneous).
- Use Java and C#.
- Consume and expose one or more web services.
- Design and use a protocol for sockets.
- Create a GUI for each client.
- Specific security considerations.

All the requirements mentioned above had to be implemented as a part of the structure to fulfill the requirements.

During the project initiation various ideas had been discussed and drafted to ensure a qualified and sufficient approach according to the requirements. Our group chose two ideas which both was qualified and accepted by the supervisors and consisted of all the given requirements and expectations.

3.1 Chat system

The chat system was an idea from the whole group, since we already had experience with such a system just in a small form factor. The system was thought as a generic chat system with a twist of modern tendencies. These modern twists would be consisting of specific timed messages, shared files, pictures, and gifs which would be stored on the database for a limited time or permanent. Providing the user with customizability to their preference. The system would have some similarities with other chat system providers, such as: Discord, Snap Chat, previous MSN etc.

3.2 Employee Planner

The employee planning system was a previous project idea from the 2nd semester project and originally thought by David. The planning system would provide a variety features to increase the scheduling productivity for small or medium organizations. Overall, would the system consist of a calendar to provide an overview of each employee's monthly shifts and working schedule.

The system will supposedly contribute to easier handling of shifts if comprehended situations occurred for an individual employee for attending work. The employee would have the ability to release their shift, while a different employee would have the option to request the released shift without informing the manager nor team leader. Such ability provides a safety towards miscommunication between coworkers, manager, or team leaders.

3.3 The Selected Project

After further discussion back and forth on the two project proposals, we decided to discard the chats system, due to lack of background knowledge corresponding to the storage of files on the database. Because neither of the group members had experience nor worked with such data information storage.

The employee planner was be chosen as our initial project, since every group member had a proper overview and idea how it could be implemented, additionally did it seem straight forward. The individual ideas were then deliberated firmly to ensure a common perspective and overview, which provided the group with a further structured implementation, architecture, and development.

4 Project Description

In the project description phase, we had to turn our project proposal into a more realistic project description, detailing among other things our project methodology, a proposed time schedule as well as a risk assessment.

One of the first things we had to consider as a group, was our initial expectations and thoughts about the project.

Those first meetings helped make sure that the group collectively had the same goal and the same idea of how the product should function. This, in turn, helped our communication later in the project development.

During the project description period, we had good help from our project supervisors regarding specific parts of the project description hand-in, which helped us finish the first draft of the project description.

After submitting the first draft, we received constructive feedback from our project supervisors, and, although not much needed to be improved, it was still significant enough that it forced us to get together and talk about how to approach the project, specifically in relation to SCRUM and how we would structure our sprints.

5 Project Execution

While approaching the execution phase of the project, a selection of appropriate software tools where chosen to manage, preserve and control progression, files, communication, and code.

5.1 Tools

Several tools where used throughout the process:

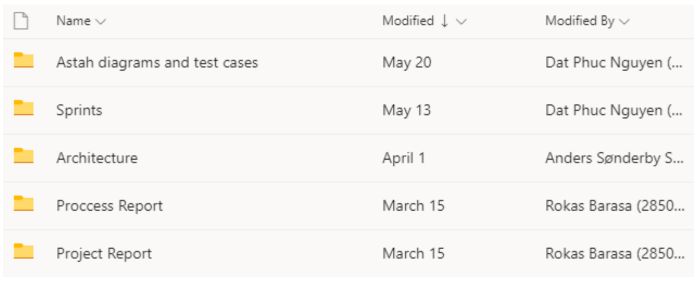
- Microsoft Teams combined with SharePoint was chosen as the central file storage.
- Discord was chosen as the platform of chat, talk conventions and screensharing.
- GitHub was chosen as version control, source code and central unit for code implementation.
- Jira was chosen as the agile management during the project.

5.1.1 Microsoft Teams and SharePoint

Microsoft Teams and SharePoint where primarily chosen since they are integrated and part of the subscription onto the Microsoft account VIA provides. This eliminated a creation of various school related or use of private accounts on different third-party providers e.g. Dropbox, Google Drive, Slack.
Furthermore, Microsoft Teams is constructed with SharePoint as a native integration which provides a single access point to file editing, sharing, or multitasking within the same file.

5.1.1.1 Microsoft Teams

Microsoft Team were used as an interface for setting the folder structure on SharePoint and keeping the structure simple and sturdy, as seen in figure 5-1.



Name	Modified ↓	Modified By
Astah diagrams and test cases	May 20	Dat Phuc Nguyen (...
Sprints	May 13	Dat Phuc Nguyen (...
Architecture	April 1	Anders Sønderby S...
Proccess Report	March 15	Rokas Barasa (2850...
Project Report	March 15	Rokas Barasa (2850...

Figure 5-1Microsoft Teams folder structure

The integration with SharePoint on the platform, made it possible to constantly being able to retrieve the newest changes to a file.

5.1.1.2 Microsoft SharePoint

The SharePoint folder where automatically created with the same structure as Teams:

- **Appendix**

  Stores all the folder and files needed for the appendix.
- **Architecture**

  Stores the hand-in of the architecture and Proof of Concept.
- **Archive**

  Stores redundant files as a backup.
- **Astah diagram and test cases**

  Stores all the diagram for the project and test cases on the project.
- **Backlog**

  Stores the backlog of requirements.
- **Finished**

  Stores the final and finished hand-in files, both as editable and non-editable.
- **Process Report**

  Stores the Process report.
- **Project description**

  Stores the project description.

- **Project Report**

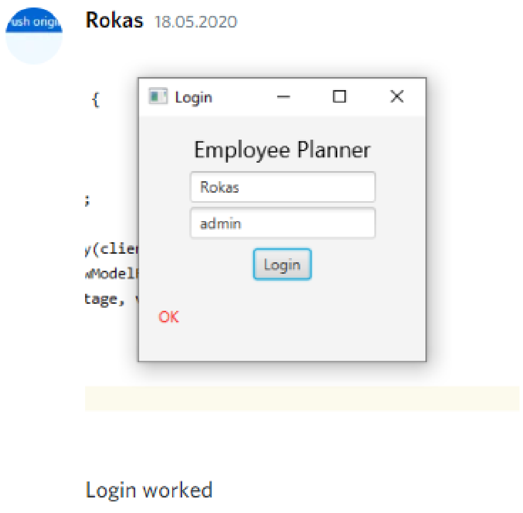
  Stores the project report.

- **Sprints**

  Stores SCRUM's daily, pre-planning, review, and retrospective meetings.

5.1.2 Discord

Discord were used as a communication platform to keep the team updated on the process. Whenever a task was completed, a bug or flaw occurred, or general information was needed, it was written in a private united chat for our group. As seen in figure 5-2, when a team member posted updates.

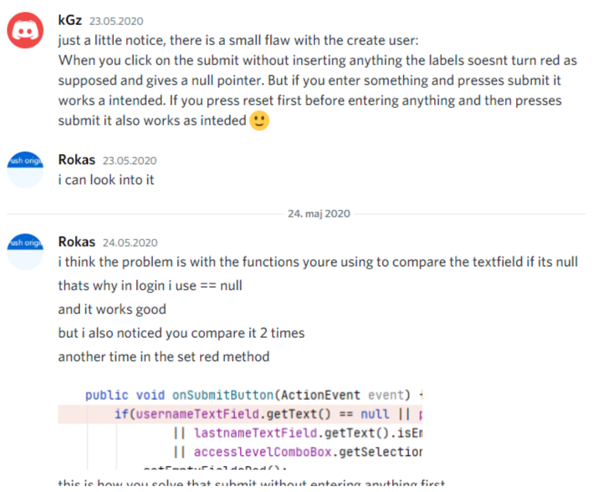

Rokas 18.05.2020

```
{
;
y(clie
wModel
tage,
```

Login worked

Figure 5-2 Team communication

Discord were also used to arrange when a meeting occurred such as daily scrum, review, retrospective or just a general meeting of discussion due to a complication. When a generic complication occurred on a specific task and further information were needed from another group member, individual chat was typically the better option. During the individually chats screen sharing could be enabled to solve an actual code problem or if elaboration were needed on a specific fragment. As seen in figure 5-3 a conversation between two group members with a concerned problem being discussed with potential fix.


kGz 23.05.2020
just a little notice, there is a small flaw with the create user:
When you click on the submit without inserting anything the labels soesnt turn red as supposed and gives a null pointer. But if you enter something and presses submit it works a intended. If you press reset first before entering anything and then presses submit it also works as inteded 🙂

Rokas 23.05.2020
i can look into it

24. maj 2020

Rokas 24.05.2020
i think the problem is with the functions youre using to compare the textfield if its null
thats why in login i use == null
and it works good
but i also noticed you compare it 2 times
another time in the set red method

```
public void onSubmitButton(ActionEvent event)
    if(usernameTextField.getText() == null || 
            || lastnameTextField.getText().isE
            || accesslevelComboBox.getSelectio
        setEmptyFieldsRed();
```

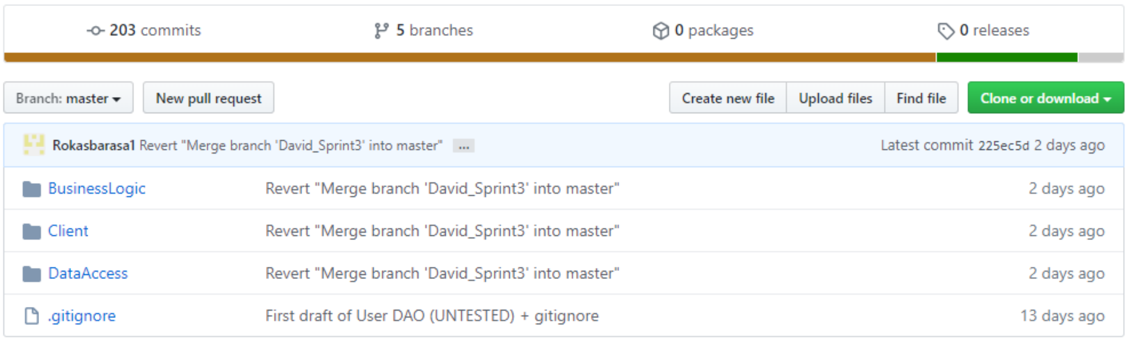
this is how you solve that submit without entering anything first

Figure 5-3 Individual chat

5.1.3 GitHub

GitHub were used as a platform for version controlling, in case of coding conflicts appeared it could be reverted or backtracked to what and where the problem was caused.

A repository was created for all group members to join on GitHub, were four individual branches was created as sub-branches to a master branch. The sub-branches would be named



203 commits | 5 branches | 0 packages | 0 releases

Rokasbarasa1 Revert "Merge branch 'David_Sprint3' into master" ... Latest commit 225ec5d 2 days ago

BusinessLogic	Revert "Merge branch 'David_Sprint3' into master"	2 days ago
Client	Revert "Merge branch 'David_Sprint3' into master"	2 days ago
DataAccess	Revert "Merge branch 'David_Sprint3' into master"	2 days ago
.gitignore	First draft of User DAO (UNTESTED) + gitignore	13 days ago

Figure 5-4 GitHub structure

as the assigned member's name, those branches were only for development purposes. Opposite to the master branch, which would only consist of fully developed and bug-free code. Figure 5-4 displays the structure of GitHub.

5.1.3.1 Procedure

When a group member started on an assigned task, their personal branch needed to be updated from the master branch to ensure the newest and updated code before development started. Whenever a task was fully functional and bug-free, the group member should merge their branch with the master to ensure others could retrieve the newly developed code.

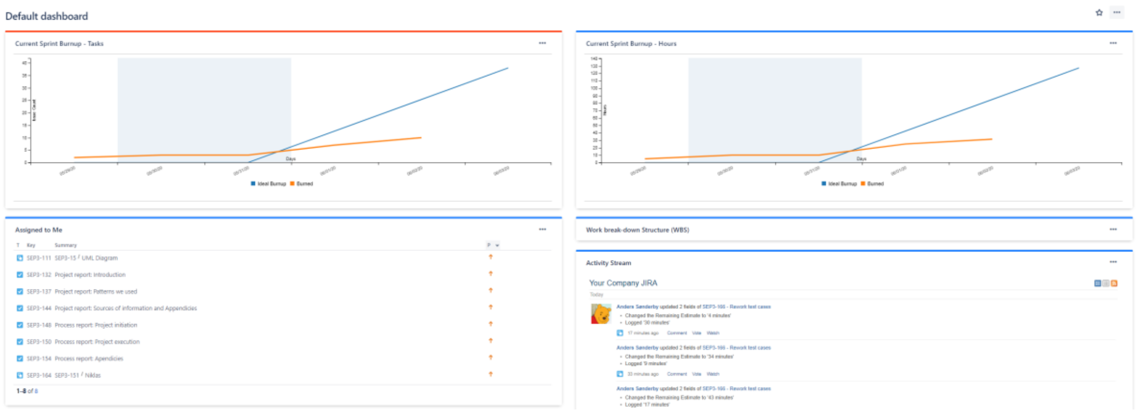
If conflicts occurred during the merging process, it was up to the group member who was merging to solve those conflicts.

5.1.4 Jira

Management of the agile process were organized with Jira Software by Atlassian since its implemented tools for management and structure of an agile framework as scrum and use of the Work Breakdown Structure. While providing a highly customizable process overview either from the Jira itself or third-party applications.

5.1.4.1 Dashboard

Jira's dashboard was customized by the team to ensure well functional generic burn up chart for the initiated sprint. Each group member had their own personal dashboard for showcasing the ongoing sprint task assigned to them, an activity stream to view the newest changes. As seen in figure 5-5 displays a personal dashboard.


Default dashboard
Current Sprint Burnup - Tasks
Days
Ideal Burnup
Burned
Current Sprint Burnup - Hours
Hours
Days
Ideal Burnup
Burned
Work break-down Structure (WBS)

Assigned to Me

T	Key	Summary	P
	SEP3-111	SEP3-15 / UML Diagram	
✓	SEP3-132	Project report: Introduction	
✓	SEP3-137	Project report: Patterns we used	
✓	SEP3-144	Project report: Sources of information and Appendicies	
✓	SEP3-148	Process report: Project initiation	
✓	SEP3-150	Process report: Project execution	
✓	SEP3-154	Process report: Apendicies	
	SEP3-164	SEP3-151 / Niklas	

1–8 of 8

Activity Stream

Your Company JIRA

Today

Anders Sønderby updated 2 fields of SEP3-166 - Rework test cases
- Changed the Remaining Estimate to '4 minutes'
- Logged '30 minutes'

17 minutes ago Comment Vote Watch

Anders Sønderby updated 2 fields of SEP3-166 - Rework test cases
- Changed the Remaining Estimate to '34 minutes'
- Logged '9 minutes'

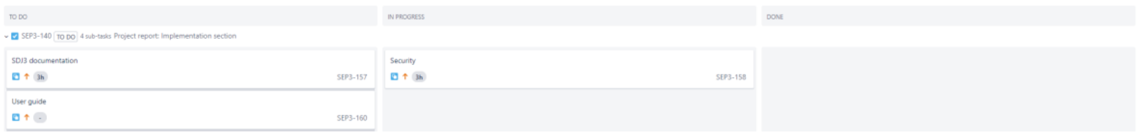
33 minutes ago Comment Vote Watch

Anders Sønderby updated 2 fields of SEP3-166 - Rework test cases
- Changed the Remaining Estimate to '43 minutes'
- Logged '17 minutes'

Figure 5-5 Personal dashboard

5.1.4.2 Duration

The creation of each sprint was specified by four-days duration, which were assigned to the agile boards (*the agile boards were customizable to some extent or fully by incorporating a third-party software*). These sprints contained our selected requirements, which then got derived into subtasks by the group using the Work Breakdown Structure. When a sprint was initiated as started, the subtasks got derived into swim lanes by default. The different swim lanes where divided into three columns, “TO DO”, “IN PROGRESS” or “DONE”. As seen in figure 5-6.


TO DO | IN PROGRESS | DONE

SEP3-140 TO DO 4 sub-tasks Project report: Implementation section

SDJ3 documentation 3h SEP3-157

User guide - SEP3-160

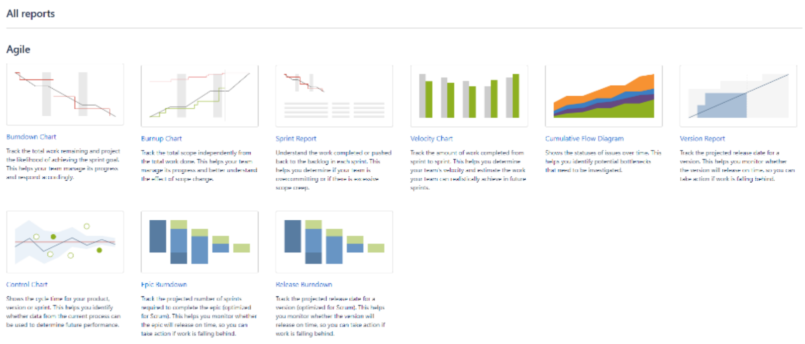
Security 3h SEP3-158

Figure 5-6 Active sprint swim lanes

Whenever the group had finished breaking a requirement into subtasks, a member would then be assigned to different tasks including the estimation of development. When the subtask was executed the spent time was registered, which then automatically re-calculates the burn up chart.

5.1.4.3 Reports

Jira provides various reports for an active project, which in most of the cases can be customized depend on which features is provided by Jira. As seen in figure 5-7. The generated reports can vary from the whole project to individual sprint.


All reports

Agile

Burndown Chart
Track the total work remaining and project the likelihood of achieving the sprint goal. This helps your team manage its progress and respond accordingly.

Burnup Chart
Track the total scope independently from the total work done. This helps your team manage its progress and better understand the effect of scope change.

Sprint Report
Understand the work completed or pushed back to the backlog in each sprint. This helps you determine if your team is overcommitting or if there is excessive scope creep.

Velocity Chart
Track the amount of work completed from sprint to sprint. This helps you determine your team's velocity and estimate the work your team can realistically achieve in future sprints.

Cumulative Flow Diagram
Shows the statuses of issues over time. This helps you identify potential bottlenecks that need to be investigated.

Version Report
Track the projected release date for a version. This helps you monitor whether the version will release on time, so you can take action if work is falling behind.

Control Chart
Shows the cycle time for your product, version or sprint. This helps you identify whether data from the current process can be used to determine future performance.

Epic Burndown
Track the projected number of sprints required to complete the epic (optimized for Scrum). This helps you monitor whether the epic will release on time, so you can take action if work is falling behind.

Release Burndown
Track the projected release date for a version (optimized for Scrum). This helps you monitor whether the version will release on time, so you can take action if work is falling behind.

Figure 5-7 Jira reports

5.2 1st Sprint

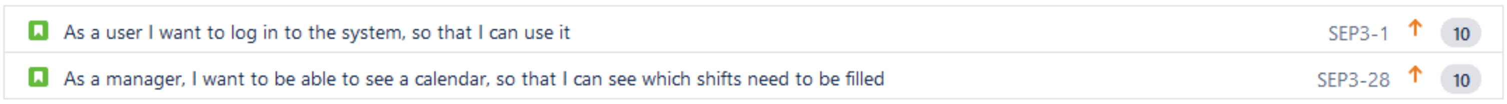
The first sprint was initiated on Wednesday the 13th of May and ended on Monday the 18th of May. It ran over the weekend; however, Saturday and Sunday were not counted as working days.

The total estimated workload for sprint 1 was 89 hours and 30 minutes.

5.2.1 Sprint Planning

The group started the 1st sprint with a meeting allocated for the sprint planning. As this was the first sprint as a group, we decided to go more in-depth than we normally would when discussing which user stories had priority and how we would approach the agile process regarding future sprints.

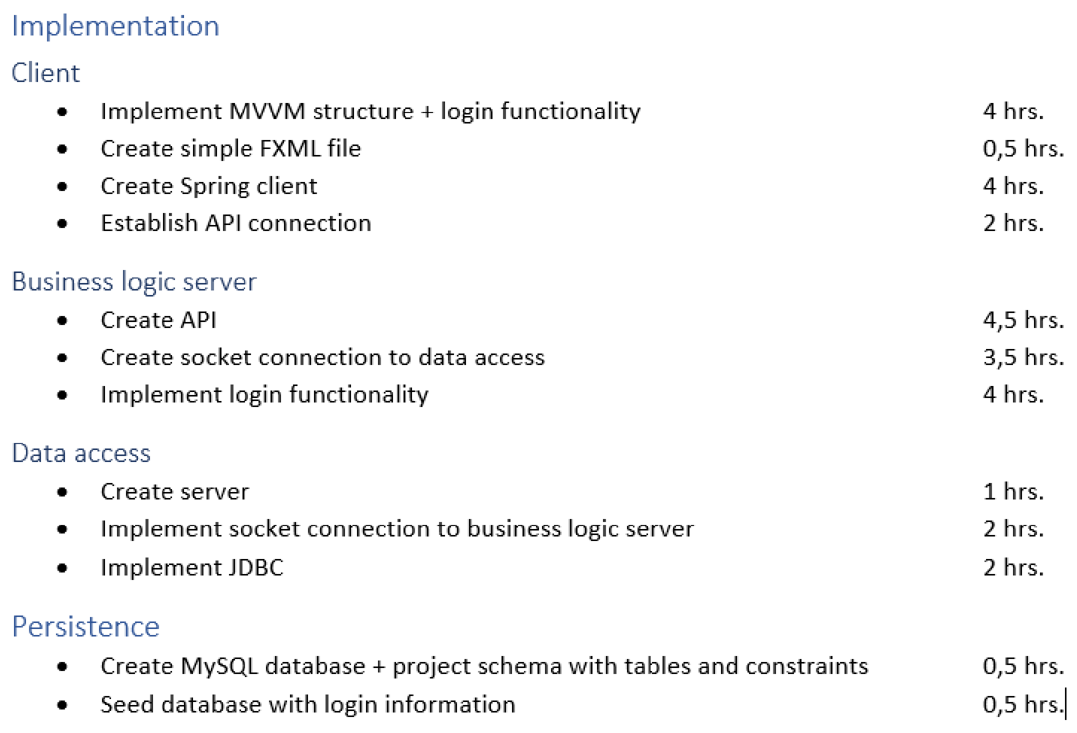
In the meeting, we discussed different user stories, specifically those who we deemed most important in terms of core functionality and structural value. The user stories we came to pick were the two "most important" user stories in terms of core functionality, namely the login and the calendar. The chosen user stories can be seen on figure 5-8.



As a user I want to log in to the system, so that I can use it	SEP3-1	10
As a manager, I want to be able to see a calendar, so that I can see which shifts need to be filled	SEP3-28	10

Figure 5-8 - Showing the two user stories chosen for Sprint 1

The user stories were then broken down into smaller, more manageable tasks using the Work Breakdown Structure (WBS) which we had experience with from the 2nd semester and for the first sprint, planning poker was used for the original time estimate.



Implementation

Client

Implement MVVM structure + login functionality	4 hrs.
Create simple FXML file	0,5 hrs.
Create Spring client	4 hrs.
Establish API connection	2 hrs.

Business logic server

Create API	4,5 hrs.
Create socket connection to data access	3,5 hrs.
Implement login functionality	4 hrs.

Data access

Create server	1 hrs.
Implement socket connection to business logic server	2 hrs.
Implement JDBC	2 hrs.

Persistence

Create MySQL database + project schema with tables and constraints	0,5 hrs.
Seed database with login information	0,5 hrs.

Figure 5-9 - Snippet of the WBS for sprint 1

After the time estimation (as seen in figure 5-9), the sprint was updated in Jira and the tasks were assigned to the group members.

The Work Breakdown Structure can be found in Appendix D.
The meeting minutes for the sprint can be found in Appendix B.

5.2.2 The Sprint

During the work period of the first sprint, the group kept up to date through the daily SCRUM meetings.

As mentioned in the meeting minutes, the group had a day-long argument over whether to switch to C# Blazor for the presentation tier, rather than staying with JavaFX, as was the original plan.

The group chose to stay with JavaFX, since all the members were more experienced with it, and it was part of the original architecture.

5.2.3 Sprint Review

In Sprint 1, 2 user stories were allocated for the sprint:

- User story 1 – User login
- User story 23 – Manager calendar

Both user stories were presented to the customer. They were satisfied with the user login, since it was implemented, tested, and worked as they expected it to.

The manager calendar was not yet satisfactory, since only the analysis, design and client-side implementation were finished. The customer told the development team that they should focus on the remaining implementation, namely the database queries as well as the API functionality.

The development team agreed to move the remaining calendar implementation to the next sprint.

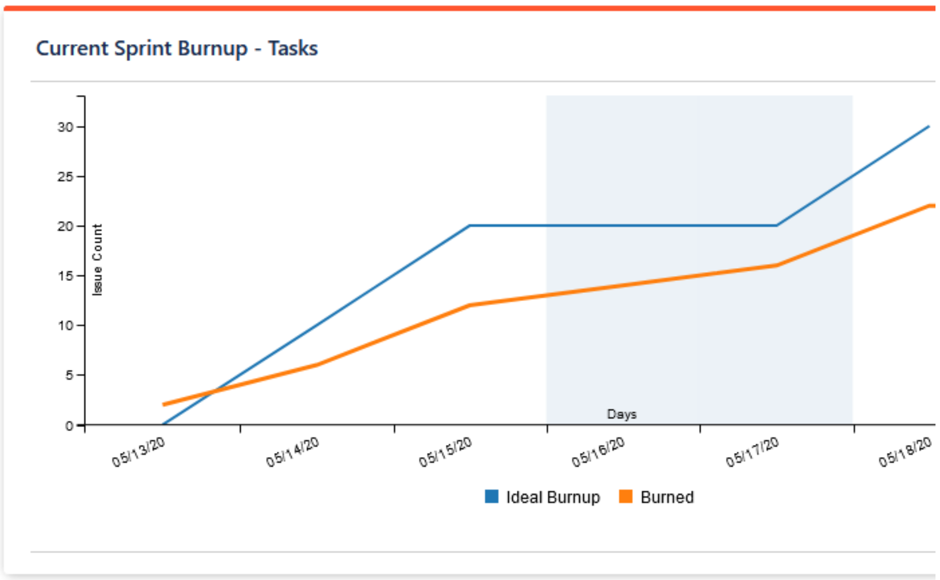


Figure 5-10 – Sprint 1 burnup chart by issues completed

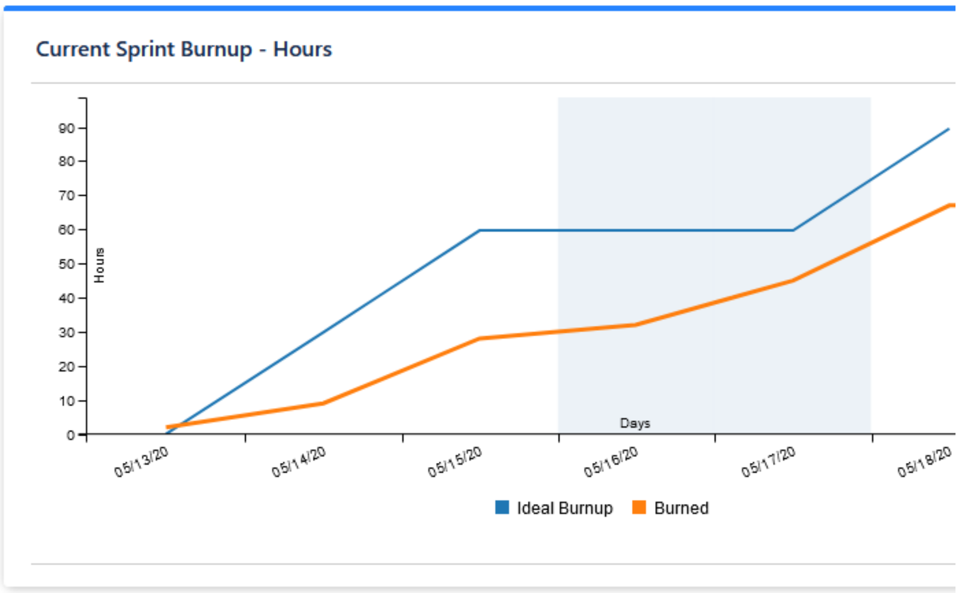


Figure .

5.2.4 Sprint Retrospective

The sprint retrospective meeting started off with a discussion about the teamwork during sprint 1.

The team agreed that the communication overall was not bad at all; although to avoid misunderstandings it was agreed that the communication via Discord needed to be slightly more precise, especially when explaining ideas for how to implement user stories.

During the sprint, the team had a significant argument over whether to switch the presentation tier to C# Blazor. As it was mentioned in the review, the group chose to stick to the original plan with JavaFX.

Moving forward from sprint 1, the team agreed to be better at notifying each other when a task is finished, specifically in cases where the implementation of one feature is reliant on the task of another team member.

Another agreement was that significant conflicts, like the one experienced during the sprint, should be solved over a call, instead of over text messages since those could be misunderstood and cause resentment.

The use of Git branches was another topic during the retrospective meeting. It was agreed to create feature-specific branches instead of branches named after each team member, so that they will always know which branch to pull to have the latest version of the implementation for a specific feature. It was also demonstrated how to solve merging conflicts manually.

SCRUM-wise, the team found it important to update the hours spent on a specific task immediately after finishing it. That would give the most accurate numbers in terms of time spent, instead of having to retroactively remember how long it took.

As mentioned in the sprint planning, the time estimates for the tasks found with WBS was done through planning poker. The team agreed that it was a good way to get to know what each team member thought of the work required for each task. The downside to this was that it took a long time to do.

For the following sprints, the team decided that after designating tasks found with WBS, the team member responsible was also responsible for creating the time estimations.

A technical discussion about tasks versus sub-tasks in Jira was also brought up for future sprints.

Lastly, the team concluded that they had been very good at helping each other, and that should be continued. The team had been good at putting in effort and hard work with the tasks that were assigned during the sprint.

The Work Breakdown Structure used for breaking down the user stories into smaller, more manageable tasks was deemed very effective, and it was agreed to continue using it for future sprints.

5.3 2nd Sprint

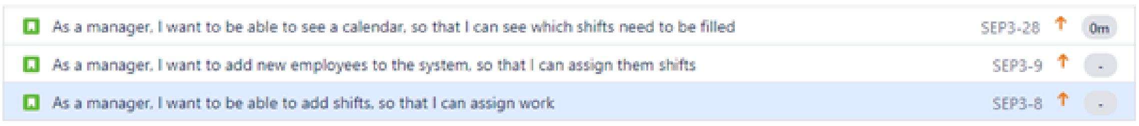
The second sprint was initiated on the 19th of May 2020. It ended on the 22nd of May 2020. The total estimated time required for sprint 2 was 90 hours and 30 minutes. This sprint included unfinished tasks from first sprint.

5.3.1 Sprint Planning

The sprint planning for sprint 2 was shorter compared to the first sprint planning.

During sprint planning the leftover task from the first sprint has been taken in consideration and thus 2 user stories has been chosen to be processed during this sprint.

In Jira, those two user stories are shown as SEP3-8 and SEP3-9. They were chosen based on their backlog rating.
The team further discussed those user stories and concluded that they were core features and thus necessary in further development of the software.



As a manager, I want to be able to see a calendar, so that I can see which shifts need to be filled	SEP3-28	0m
As a manager, I want to add new employees to the system, so that I can assign them shifts	SEP3-9	-
As a manager, I want to be able to add shifts, so that I can assign work	SEP3-8	-

Figure 5-10 Selected user stories for Sprint 2

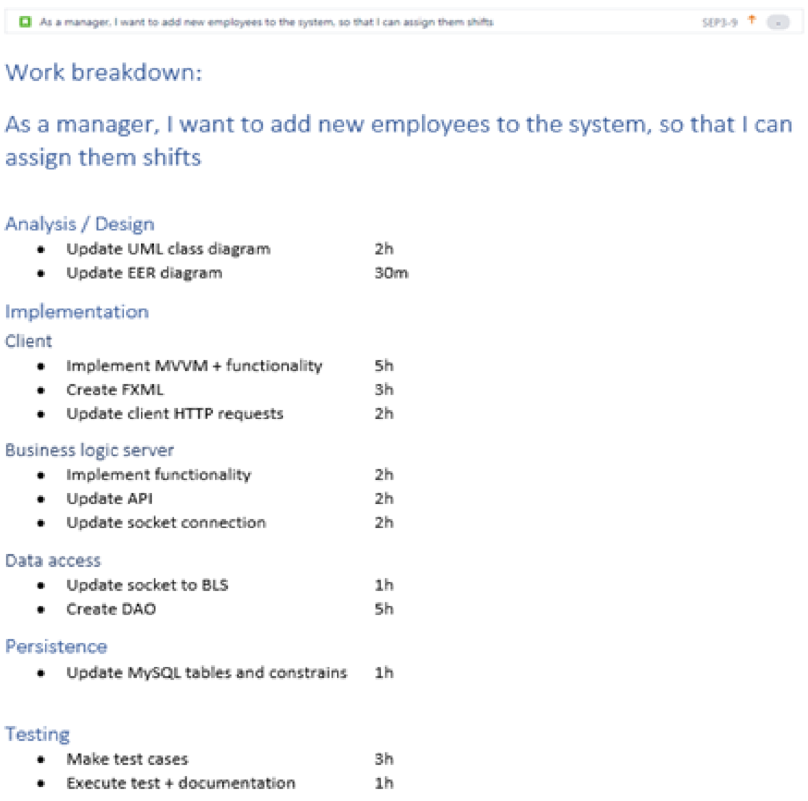
During the first sprint retrospective, it was decided to use a different method to estimate time required on tasks. The decision was to for each team member to personally estimate the time needed to finish assigned tasks.

This proved to be helpful, especially when taking in consideration of the different personal levels of technical knowledge and work speed. This shaved approximately 1-2 hours from planning.

During our WBS, each member tried to choose something they hadn't tried working with yet.

This helped us further understand our 3-tier architecture and how it works throughout the system.

The WBS looked like the one used in the first sprint, since it proved to be effective and informative.



As a manager, I want to add new employees to the system, so that I can assign them shifts SEP3-9

Work breakdown:

As a manager, I want to add new employees to the system, so that I can assign them shifts

Analysis / Design

Update UML class diagram	2h
Update EER diagram	30m

Implementation

Client

Implement MVVM + functionality	5h
Create FXML	3h
Update client HTTP requests	2h

Business logic server

Implement functionality	2h
Update API	2h
Update socket connection	2h

Data access

Update socket to BLS	1h
Create DAO	5h

Persistence

Update MySQL tables and constrains	1h

Testing

Make test cases	3h
Execute test + documentation	1h

Figure 5-11 WBS for user story 9

Lastly, the team started to use "sub-master" branches for Git, which are user story specific branches. They were meant to be the most actual version of code of each user story.

5.3.2 The Sprint

The start of the sprint was the most productive period. Finishing some tasks from first sprint while doing new task from other user stories.

During the beginning of the sprint, the functionality of calendar view was proven to be a bigger challenge than initially expected. This dragged all the way to end of the 2nd day.

The team also had some miscommunication, when it came to what needed to be fetched and sent to database this had delayed our progress. We had to adapt to the client, and thus changed our JDBC input to match it.

On 3rd day, the team experienced a huge set back due to GitHub versioning. Some pushed changes were not actually pushed to the branch, thus resulting in time loss because we needed to manually merge all the files and double check if all methods and classes which we had implemented were in master.

On the 4th day of the sprint, the focus was to finish what we could, but due to problems with Git it was decided to update the master branch from the most recent version of the code.

5.3.3 Sprint Review

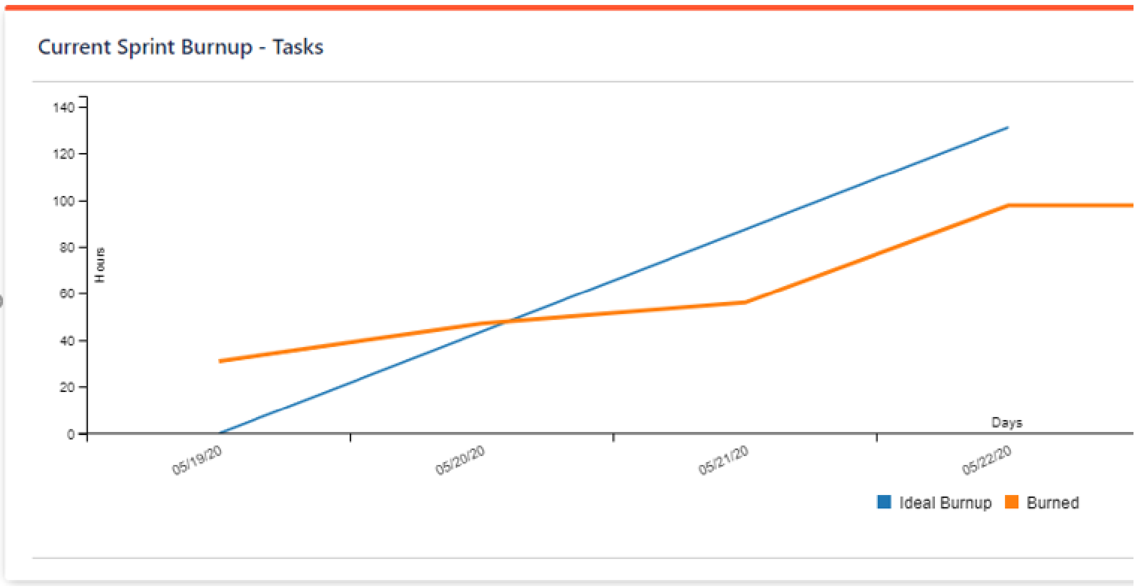


Figure 5-12 Sprint 2 burnup chart – Tasks

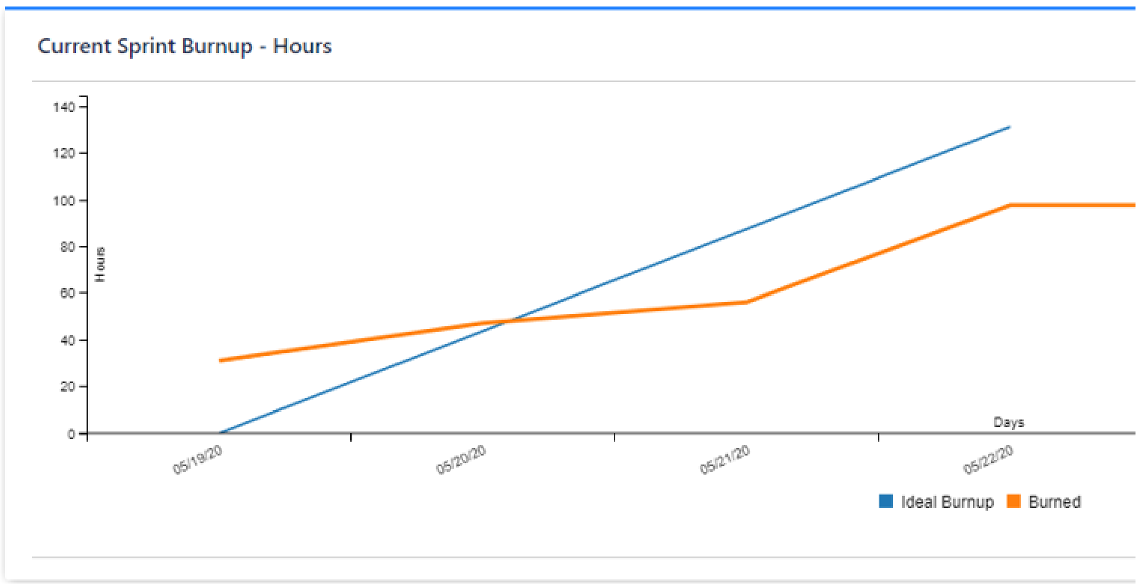


Figure 5-13 Sprint 2 burnup chart – Hours

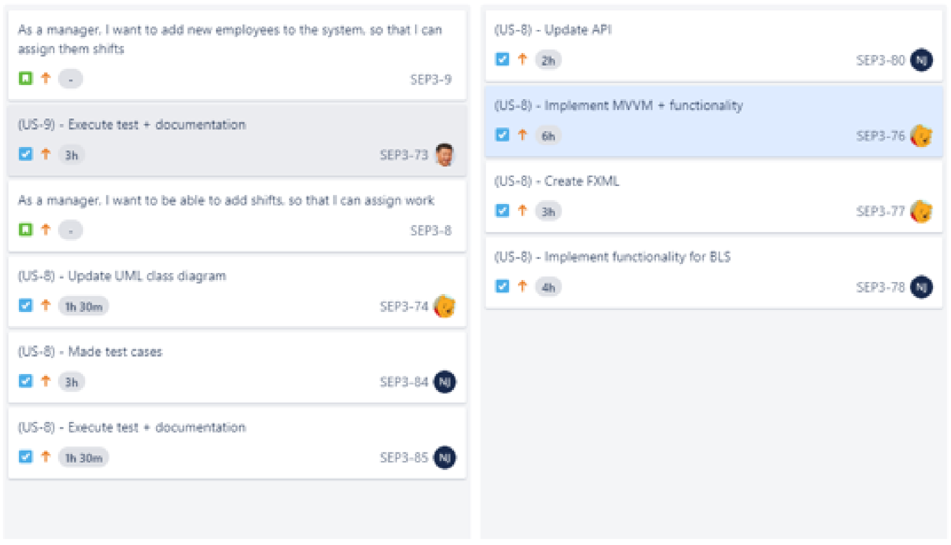
Due to a mishap happening with Git, we were not able to deliver the expected result of the sprint.

The tasks shown in the figure below were unfortunately not implemented in time. Testing for user stories 8 and 9 were not made, since the implementations were not ready to be tested. These tasks were pushed to next sprint as a priority task.

The Calendar view was successfully implemented and as a core of the software this was crucial. The calendar could be loaded with years between 2019 – 2026 and more years could be added later if needed.

All the months were able to be retrieved.

The results of the sprint were presented to the customer, who was not satisfied with the final output of this sprint, since 2 user stories were not finished.



As a manager, I want to add new employees to the system, so that I can assign them shifts
- SEP3-9

(US-9) - Execute test + documentation
3h SEP3-73

As a manager, I want to be able to add shifts, so that I can assign work
- SEP3-8

(US-8) - Update UML class diagram
1h 30m SEP3-74

(US-8) - Made test cases
3h SEP3-84

(US-8) - Execute test + documentation
1h 30m SEP3-85

(US-8) - Update API
2h SEP3-80

(US-8) - Implement MVVM + functionality
6h SEP3-76

(US-8) - Create FXML
3h SEP3-77

(US-8) - Implement functionality for BLS
4h SEP3-78

Figure 5-14 Left over task which will be left for Sprint 3

5.3.4 Sprint Retrospective

During the sprint retrospective meeting, the team decided to go back to Master and personal branches for development with Git version control.

The Work Breakdown Structure was kept the same for upcoming sprints, since it was proven to be more efficient than the planning poker used initially. This resulted in less communication, so the team decided to plan more thoroughly and discuss potential ways of implementing the system. This was done to make sure that the whole team is on the same page.

The meeting concluded with encouragement of direct communication with team members who are working on same user story.

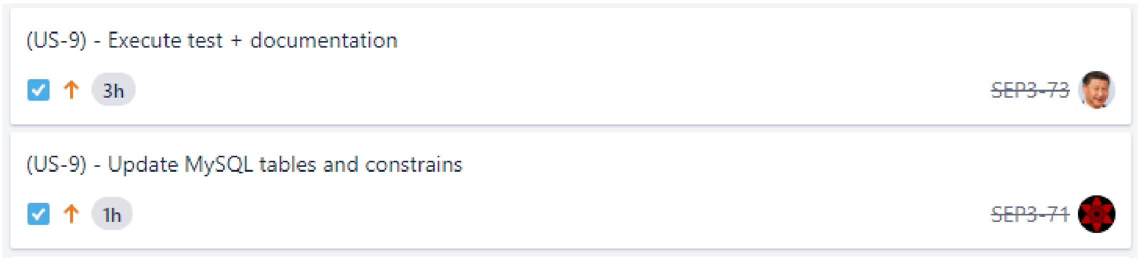
5.4 3rd Sprint

The third sprint was initiated on May the 25th and ended 28th of May. The total workload was summed to 108 hours during this sprint.

5.4.1 Sprint Planning

Hence, the sprint planning meetings went well the past sprints, we decided not to change the approach of the Work Breakdown Structure. During the estimation of workload, we decided instead to scrap the planning poker estimation method and individually estimate the tasks assigned, since everybody had improved their knowledge and skills towards the implementation parts in the different tiers of the project. It was a test scenario towards being more efficient and less time consuming while planning the tasks for the upcoming sprint. This meant that we would have an advantage when starting on the first requirement implementation.

Since user story 23 *(As a manager, I want to be able to see a calendar, so that I can see which shifts need to be filled)*, 8 *(As a manager, I want to add new employees to the system, so that I can assign them shifts)* and 7 *(As a manager, I want to be able to add shifts, so that I can assign work)* was not fully implemented due to complications and underestimation of workload and would therefore have a higher priority.


(US-9) - Execute test + documentation
✓ ↑ 3h ~~SEP3-73~~

(US-9) - Update MySQL tables and constrains
✓ ↑ 1h ~~SEP3-71~~

Figure 5-15 - Snippet of unfinished tasks from user story 8

Figure 5-15 shows a snippet of some unfinished tasks, which have been carried into Sprint 3 from the previous sprint.

After further discussion upon the best appealing and providing a greater architectural structure to establish the UI (interface) and further progression.

User story 4:

- *(As a manager, I want to see a list of employees which I supervise, so that I can get an overview of who is available for work)*

User story 5:

- *(As an employee, I want to see a calendar with my shifts, so that I can get an overview of when I need to work)*

User story 13

- *(As manager, I want to be able to edit shifts, so that the information is up to date)*

Where selected to provide a further expansion upon the implementation that have already been fulfilled. As mentioned above, the assigned group member would provide the estimation by themselves to ensure a more specified and accurate measurements. Since some of the estimations done in previous sprints with Planning Poker was to some extent underestimated or had a higher complication than initially thought. We thought with our previous Semester project experiences and knowledge it would be the better approach, since we individually have different skill and knowledge between subjects.

The Work Breakdown Structure can be found in Appendix D.
The meeting minutes for the sprint can be found in Appendix B.

5.4.2 The Sprint

A daily scrum meeting kept the team updated, according to what was finished, needed more attention, in progress, complications and what the daily approach would be per group member. As seen in appendix B – Minutes. The team experienced a minor setback due to lack of communication between creation of various tier classes and methods, which resulted in reconstruction. The reconstruction had to be made to ensure only the needed information would be retrieved form the Database to the Client UI.

On the third day during implementation of user story 13 *(As manager, I want to be able to edit shifts, so that the information is up to date)*, a group member figured that the complexity was further beyond what we had expected and estimated. Because a whole reconstruction on the calendar UI and methods would have been necessary to construct that specific user story. A meeting was settled during the evening to discuss the complexity and approach, unfortunately a deeper insight made it clear an implementation would be too time consuming considering the remaining time until the project deadline. This meant we had to push the user story into the future of the project, without further progression on user story 13, but instead include a couple extra user stories to the sprint:

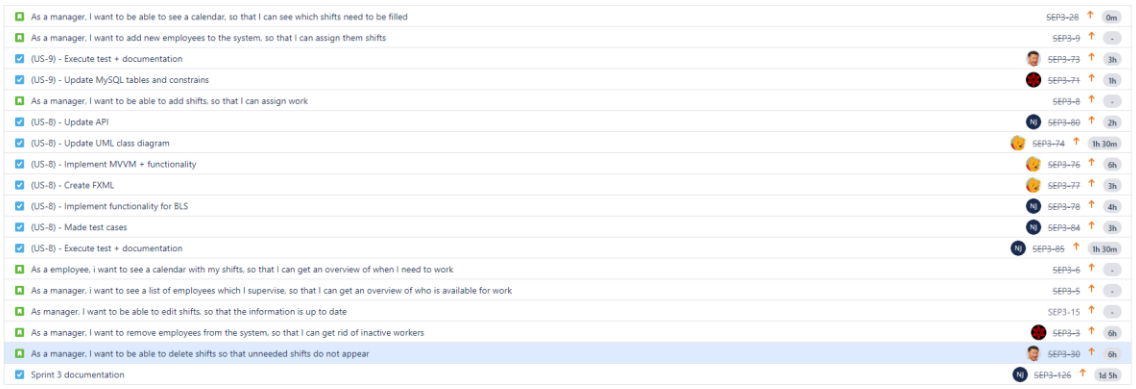
User Story 21

- *(As a manager, I want to be able to delete shifts so that unneeded shifts do not appear)*

User story 2

- *(As a manager, I want to remove employees from the system, so that I can get rid of inactive employees)*

Those user stories where selected to increase the implementation integrity, progression and to gain further closure to the remaining hours of workload. The estimation where decided as a team to 6 hours each since most of the implementation had similarities to previous implementations. As seen in figure 5-16, the full spectrum of sprint 3 tasks.



As a manager, I want to be able to see a calendar, so that I can see which shifts need to be filled		~~SEP3-28~~	0m
As a manager, I want to add new employees to the system, so that I can assign them shifts		~~SEP3-9~~	-
(US-9) - Execute test + documentation		~~SEP3-73~~	3h
(US-9) - Update MySQL tables and constrains		~~SEP3-71~~	1h
As a manager, I want to be able to add shifts, so that I can assign work		~~SEP3-8~~	-
(US-8) - Update API	NJ	~~SEP3-80~~	2h
(US-8) - Update UML class diagram		~~SEP3-74~~	1h 30m
(US-8) - Implement MVVM + functionality		~~SEP3-76~~	6h
(US-8) - Create FXML		~~SEP3-77~~	3h
(US-8) - Implement functionality for BLS	NJ	~~SEP3-78~~	4h
(US-8) - Made test cases	NJ	~~SEP3-84~~	3h
(US-8) - Execute test + documentation	NJ	~~SEP3-85~~	1h 30m
As a employee, i want to see a calendar with my shifts, so that I can get an overview of when I need to work		~~SEP3-6~~	-
As a manager, i want to see a list of employees which I supervise, so that I can get an overview of who is available for work		~~SEP3-5~~	-
As manager, I want to be able to edit shifts, so that the information is up to date		SEP3-15	-
As a manager, I want to remove employees from the system, so that I can get rid of inactive workers		~~SEP3-3~~	6h
As a manager, I want to be able to delete shifts so that unneeded shifts do not appear		~~SEP3-30~~	6h
Sprint 3 documentation	NJ	~~SEP3-126~~	1d 5h

Figure 5-16 Sprint 3 task spectrum

5.4.3 Sprint Review

In the sprint review meeting, user story 2, 4, 5, 7, 8, 21 and 23 was presented and showcased for the customer. All user stories where completed to the customers satisfaction and quoted as “Finished”, besides the showcasing test cases where shown and explained for greater understanding the different scenarios that have been tested as “Successful” including which was expected to provide a specified result.

User story 13 was pushed into the project future and therefore not able to present nor showcase. Explanation were given due to reconstruction of the main architecture of the calendar view, would exceed the time period before deadline and therefore had to paused for implementation until the project future. Sprint 3 burnup charts can be seen on figure 5-17.

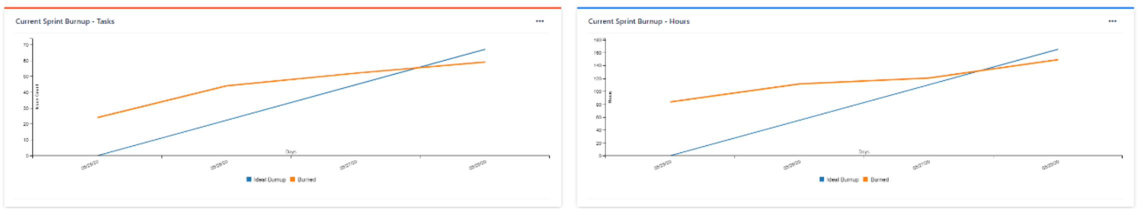


Figure 5-17 Sprint 3 burnup charts by both issue-completion as well as hours

5.4.4 Sprint Retrospective

During the retrospective meeting, the team discussed the estimation done by the assigned group member, which returned a positive agreement on continuing with the same strategy. Secondly, while during the planning meeting more expensive detailing needed attention to avoid minor setback due to miscommunication nor different opinions towards what information was necessary to either send or retrieve to/from the database and have a common objective.

A common complaint about the estimation needed to be done before the sprint would be set as ‘started’ on Jira to prevent the burn up/down chart to look eccentric.

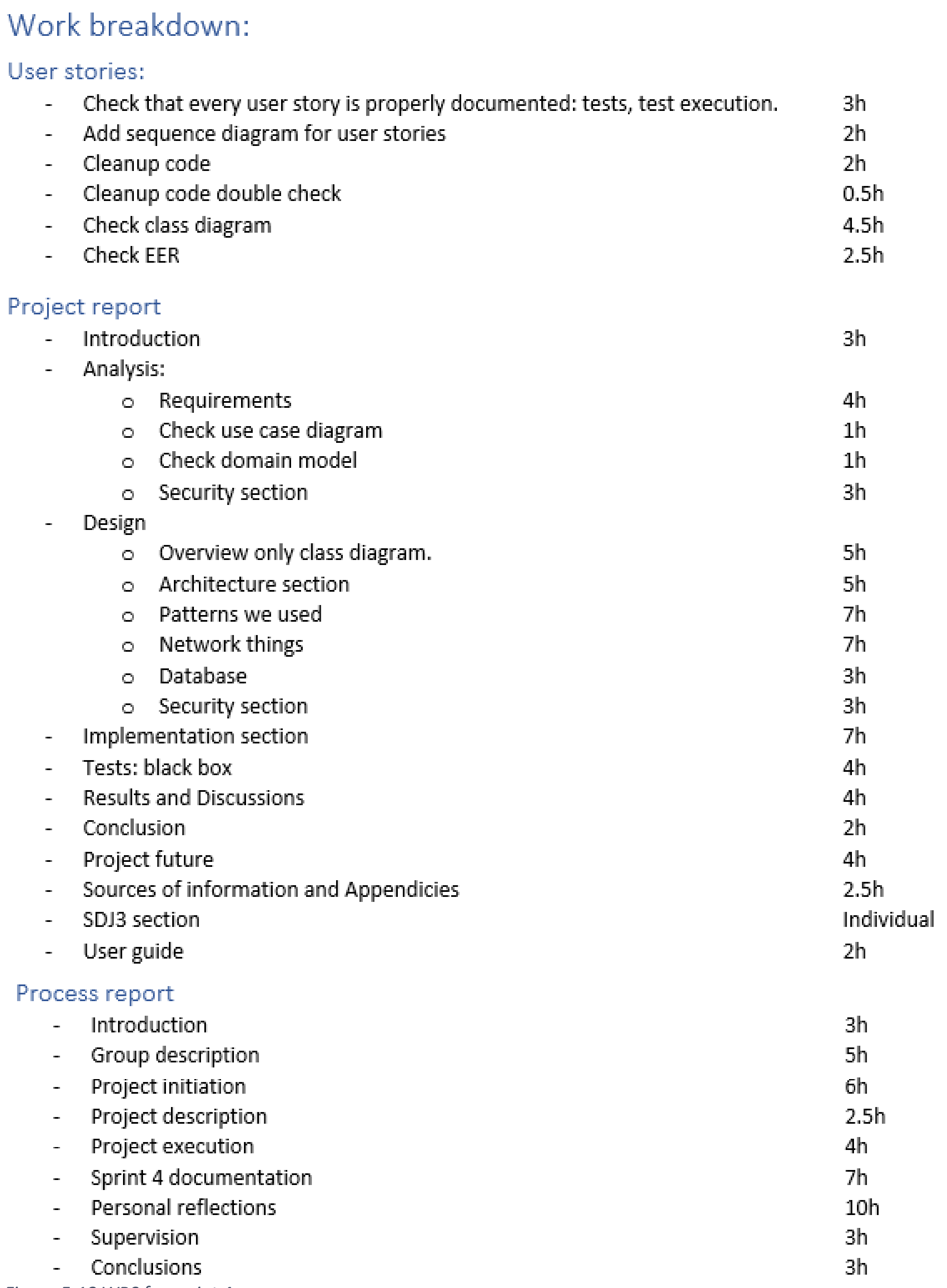
Another suggestion was proposed in the business logic tier, to divide a class into smaller classes so it would become SOLID. The suggestion was agreed upon and was instantly constructed after the meeting.

5.5 4th Sprint

The 4th sprint was initiated on May 29th and ended on June 3rd. The sprint is in the days of the weekend, those days are not counted as sprint days. The total hours of upcoming work: 127.5 hours.

5.5.1 Sprint Planning

After a mostly successful third sprint the team decided to freeze development of new features in the fourth sprint. The focus was put on code cleanup, fixing loose ends in the system and checking documentation of developed features. Due to this no, new user stories were picked from the backlog.



Work breakdown:

User stories:

- Check that every user story is properly documented: tests, test execution.	3h
- Add sequence diagram for user stories	2h
- Cleanup code	2h
- Cleanup code double check	0.5h
- Check class diagram	4.5h
- Check EER	2.5h

Project report

- Introduction	3h
- Analysis:	
o Requirements	4h
o Check use case diagram	1h
o Check domain model	1h
o Security section	3h
- Design	
o Overview only class diagram.	5h
o Architecture section	5h
o Patterns we used	7h
o Network things	7h
o Database	3h
o Security section	3h
- Implementation section	7h
- Tests: black box	4h
- Results and Discussions	4h
- Conclusion	2h
- Project future	4h
- Sources of information and Appendicies	2.5h
- SDJ3 section	Individual
- User guide	2h

Process report

- Introduction	3h
- Group description	5h
- Project initiation	6h
- Project description	2.5h
- Project execution	4h
- Sprint 4 documentation	7h
- Personal reflections	10h
- Supervision	3h
- Conclusions	3h

Figure 5-18 WBS for sprint 4

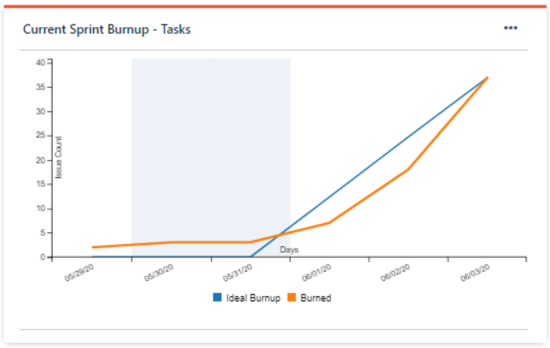
While planning the next sprint we forgot to add 4 tasks to the breakdown. Security documentation, SDJ documentation, user guide and project conclusion. They were added into the sprint the same day, except user guide which was added on the third day of the sprint.

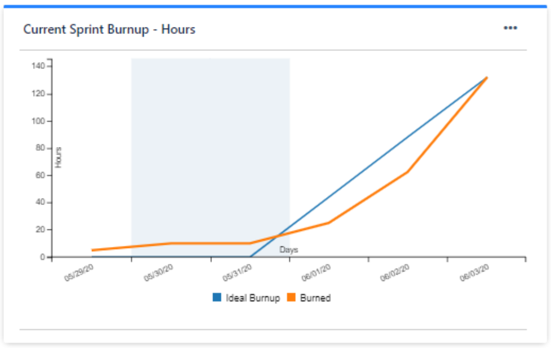
5.5.2 The Sprint

Some of the user stories could not be completed early because they needed diagrams in them that were not yet checked. There was a lack of prioritization in areas that we addressed. In the daily meeting there was a different view of the situation, everyone seemed to be working on a lot of things at the same time, developing a task over time till they are happy with it. There was a lot of functionality correction happening, although it was clear the system was not perfect. There were some bugs just could not be recreated to fix them.

There were some user stories that did not show up in the previous sprints that were not yet set as completed, even though they were fully implemented and tested. These user stories are 14 and 19. We made an error and overlooked them when we were planning the second sprint and did not pick them from the backlog. We instead finished them when we were doing the bigger user stories as we decided it would be very easy to implement them at the same time. The user story 14 was implemented when we implemented add shift user story. The user story 19 was implemented when we implemented the employee list because it required a table to see the employee's, so it made sense to see the details. It is a bad practice to decide to add a user story to a sprint and not add a notice of that event happening.
On the third day of the sprint progress was accelerating, but we were working a lot of overtime on the tasks, particularly security which was a rabbit hole to document. There was a sign of task underestimation.
At the start of the fourth day the team was almost finished with the tasks. The sprint was finished in the evening in order to have time to complete all task.

5.5.3 Sprint Review





The sprint resulted in a lot of small GUI changes that improve user experience and make the system more intuitive to use. The source code was cleaned up and made ready for further functionality implementation. The system crashes that the team was able to recreate were fixed. The documentation of the systems current functionality was finished.
The results of the sprint were presented to the customer. It was noted that the customer is satisfied that the system is more stable but wants us to go back to implementing features for the system.

5.5.4 Sprint Retrospective

The results of the sprint 4 retrospective.
The team should handle documentation work breakdown carefully, it is very different from how implementation breakdown works and should have more time dedicated to it. It would also help estimating the time to complete documentation tasks along with being more efficient with the time we have. The time should also divide tasks into smaller pieces.
The team should agree on a documentation style before starting the sprint so editing.
The team should continue the fast and direct communication that we used this sprint.

6 Personal Reflections

6.1 Anders Sønderby Sørensen

I think this semester project has been a good experience overall. I started off the semester in an entirely new group, compared to the last 2 semesters, where I had been with the same group. I was, however, very confident in the group since I have become quite good friends with all of them over the last year and a half and they all seem dedicated to doing good in school.

The project started off very well with the group. We met the initial deadlines and did good work on the initial aspects of the project.

After that though, COVID-19 hit us, and we were forced to work from home. I am much more comfortable having project meetings in person since it is much easier to look at people while having discussions. Despite this, we kept working well together, which I was very happy about.

A major personal setback caused by the COVID outbreak, is that I have consistently had a very hard time concentrating on schoolwork when home alone in my apartment. This caused me to be less efficient than I would usually be.

Even though I completed my tasks during the project execution phase and during the sprints, I felt like I had to spend more time on the tasks than I would usually do.
Usually I would migrate to either the public library or the one we have at VIA, but they were understandably closed to minimize the risk of infection.

I noticed that working with SCRUM and agile development was much easier this semester than the last. While we initially had to spend time making sure we were all "on the same page", a lot of the doubts of whether we were doing the right thing was not very prevalent, if at all. Introducing an issue-tracking tool like Jira made keeping track of the sprints and user stories a lot smoother than last semester, where we had the information scattered across several MS Word documents.

Development wise, I felt like we did a decent job. While there were some “bumps on the road”, it ultimately felt like a “smooth ride”. While we were only 2 developers in my previous group, having 4 dedicated developers in a group felt like a lot more work was getting done.

6.2 Dat Phuc Nguyen (David)

We have started this semester in new study group only person I worked with was Niklas who I work with since 1st semester and we have developed good synergy. I knew other team members but have never worked with them. At beginning the work was not hard since we could talk to each other and we were on friendly terms before joining in group.

But This semester has been harder than expected due to COVID-19 quarantine. We could not meet with our teammates and this has been detrimental to me as a person who prefer face to face communication than over text or call. Due to this we have had few miscommunications and arguments.

Also, online classes were not ideal for my liking. I felt like there was not enough incentive to study as hard as I would have if I were to visit regular classes. Since some classes were not ideal. So, my overall knowledge from this semester is lacking compared to what I was able to learn in first and second semester. Even though the third semester is considerably harder.

During our project execution phase I fell a bit behind when it came to development in webservices. But observing the work other made before me I was able to catch up. But still everything took longer than expected.
In our first sprint we had argument about what to use for GUI in our Client. This argument

spiraled out too much for what it was but, in the end, we went for safer chose of Java for GUI. After that everything went almost smoothly with some minor roadblocks.

Overall, I am satisfied with result which we were able to achieve even though we faced challenge after challenge. This has been very interesting experience and I have learned a lot about myself and same about my teammates who were always positive and helping each other. I am glad for choosing this team.

6.3 Niklas Krogh Jensen

As the semester started, I was seeking a new study group along with David, which I have worked with since 1st semester project. We had developed a good understanding towards each other's expectation and skills and wanted to find fellow students which would connect personally with and had similarities of expectations how a semester project should be performed.

The study group was settled with two fellow students which I had been talking with gotten to know a bit personally during the education. All of us bounded well personally as well during the beginning of the project.

Unfortunately, the COVID-19 pandemic made this semester difficult, since I am not a big of schooling from home through a computer. I prefer interaction with the study group, fellow student and being able to sit in a classroom, where distraction is minimal. During the quarantine and online classes, I constantly felt hanging behind the wagon, even though I attended all classes with dedication, which caused a minor motivation loss.

Fortunately, our bonding within the study group kept the heads high and managed to stay on track with the project during the analysis phase. As the project started to take its structure minor discussions occurred but got settled quickly with a common approach.

During the execution of the project, different opinions occurred upon whether the architectural structure should be changed or kept as the proof of concept were provided. After a further discussion back and forth which approach would be the best interest for the project, a settlement came sticking to the initially concept.

Overall, has this semester been a challenge to keep pushing forward and continuously staying motivated. Although I am more than satisfied with the result we have achieved during this exceptional experience with quarantine. It has been without a doubt an absolute pleasure working this studying group through good and rough times.

6.4 Rokas Barasa

This was a very hard semester for me. I failed last semester project due to a team that did not know how to implement and were not paying attention to lessons. This semester my goal was to fix what I was doing wrong in the last semester and to find a good team to work with so I would be the only one doing the work. I joined a completely new team that that had 2 members who have worked together before. I knew all of these people before. It was nice finally being in a team with people that are competent in what they do. They are very efficient in project planning; I did not feel constantly bothered by project things and actually had time to talk to them as friends instead of coworkers. There were no arguments at the start, and we agreed on decisions quickly. It felt nice not to redo a backlog 5 times.

The coronavirus situation did affect this semester. I quickly got used to studying from home even though I very much did not like it. Everything took about twice the time it would normally take to understand concepts.

When the sprint period started, what was in my head was that I was not going to fail this semester project and that meant I was extra careful with things. I did not want to make big risks. One thing I very much did not like about how we did things was half a month after we agreed on a plan for the system conversations started again on changing the architecture, in the middle of the first sprint. That the proof of concept period felt like a of wasted time. Decision were being made at the wrong time, on promises that the grass is greener on the other side and a lack of wanting to stick to a plan. All of this was against my current mindset of playing it safe and making me feel very anxious about the project future. After resolving the disputes work went smoothly. I still had the urge to make the system we were making perfect, constantly trying to improve the code, and working overtime. That is a bad thing because the stress of it started messing with my head, I was very argumentative, and I feel like I took lead in system architecture decisions too much. Besides that, everything went smoothly. I think there were more problems happening with me rather than my team mates this semester. I was having trust issues, doubting if my team members could finish some tasks, but I was proven wrong. We completed a lot of user stories by my standard and the system has basic functionality that I would consider a working employee planner.

In conclusion I would say that I enjoyed working with this team.

7 Supervision

The feedback is going to be divided by teacher, to avoid mix ups.
Joseph – We did not get a very good first impression of him as a semester project supervisor when he did not provide us with our feedback for the project description, something that was scheduled to happen. We had to ask for it to be provided. There were no asking questions of what he could do to help and what we do not understand, even though that is kind of acceptable, because a student should ask questions if he needs anything, it would have been nice if he did that. He seemed unavailable at the beginning of the project. There were difficulties communicating. He did not understand the questions we were giving him, and the answer would be something that was not the problem we were having. All of this happened in the pre sprint period. When the sprints started, he started being available and giving good and fast feedback when we asked questions. Although we asked less questions from him than we should have because of the bad experience before, there were things we asked Nicolai that we should have instead asked Joseph. We think he is not informative enough, there were always some things that were left on the table.

Nicolai – Most of the time he provided fast and good responses to questions if he knew something about them. We were given good ideas, feedback and Nicolai was asking other teachers if there was something, he did not have good knowledge about. We noticed he doubted himself a lot and did not take charge when Joseph was not available. And was not confident in his knowledge of what the project must look like. But this is understandable since it is his first semester project where he is a supervisor.

In conclusion, it was a hard semester, there were a lot of things to handle, like the COVID-19 situation that made teaching very different. We got most of the information we needed to finish the project.

8 Conclusions

This project can be assessed as successful considering the situation in which project was produced. Due to COVID-19 outbreak the team was not as cohesive as it would have been if there were daily meeting and face to face interactions. This could have made whole process easier and we could have voided many mistakes and miscommunications. Due to that team has misjudged the scale of the project and ended up with few unfinished requirements.
SCRUM was used as a management process to develop the system. The team was following it but throughout the development period some parts of it were simplified. This was needed due to lack of time. Work break down structure was simplified by reducing planning poker which estimated time of each task to individual estimation of assigned tasks.
Team was unsatisfied with Jira to manage the SCRUM process. For next semester the group would choose YouTrack instead which has similar functionality but its more profound and modifiable.

Overall, we have succeeded in our objective and went through hard times which has provided us experience for upcoming semester projects.

Appendices

Appendix A – Group Contract.

Appendix B – Minutes.

Appendix C – Product Backlog

Appendix D – Work Breakdown Structure.

Appendix E – Project Proposal.